Department of Social Systems and Management

Discussion Paper Series

No.1215

ビジネスプロセス工学序説

by

佐藤亮

August 2008

UNIVERSITY OF TSUKUBA

Tsukuba, Ibaraki 305-8573

JAPAN

ビジネスプロセス工学序説

佐藤亮

筑波大学

目次

まえがき

「いったいどれぐらいの在庫をもてばいいんだろう。どれほどの設備の用意があれば適当なんだろう。」

この問いは古くて新しくて応えるのが難しい。

多品目を生産し仕入れと販売だけではなく流通も含むビジネスプロセスを相手に、最近の情報技術を取入れたり経験をたよりに試行錯誤的に調整したりするわけだが、ビジネスプロセスの組み方の可能性があり過ぎるし、需要も計画からはずれて変動する。

情報システムを自社開発したり、基幹情報システム(ERP)を導入しても、あるいは、需要予測ソフトウエアや SCM ソフトウエアを導入してもコトは簡単には片付かない。ある部分の在庫量がかなり改善したところもあれば、人には言えないが導入・維持のコストの割には効果がいまひとつというところも珍しくないはずである。企業が顧客に提供するものは歯磨きやパソコンや自動車などのモノだけではなく、サービスやシステムソリューションも含まれる。セキュリティソフトウエアのダウンロードサービスや、製品保守時の部品入手の早さや予約の確実性、さらには、e-マーケットプレイスを通した企業間での営業代行や MRO 購買代行や経理外注などの業務外注化、個人生活での保険選びや商品選択支援の支援サイトのサービスもビジネスの対象になる。

製品別の事業部制とかカンパニー制組織でも、営業や製造の職能別組織でも、ビジネスプロセスが作られていてはじめて機能する。ビジネスプロセスの構造が企業全体のパフォーマンスにとって最大級の重要性を持つのである。

こうしたことから、初めに挙げた問は次のようにブレークダウンされる。

「いったいどんなビジネスプロセスを作ればビジネスが滞らずにスムーズにできるか。」

「どういう計画・管理の仕組みと情報システムでいこうか。」

「精密・的確な仕組みや評価を得るためのビジネスプロセスの性能を設計できるだろうか。性能評価のためのビジネスプロセス方程式は可能だろうか。あるとすると、どういう問題状況でどんな現実的解答をえられるのか。」

適切な概念なしでこれらのことを直感で場当たり的に考えるのは限界がある。たとえば、サプラインチェーン・マネジメント関係の本や記事によく見かけように、「顧客側の下流で発生したデータをチェーン全体で共有しましょう。」と

言われる。共有するとはどういうことか。単に日々の需要が見えれば、それで解決するのか。需要を見せるのは簡単だが、計画システムに取り込むにはどうやるのか。そして、その結果として、在庫やリードタイムのような特性がどういう状況でどれほど改善されるのか。情報システムを導入するとどういうふうにパフォーマンスが改善されるのか。こうしたことは、考えるための概念的コンセンサスがないことも手伝って明確ではないのである。

いわゆるケーススタディが、いくつかの事例や調査から一種の法則を見い出し、さらにそれ以降の事例によって法則を「検証」しつつ練り上げを行っていくというものだとすると、本書の内容はその対極にあるものになった。本書はまず、ビジネス実行を可能にしているビジネスプロセスのモデルを示し、モデルの特性や応用例を示す。IT・情報システムがビジネスプロセスで占める位置づけも行われる。ビジネスプロセスのモデルはファイルシステムを持つ離散事象システムとして提示される。さらに、離散事象システム自体については、物理システムで使われる連続変化を扱う微分方程式モデルや、論理計算の可能性と限界を明らかにした論理システム（離散時間モデル）との違いを示して、深い理解につなげる。

ビジネスプロセス工学の目標は、ビジネスプロセスの分析と設計である。分析は図的ツールとその背後の論理的数学的仕組みに基づいて行われる。設計は、仕事を進める上でのデータと仕組みの構造を決めることと、リトルの定理を基盤として、主に種々の在庫とリードタイムと全体的サービス速度の関係という動的特性を設計することになる。IT や情報システムはビジネスプロセスの構成要素や計画要素としての必須であるから、それらの分析や設計は、必然的に情報システムを扱う。

本書の歩き方

（1）ビジネスプロセスのモデル化能力を養成したい場合：ビジネスマンも学生も。

あなたは、現場で使われている売上げ伝票や勤務集計表等々の帳票から、データと業務を抜き出したり、ビジネスプロセスを描くことができますか。描いたものの正当性を論理的に説明できますか。まあまあの理解で満足したまま放っておいてませんか。たとえば、自分に届く電話の請求書や領収書、インターネットによる注文画面といったビジネスに関連する帳表データのモデルを作れ

ますか。顧客に関するデータを記述する時に、氏名や住所の他にもケータイの番号やメールアドレスを扱うのはいいとしても、なぜ顧客の血液型や帰省先や親兄弟・親戚・友達や飼い犬のことは記述しなくてよいのか、その理由を他人が納得できるように説明できますか。また、取引データをモデル化する際には、繰り返し項目に注目してから正規形に直していくような知識はほとんど使いません。どうやって、どんな原理によってそんなことができるのか、と不思議ですか？

第 2 章ではビジネスプロセスのもっとも基本的な仕組みとして離散事象システムを提示する。帳表データや世の中で発生しうるデータを整理する考え方がデータモデリング機能である。データモデルは第 3 章で示す。この章だけでもデータモデルの基本を理解し帳表をモデル化できるようになるはずである。データモデルを取り込んだビジネスプロセスのモデル化は業務取引システムと呼ばれ、第 4 章で説明する。第 5 章ではビジネスプロセスの伝統的な計画・評価・管理の仕組みをビジネスプロセス工学の観点から説明してモデルつくりの基礎知識とする。

（2）情報システム方法論の図的ツールをよく使えるようになりたい場合。

図的ツールを使ってデータやビジネスプロセスをモデル化できますか。図的ツールを使う際の「文法規則」が、物理学のモデルよりはるかに柔らかくていい加減だと感じたり、情報システムとビジネスプロセスには部品の寸法精度のようなきっちりしたものがなくて不安を感じてませんか。何に注目してツールで描いていくのか、また、自分が描いたものの正当性や妥当性に疑問を感じて、世の中にあふれる多くの図的ツールの中で溺れていませんか。

たくさんの図的ツールが提案されて、それぞれ便利に使われている。情報システムは真空中にそれ自身のためにあるのはなく、ビジネスプロセスをうまく動かすための仕掛けの一部として存在するのだから、いろいろなツールはビジネスプロセスをある程度は記述するはずである。では、何をどのように描いているのだろうか。図的ツール利用方法の社内標準規則はあるが、その理由について経験的でない理論的説明を自分は持っているだろうか。図的ツールの単なる「文法規則」を超えた意味を知ってモデル化する方がより適切であろう。本書ではアクティビティ・インタラクション・ダイアグラム（AID）を基本モデルとして使う。言い換えれば、世の中のすべてのビジネスプロセスの背後には数学的構造として業務取引システムの仕組みを持っていて、その構造を図的にあ

らわすと AID になるというわけである。第 6 章で、代表的な図的ツールがモデル化しているものを明らかにする方法を示す。証明する方法も示す。4 章で説明される動的な仕組みを動かすのに必要十分な情報を取り出しているかという基準によって、モデルの正当性が得られる。

（3）講義案

学部や大学院の講義には本書の内容を次のように利用できる。実際に、筑波大学で講義されて来たものである。

[A] 学部での経営情報システムやビジネスプロセス工学入門の講義科目

第 2 章離散事象システム（2.8 節は省略可能)、第 3 章データモデル（3.5 節は省略可能)、第 4 章業務取引システム（4.3 節の中の証明は省略可能)、第 5 章ビジネスプロセスのコントロール。第 6 章情報システム方法論の図的ツール。

各章で練習問題と練習問題が載っている本などを紹介したので、それらの例を実際にモデル化することを演習問題や実力養成利用できる。

[B] 経営工学系学部やビジネス系大学院でのビジネスプロセスのゲーミングとシミュレーションのモデル化を行う演習科目

第 2 章離散事象システム（2.1 節から 2.5 節まで)、第 4 章業務取引システム（4.1 節と 4.2 節と 4.4.節）を講義する。その際に第 5 章と第 6 章のトピックについて自習してレポートにまとめる課題とするなど、補足的に用いる。第 2 章の組立てゲームは第 2 章末に示したような、人間による組立プロセスのゲーミングシミュレーションを行うことができる。チームになった人間同士のコミュニケーションが進展するのも楽しい経験だし、さらにまた、プログラムによる分析への展開も可能である。

[C] 大学院でのビジネスプロセス工学または経営情報システム

第 2 章離散事象システム、第 3 章データモデル、第 4 章業務取引システム、第 5 章ビジネスプロセスのコントロール、第 6 章情報システム方法論、第 7 章動特性、第 8 章ペトリネットによる方程式入門。演習問題として他から追加することが好ましいのは、データモデル構築練習、ERP や MRP による生産計画作成やプロダクトデータの登録練習、生産計画システムや物流計画システムの実習である。ビジネスプロセスのシミュレーションモデル作成演習も追加できる。第９章の実現性理論は重要だが内容的にやや高度である。アドバンストなトピックとして、計算理論か数理的システム理論か制御理論を学んだ経験を持つ学生

にビジネスプロセスのモデルの概念基盤という位置付けで講義することで、離散事象システムとしてのビジネスプロセスという新しい工学的対象への深い理解につなげられる。

[D] 学部の研究やビジネスパーソンの能力向上課題

第２章、３章、４章、５章、６章から[B]の講義案のような節を選んでビジネスプロセスについて理解する。それをもとに、AID とデータモデル概念を実際にモデルとして用いて分析する。たとえば、現在の e-ビジネスや e-マーケットプレイス企業を取りあげて、その企業のユーザとなる顧客企業のビジネスプロセスを論理的に想像してモデルとしてまとめあげ、企業側が提供するサービスプロセスが顧客のプロセスのどこを代行できそうかを検討する。伝統的な企業の巨大ビジネスや、あるいは零細な商売について、どんなビジネスプロセスで仕事を行っていて、何を改善するかについてモデルを使って分析する。大学での理論と現実ビジネスプロセスを対応させることができる。

データモデルは人間の社会的活動の記述方法として基本的であるため、今後ますますビジネス的重要性が増す。たとえば、「XML は HTML にタグ情報を追加したものだ」という詳細レベルの見方よりもビジネス的に有用なのは、「XML は、データだけだった HTML にメタデータ情報を追加した」というビジネスプロセスレベルの意味である。データ転送の場面では、この追加によって、メッセージを通じた企業間でのビジネスプロセスの共同作業・同期化の道が開けたのである。メタデータが転送データの意味を表現できるからである。EDI の標準伝票として売られている帳表イメージを Web で見て XML スキーマで自分で定義するなど、その時々の IT 技術をデータモデルに置き換える練習課題が、データモデル化能力の向上と練成のための研究テーマとなる。

さらにすこし別方向の課題として、Web サービスや SOA（サービス指向アーキテクチャ）の開発環境ソフトの評価版や廉価版を使って、それらの技術内容と操作方法をキャッチアップする時の小プロジェクトの題材として、まず対象とするビジネスプロセスをモデル化を通じて分析し、IT をビジネスに位置づけことに用いる。６章末問題を参照されたい。

[E]大学院でのビジネスプロセス工学の研究課題

各章のトピックに関連した研究課題がある。第 6 章情報システム方法論の意味論を他のツールに対して広範囲に展開することで、理解と知識を充実させられる。第７章のビジネスプロセスの動特性の分析・設計方法の開発も必要とさ

れている。サプライチェーンのみならず SOA を情報技術基盤としたサービス連携体としての全体ビジネスプロセスや、あるいはまた、ユビキタス機器が充実した近未来での 1 個単位の管理方式のビジネス・ロジスティクスの計画システムに役立てる方向に充実させる必要がある。第 8 章業務取引ペトリネットを使った方程式表現の発展可能性は広大である。第 9 章で示した実現性理論を深化させて連続システム離散事象システムを融合したハイブリッドシステムを扱う。こうした重要で興味深い課題がたくさんある分野がビジネスプロセス工学である。

謝辞

本資料は pdf ファイルによる配付なので、感謝の気持ちは非常に大きいのですがごく簡単にお名前のみを挙げて謝意を表したいと思います。

高原康彦先生（東京工業大学名誉教授)。東京工業大学のシステム科学専攻の松田武彦・高原康彦・中野文平研究室の諸兄、専攻の当時の先生方。穂鷹良介先生（筑波大学名誉教授)。筑波大学の社会システム・マネジメント専攻および経営政策科学専攻の諸先生と研究室の学生諸君。オーストリア・リンツ市のヨハネス・ケプラー大学のピーフラー先生と研究室の方々。アリゾナ大学のジーグラー先生。経営情報学会の先生方。SAP 社の大学アライアンスの方々。

そして、妻と家族、母と亡父。

第１章　ビジネスプロセス工学の現状と展望

1.1 本書のねらい

ビジネスモデルやビジネスプロセスが注目され重要性を増している。情報技術が発展と成熟を繰り返すことで、ビジネスの重要なパーツとなると同時にまた、ビジネスを効果的に運転するための管理・運転装置になっている。高品質大量生産・大量消費の後に来たポスト工業化社会／情報化社会においては、ビジネスの世界ではモノとともに質の高いサービスをどういうふうに提供するかを競っている。それを可能にするビジネスの仕組みがビジネスプロセスであり、情報システムを組み込んだ形で作られている。情報システムの意味もはるかに広がって、コンピュータとネットワークを超えて顧客の購買履歴や原材料の出所管理や、あるいはまた、品目ではなく１品ごとの商品移動管理と在庫管理なども意味するようになった。いずれもビジネスプロセスの仕組みの作り方が重要な論点となってきている。

図 1-1 から図 1-5 はいくつかのビジネスプロセスの例示である。かなり直感的な理解が可能である。ここで注意すべきことは、いろいろな図的ツールがあること、人間の活動と機械や情報システムの組み合わせによってビジネスプロセスができていることである。

現代社会では、製造業や生活環境のビジネスプロセスが IT（情報技術）や新材料の発展や経済のグローバル化によってダイナミックに発展している。それに較べるとビジネスプロセスの工学発展が遅い。ビジネスプロセスのモデルはたくさんあるが、設計するという観点が希薄だという指摘もある。また、ビジネスプロセスとは何かという問いに対する答えがさまざまあるが、定まったものがないといってよい。

ビジネスプロセスはどこにでも存在する。仕入れから販売、請求の小売りのプロセスや、営業部のビジネスプロセスもある。生産管理方式を組み込んだ生産プロセスや、販売予測・在庫管理・予算統制・原価管理を実施している販売プロセスなど、ビジネスを実行したり、ビジネス遂行のための計画や管理のための仕組みが一体となっている。販売金額が零細なお弁当屋さんとか文房具屋さんなどでも、そのビジネスを実行するプロセスを持つ。また、巨大な石油プラントで原油からプラスチックの原料を大量に取り出し、多くの工程と卸売り

を経て、プラスチック製品や繊維製品やペットボトル等を製造するような広範囲に渡るビジネスプロセスもある。いくつかのプロセスがひとまとまりになってサブプロセスを構成し、それらのサブシステムが集って階層的にさらに大きなサブシステムを構成するという、巨大なビジネスプロセスとなっている。

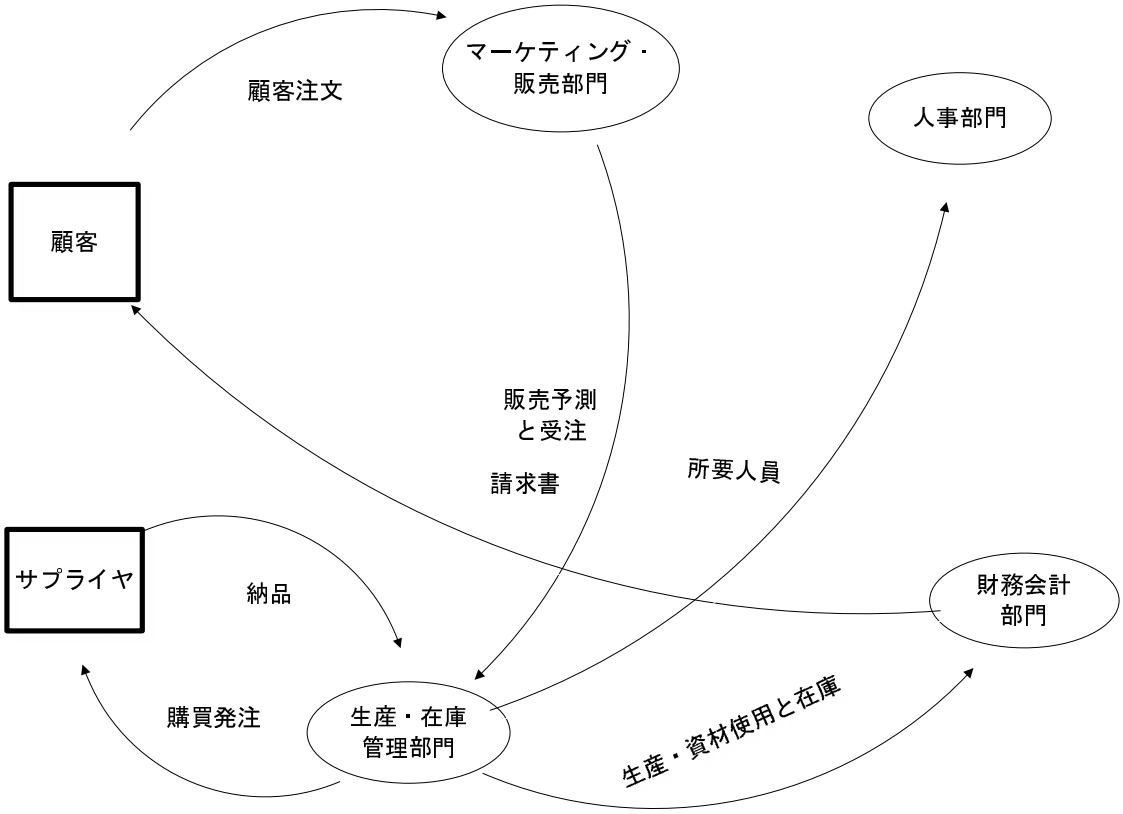
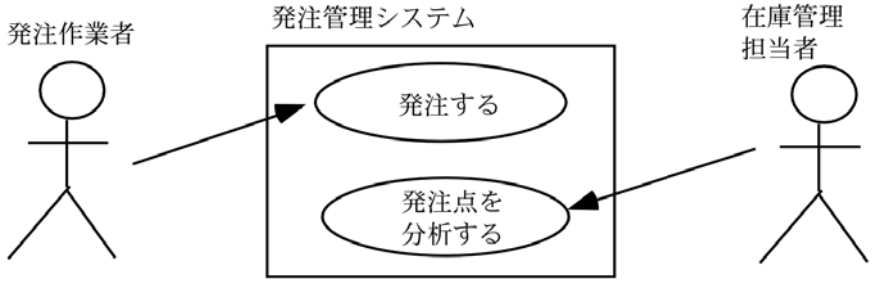


図１－１　販売と請求プロセスのデータフロー・ダイアグラム例



ユースケース名：発注する。
概要：発注点以下の品目について納入先へ発注する伝票を印刷する。
処理ステップ： (1) 在庫表で在庫が発注点以下になっているものを見つける。 (2) 発注する発注品目を決めて、発注を記録する。 (3) 納入業者ごとにまとめて発注品目を発注する発注票を印刷する。

図１－２　発注管理システムのユースケース図と活動のユースケース記述

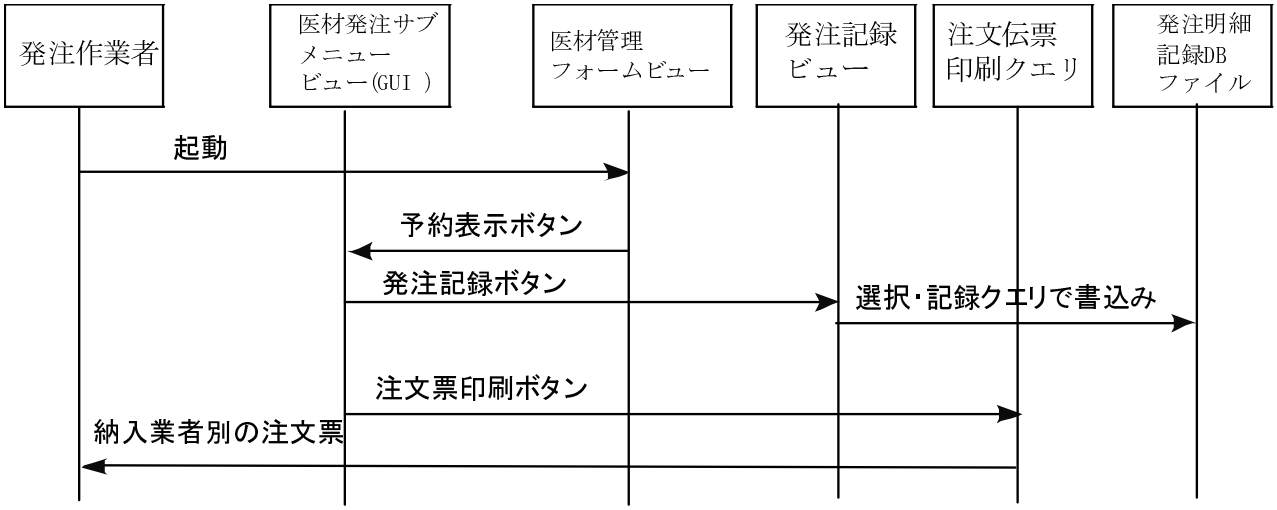


図１－３　「発注する」のシーケンス図

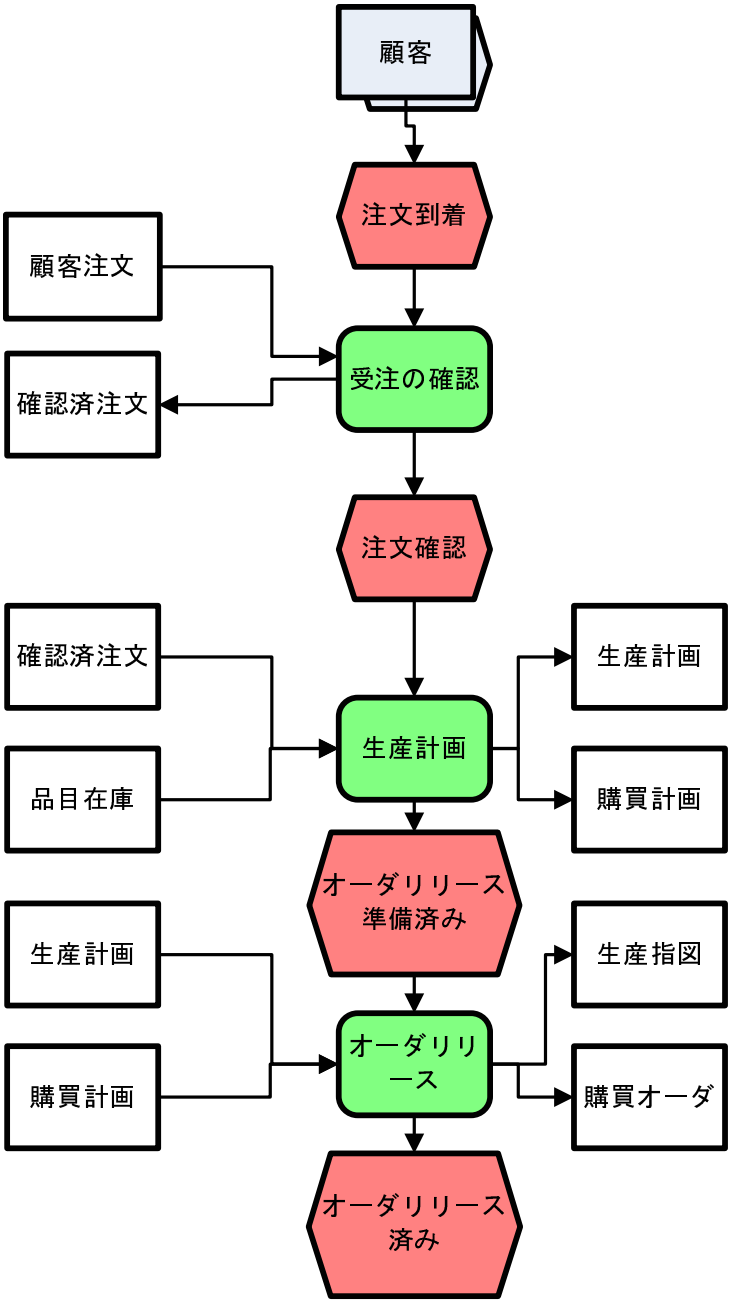


図 1－4　生産計画プロセスのイベント駆動チェイン図（event-driven process chain)

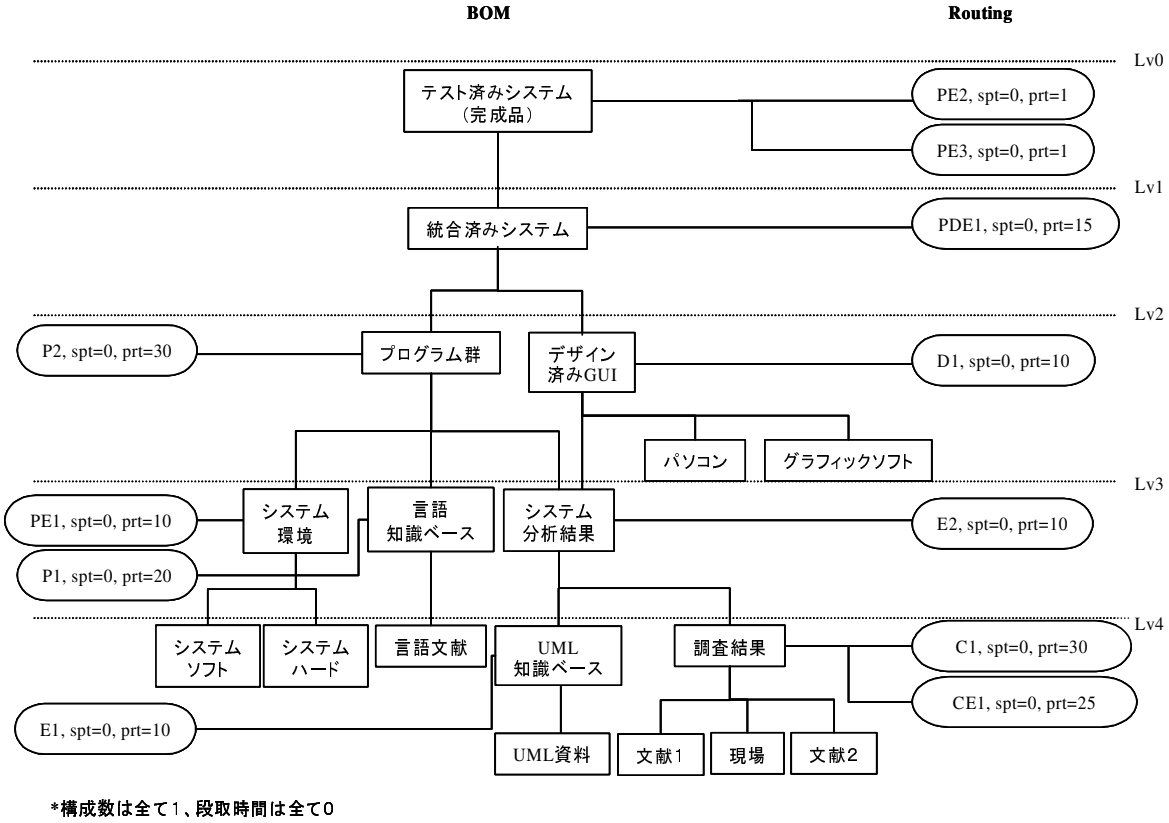


図１－５ソフトウエア試作プロジェクトの WBS 部品表型プロセス

本書では、業務取引システムというビジネスプロセスの工学的モデルを提示する。工学的モデルということの意味は、対象にとって本質的で客観的な論理構造を記述していること、それを使って分析と設計ができること、そのための方程式があって計算ができることである。たとえば、地球を回る人工衛星の軌道半径を目標値に変えるためには、物理学の万有引力や運動のモデルを使い微分方程式による計算ができるし、制御という概念と方法が使える。

ビジネスプロセスには図的モデル化ツールがたくさん提案されて実務に便利に使われているが、工学的モデルとしては未完成というべき状況である。図的ツールには歴史があり、出現順に、事務分析ツール、構造化分析設計ツール、オブジェクト指向ツールが代表的なビジネスプロセスのモデル化ツールである。ビジネスプロセスの重要部分であるデータモデルの図的ツールもある。これらは、既存ツールの欠点を克服するために提案されて来たというよりは、別の要因や流行的な要因から多様なものが混在するに至っている。たとえば、構造化

プログラミング・パラダイムの影響によって、構造化分析のツールであるデータフロー・ダイアグラムと関連する技法が出てきた。やがてパソコンのハードウエア向上から画面上にグラフィクスを扱いマウスで指示するようになり、GUI（グラフィカル・ユーザ・インターフェイス）が使い勝手を決めるようになり、オブジェクト指向プログラミングのパラダイムに影響されて、オブジェクト指向ツールが提案され、また、そのツールを扱う開発環境ソフトウエアが登場した。情報システムはビジネスプロセスの一部になったり管理の仕組みとして働くため、ビジネスプロセスをモデル化する図的ツールも同時に提案されている。本書では、業務取引システムというモデルを用いることで、ビジネスプロセスのさまざまな図的モデル化ツールの意味を確定する方法を示す。つまり、図的ツールの意味論の構成法である。

本書にはもうひとつの目的がある。ビジネスプロセスの動的な特性の工学的設計法を示すことである。それは同時に、情報システム方法論とオペレーションズ・リサーチ的な在庫管理の最適化モデルや技法との橋渡しをする方法となる。ただし、その設計法は緒についたばかりであり、発展すべき余地は広大だ。

ビジネスプロセスは原料から商品・生活用品へのモノの変換と蓄積を伴うので、巨大な在庫管理システムを構成している。販売や生産計画や在庫管理などの機能とともに構成されるモノの変換と蓄積の仕組みをロジスティクス・プロセスといわれる。伝統的在庫管理論は、ロジスティクス・プロセスの様々な変動要因を確率的挙動の中に押し込めてしまってはいるがリトルの定理という名前で知られる基本法則を生み出した。リトルは始めて一般的に証明した人である。物理の世界では、重い自動車を急加速させるには軽い自動車の場合よりも大きなエンジンのエネルギーが必要であり、ニュートンの力学法則として表現されている。外車だろうが国産車だろうが、あるいは乳母車でもおもちゃの自動車でもその原理から逃れることはない。加速しようとしたらそれなりの力が必要なのだ。しかも、ミクロの世界から宇宙の果てまで徹底して支配しているのである。リトルの定理も同じ厳格さでロジスティクス・プロセスにおいて成立する。e-ビジネスになろうが、ユビキタスな装置を使ったビジネスであろうが、あるいは大航海時代の植民地支配による貿易ビジネスだろうが、ビジネスを構成しているロジスティクス・プロセスはリトルの定理から逃れることはできない。顧客へのサービスや商品の提供可能速度と、プロセスの内の在庫と作業時間の関係について、逃れられない相互関係を表しているのである。

リトルの定理は基本的な動的特性の原理なのであるが、その原理を具体的にビジネスプロセス設計に適用する際に問題になるのは、サプライチェーンとよばれる広がりをもつロジスティクス・プロセスの在庫とか作業時間とは何のことか、何を表すのかということである。われわれは多品種少量生産の経済社会の中で生活し、また、情報社会や知識社会というように、モノを扱う具体的手段と同時に計画や在庫の情報が重要になってきている。特に情報システムを内部に持つようなプロセスでは動的な原理の適用が困難である。また、情報システム方法論やコンサルティングの手法でよく用いられる、ビジネスプロセスの図的モデルではどのように動的な性質を表しているのかわからない。つまり、ソフトウエアが満足に動く仕組みを記述はするが、それによって全社的なプロセス特性との関連はどうなるのか、ということが論理的に切れているのである。本書ではビジネスプロセスの工学モデルとして、業務取引システムの構造を使ってリトルの定理が表すような動的な性質分析方法を示そうとするものである。さらに代数的に計算する方法のヒントも用意する。

以上のように、本書はビジネスプロセスのモデルとして業務取引システムを提唱し、それを使って種々の図的ツールによるビジネスプロセスの構造設計と、在庫量やリードタイムといった動的特性の設計方法の工学的な基盤を与えようとするものである。

1.2 ビジネスプロセス・エンジニアリングの現状

かつて、いや今なお、ビジネスプロセス・リエンジニアリング - BPR - という言葉がある。業務構造を劇的に変更して、一度に業務の効果と効率を上げようというほどのことを意味し、数理論理的な内容よりも、サクセスストーリーや脚色された事例解説によって語られることが多い。工学としては、業務を「劇的に改善する」というような表現は、このままでは扱えない。もう一段の明確で操作的な記述・定義が必要である。

一方で、コンピュータを基盤とし仕事の組織的仕組みを構築するための情報システム方法論がある。情報システムは、抽象的な仕事の仕組みそのものを指すこともあれば、コンピュータと通信システムという具体的物体を意味することもある。いずれの場合でも、仕事の伝票や文書をコンピュータのデータベー

スやネットワークにのるかたちにする必要がある。それを可能にするためには、ビジネスプロセスを取り扱う必要があった。情報システム方法論では、情報システムのユーザになるであろう業務担当者から「どんなデータが表示されればよいか」を素朴に聞いて一覧表やチェックリストにして、それらを産出するためのデータを用意するというアプローチをとるものがあった。いや、いまでもかなりの方法論が、突き詰めればそれに類した論理でもって必要な情報を定義しようとしている可能性がある。オブジェクト指向の方法論も、グラフィックでインタラクティブなソフトウエアの実現方法としては相当の進歩をとげているが、それを制御装置とし組み込む「本体」であるところのビジネスプロセス自体のモデル化は、とても十分とは言いがたい。ビジネスプロセス・エンジニアリングという「工学」としては、寂しい。ビジネスのオペレーション上の望ましい特性を持つような管理系を設計するというような工学的な常識というか現状から見ると相当未発達である。この巨大な真空を埋めて設計方法を発展させて行くことが、広い意味の情報技術の発展のひとつの大きな方向なのである。

また、オペレーションズ・マネジメントという切り口は、伝統的な在庫管理論をベースにして、統計理論と待ち行列理論とシミュレーション最適化をツールとして提示する。ＳＣＭ（サプライチェーン・マネジメント）やＣＲＭ（カスタマーリレーションシップ・マネジメント）とも関係が深い。理論や手法も発展している。ただし、現状では、情報システムとの直接的関連が薄い。これを勉強した人が、いざ情報システムの利用法とビジネスプロセスの連携のためにその成果を使おうとするとき、自分自身で概念の統合から始めて、ビジネスプロセス全体の特性の設計を直感的に、あるいは発見的に行っていく必要がある状況である。たとえば、伝統的在庫管理論とその発展の結果を、MRP（資源所要量計画という生産計画・管理システム）や ERP（統合基幹情報システム）とは独立に説明している教科書しかまだ利用可能でないといってよい。

A.W. Scheer 氏の Business Process Engineering（Springer 社）ような、そのものずばりのビジネスプロセスエンジニアリングという非常に成功した本もある。Scheer 著は、CIM(computer integrated manufacturing）を発展させて、MRP II（ERP と同義的）という多品種少量生産の製造業を可能にする全社的な統合基幹情報システムを、図的レベルのビジネスプロセスモデル（EPC- event driven process cahin)図法と、ER データモデル（エンティティ・リレーションシップ）を用いて、情報のモデル化という側面から MRP II の構造のデータ設計図をほぼ完全に

描いている。これ自体相当に有用なものだが、やはり、たとえば製造時間と需要と各種在庫量の関係などの、動的側面の分析装置はない。

データモデルは現実世界のモノとコトの記述枠組なので、データをコンピュータを使って間違いなく扱うためには必須の概念である。そのため、特にデータベース技術とビジネスの境界において、さまざまなデータモデル概念が作られ発展した。データモデルという概念の意味は、技術発展とともに変化したが、現在ではビジネスプロセス工学の重要な要素となっている。

1.3 工学的展開としてのビジネスプロセス・エンジニアリング

現代社会ではひとつの確かな社会基盤となっているが、かつて化学工学という分野が興隆を極めていた。高度成長のころである。重油を原料としてさまざまな石油化学原料を効率よく製造するための「化学プラント」の設計や運転が重要な問題であった。石油化学プラント建設の苦労や喜び、化学工業の発展は、高杉良氏の小説「生命燃ゆ」などでいきいきと描かれている。それをひとつの応用分野として、制御工学という分野横断的な原理を扱う工学分野が成立した。プラントも電子回路も油圧装置も計測機械も、何でも微分方程式でモデル化できるものは、うまくやればフィードバック装置を組み込むことでその微分方程式が望ましい特性を持つように変えることができる。微分方程式が実際のモノの動作の安定性や動作速度を直接に表現しているから、現実の対象も新たに変更された微分方程式のように動くというわけである。同時に、微分方程式をより簡単に扱うための理論も発展した。ラプラス変換という対応によって、微分方程式が 2 次方程式や 3 次方程式になり、高校程度の計算で方程式を解き、今度はラプラス逆変換表で元の微分方程式の解を得られたのである。そうして、対象の特性をデザインする道具立てができあがっていった。

ニュートンの万有引力の法則はそれこそ存在する「万物」の間に働き、取り除くようなことはできない。しかし、太陽の周りを回る彗星や地球の軌道を微分方程式でモデル化するときも、実際には、軽すぎて影響がないような小さな宇宙空間内のチリなどは無視しても充分な精度が得られるし、また直径 1 万 3 千キロメートルもの大きさの地球を、その中心にすべての重さが集まっているかのように考えることで簡単化して、運動方程式を得ることができる。つまり、現実の対象の背後にある物的関係を微分方程式という数理論理的表現で表わす

場合の近似の問題がある。システムダイナミックスというアプローチでは、初期には、地球上の人間活動を、人口や環境汚染や食料供給などのただ５つの変数からなる連立微分方程式で表わし、たとえば人口増加率の変化が環境汚染や自然資源消費におよぼす影響などが調べられた。おそるべき単純化をしたわけだが、しかし結果として「宇宙船地球号（環境の有限性)」というコンセプトができたので注目を引き、人間が地球に及ぼす影響を注目させる上で大きな力があった。

現実対象の何に注目して、どういう論理関係を表わすモデルにするかは、プロフェッショナルとしての「職人芸」が要求されるところである。職人芸の共通的原理はといえば、試行錯誤ぐらいのものである。モデルをつくり、動かしたり当てはめたりして、モデルが失敗する側面を注意深くながめ、次のモデルつくりに活かす。そのためにはシステムモデルについての深い洞察が有用となる。

ビジネスプロセス・エンジニアリングにとって、販売のデータや計画のフィードバックとか、生産や在庫計画の最適化という考え方が有用に思われる。生産管理情報システムにせよ、サプライチェーン管理ソフトウエアにせよ、それらはビジネスプロセスの制御機構の一部として組み込まれ、ビジネスプロセス全体の特性を改善するのが目的だからである。ただ、ビジネスを制御工学とか物理学なのでよく使われる微分方程式で書いたとしても何に使えるのかは明らかではない。ストレートにいえば「使えない」。細かすぎてビジネスプロセスの適切な意味がとれないのである。伝統的在庫管理の統計モデルや待ち行列モデルについても、そのままでは、現在これだけ大規模に発展している、情報システムや e-ビジネスと結びついていない。たとえば、重要な側面であるデータの設計図との関連も明示的でないために、特性を実現するためのビジネスプロセス構造の設計に結びつかないのである。コンピュータとネットワークに結びついたビジネスプロセスのいろいろな特性を明らかにできたり、情報システムの設計に利用できるのが望ましい。そのような工学理論を発展させていく必要があるのだ。本書では、ビジネスプロセスの工学的モデルとして離散事象システムと呼ばれるシステムのクラスが適すると考えている。なお、いくつかの理論が使えないというのは、正確には当時のままでは使うのに困難であるというだけの意味である。先達がそれぞれの場面で考え抜いた理論や手法なのでモデルを少し抽象化や変形してビジネスプロセスに適用できるようにすることが、自

らの困難な問題の解決への効率的効果的方法であることも多い。

　ビジネスプロセスの工学は、工学領域としては巨大な真空であり、これからの非常に大きな発展が可能な分野なのである。

第 2 章離散事象システム

ビジネスプロセスが持つ基本構造は何だろうか。

本章ではそれを離散事象システムとして取り出す。2.1 では例として製造プロセスを取り上げる 2.2 以降では、たくさんの活動や部署から構成されるビジネスプロセスを離散事象システムとしてとらえるために概念を説明する。その上でさらに、離散事象システムの動作を精密にとらえるためにシステム理論の状態という概念を使う方法を説明する。これによって、これまでに考え出されたビジネスプロセスはもとより、将来発明され改善され続けるであろうビジネスプロセスに対しても適用できるような基本的な概念枠組みを得ることができる。

2.1　シンプルなビジネスプロセス

ドイツ高級車メーカのポルシェでは、POLE という名前で部品メーカ改善チームがサプライヤの管理職訓練のためにジャスト・イン・タイム納入訓練ゲームを使っていた[ムダなし企業への挑戦、ウォマック＆ジョーンズ（稲垣訳）、日経ＢＰ社、1997]。ポールポジションを自動車ビジネスというレースで獲得するねらいを込めた名前である。ゲームは人間どうしが行うもので、そのエッセンスは多品種少量生産のためのビジネスプロセスのコントロールを体験し学習することである。なお、章末でこのゲーミングをクリスマスカード作成ゲーミングとして実施する方法を説明する。

そのゲームのプロセスを抜き出すと、図 2-1 のような組立プロセスとなる。大小２つの部品を組み立てて製品を作り、それを検査した後で製品倉庫に入れる組立プロセスを考える。材料は予め材料倉庫に入れられており、注文に応じて組み立てていく。

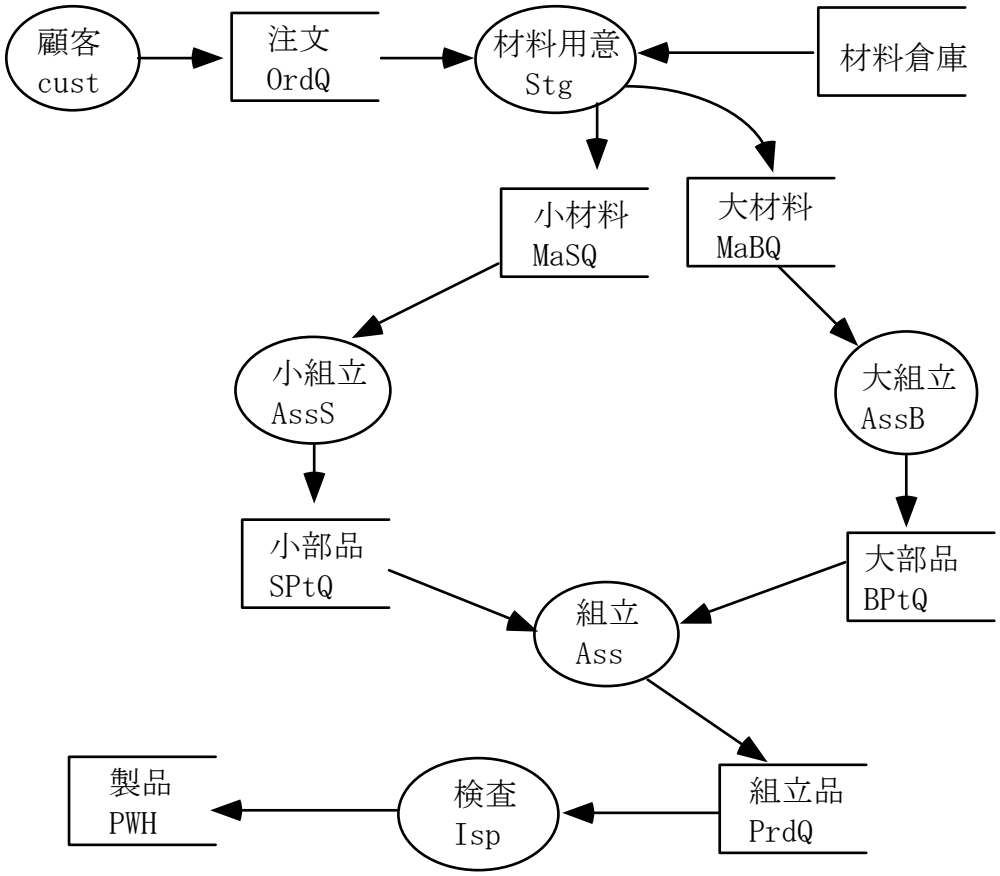


図 2-1　組立プロセス

2.2　各活動の作業者・開始条件・所要時間

各活動を実行するのは人か機械である。どちらでも同じモデルで扱うことができる。簡単のために、どの活動も作業者がひとりだけいるものとする。

１つの製品は、大小ひとつずつの部品から組み立てられる。各部品ひとつは各材料ひとつから作られる。注文は、1 つづつ注文するものとする。

各活動はある条件が満たされれば開始される。

材料用意　　受注が来ていて、材料が利用可能で、かつ作業者が他の作業をしてないとき開始される。5 分かかる。

小材料組立　小材料が利用可能で、かつ作業者が他の作業をしてないとき開始される。小材料から小部品を組み立てるのに 1 個につき 5 分かかる。

大材料組立　大材料が利用可能で、かつ作業者が他の作業をしてないとき開始され

る。1 個につき 6 分かかる。

組立　　大小材料が利用可能で、かつ作業者が他の作業をしてないとき開始される。1 個につき 6 分かかる。

検査　　製品が出来上がっていて、かつ作業者が他の作業をしてないとき開始される。1 個につき 2 分かかる。

顧客　　1 つづつ注文する。4 分おきにかならず誰かが 1 個注文する。言い換えると、4 分おきに注文という活動が開始される。普通はこんなことはあり得ないが、まずは、こんな例で考えてみる。たとえば注文個数がポアソン分布で表わされるようなランダムに来るときも、同じ構造でそのまま扱えることがシミュレーションについて後で説明される。

2.3　状態遷移表

ビジネスプロセス工学で問題となるのは、それぞれの活動に実行時間が設定されているときに、システム全体としての時間的な変化を追うための普遍的な方法である。ビジネスプロセスの評価は、動作が記述されてはじめて可能になる。ビジネスプロセスの性能を評価する側面はいろいろとあるが、この意味で、ここで説明する方法はどんな評価をするにせよ共通の基礎になる。

システム全体の動作を調べるために、すべての活動のようすと、すべての「データストア」つまり待ち行列のようすを調べていく。

始業時には、どの待ち行列もからっぽだとすると表 2-1 のようになる。

表 2-1 において、一番左の列が経過時間である。図 2- 1 の Stg とは材料用意(staging)の活動であり、図 2 の -stg- とは、その時刻において活動 Stg が何もしていないことを表わす。1(5) というのは、1 個の仕事を扱っており、あと 5 分で活動による処理が終わるということを表わしている。外部からの入力である(Cust)の列も同様だが、ordNo4(2)とは、4 番目の注文があと 2 分で発生すること。

time (Cust)	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
0 odNo 1(4)	0	-stg-	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
4 odNo 2(4)	1	-stg-	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
4 odNo 2(4)	0	1(5)	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
8 odNo 3(4)	1	1(1)	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
9 odNo 3(3)	1	-stg-	1	-SAm-	1	-BAm-	0	0	-Asm-	0	-Ins-	0
9 odNo 3(3)	0	1(5)	1	-SAm-	1	-BAm-	0	0	-Asm-	0	-Ins-	0
9 odNo 3(3)	0	1(5)	0	1(5)	1	-BAm-	0	0	-Asm-	0	-Ins-	0
9 odNo 3(3)	0	1(5)	0	1(5)	0	1(6)	0	0	-Asm-	0	-Ins-	0
12 odNo 4(4)	1	1(2)	0	1(2)	0	1(3)	0	0	-Asm-	0	-Ins-	0
14 odNo 4(2)	1	-stg-	1	1(0)	1	1(1)	0	0	-Asm-	0	-Ins-	0
14 odNo 4(2)	1	-stg-	1	-SAm-	1	1(1)	1	0	-Asm-	0	-Ins-	0
14 odNo 4(2)	0	1(5)	1	-SAm-	1	1(1)	1	0	-Asm-	0	-Ins-	0
14 odNo 4(2)	0	1(5)	0	1(5)	1	1(1)	1	0	-Asm-	0	-Ins-	0
15 odNo 4(1)	0	1(4)	0	1(4)	1	-BAm-	1	1	-Asm-	0	-Ins-	0
15 odNo 4(1)	0	1(4)	0	1(4)	0	1(6)	1	1	-Asm-	0	-Ins-	0
15 odNo 4(1)	0	1(4)	0	1(4)	0	1(6)	0	0	1(7)	0	-Ins-	0
16 odNo 5(4)	1	1(3)	0	1(3)	0	1(5)	0	0	1(6)	0	-Ins-	0
19 odNo 5(1)	1	-stg-	1	1(0)	1	1(2)	0	0	1(3)	0	-Ins-	0
19 odNo 5(1)	1	-stg-	1	-SAm-	1	1(2)	1	0	1(3)	0	-Ins-	0
19 odNo 5(1)	0	1(5)	1	-SAm-	1	1(2)	1	0	1(3)	0	-Ins-	0
19 odNo 5(1)	0	1(5)	0	1(5)	1	1(2)	1	0	1(3)	0	-Ins-	0
20 odNo 6(4)	1	1(4)	0	1(4)	1	1(1)	1	0	1(2)	0	-Ins-	0
21 odNo 6(3)	1	1(3)	0	1(3)	1	-BAm-	1	1	1(1)	0	-Ins-	0
21 odNo 6(3)	1	1(3)	0	1(3)	0	1(6)	1	1	1(1)	0	-Ins-	0
22 odNo 6(2)	1	1(2)	0	1(2)	0	1(5)	1	1	-Asm-	1	-Ins-	0
22 odNo 6(2)	1	1(2)	0	1(2)	0	1(5)	0	0	1(7)	1	-Ins-	0
22 odNo 6(2)	1	1(2)	0	1(2)	0	1(5)	0	0	1(7)	0	1(2)	0
24 odNo 7(4)	2	1(0)	0	1(0)	0	1(3)	0	0	1(5)	0	1(0)	0
24 odNo 7(4)	2	-stg-	1	1(0)	1	1(3)	0	0	1(5)	0	1(0)	0
24 odNo 7(4)	2	-stg-	1	-SAm-	1	1(3)	1	0	1(5)	0	1(0)	0
24 odNo 7(4)	2	-stg-	1	-SAm-	1	1(3)	1	0	1(5)	0	-Ins-	1
24 odNo 7(4)	1	1(5)	1	-SAm-	1	1(3)	1	0	1(5)	0	-Ins-	1
24 odNo 7(4)	1	1(5)	0	1(5)	1	1(3)	1	0	1(5)	0	-Ins-	1
27 odNo 7(1)	1	1(2)	0	1(2)	1	-BAm-	1	1	1(2)	0	-Ins-	1

表 2-1　組立プロセスのふるまい（状態遷移表）

2.4　離散事象システムの概念

ビジネスプロセスは、それがどれほどの広範囲にわたって巨大であっても、離散事象

システムとしてとらえることができる。それは，時間的に変化する入出力システムであって，システムの入力の変化が離散的であり，事象が並行的に起こり得るシステムである。時間軸は実数のように連続でもいいし、秒や週のような離散時間でもよい。ただし、実務的には、表 2-1 のような細かい動作情報の膨大なデータが利用可能だとしても、ビジネスに効果的に結び付けるためには、ビジネスプロセスのコントロールに関係した管理方式を作り込む必要がある。適切な特徴抽出の情報システムがデータマイニングや意思決定支援システムなどの形で利用可能であるとか、あるいは、オンライン的にフィードバックしてプロセスの計画管理情報として利用する必要がある。

なお、本書でシステムという場合は、ソフトウエアではなくハードウエアでもない。数などと同じ抽象的・観念的な存在である。リンゴでもパソコンでも 10 個あるというとき、10 という数が想定された世界があって、それをリンゴ 10 個に対応させている。システムは、いろんな実在するものが作り出す関係とか機能のことであり、入力と出力が可能にする関係を入出力システムという。

まずビジネスプロセスの背後構造として存在する離散事象システムについて、原理的なことを取り扱う。

離散事象システムの構成要素は２種類ある。システム内に存在するエンティティと，エンティティに影響を及ぼす活動である。エンティティはオブジェクトとも呼ばれる。

定義１　エンティティ（entity, object）

離散事象システムの中に存在するモノとコトで、活動によって発生したり、処理され変化するものである。エンティティやオブジェクトは、実体とも呼ばれることがある。

エンティティは待ち行列を作り、いくつかの待ち行列の状況によって活動が開始される。エンティティやオブジェクトは実体と訳されることが多いが、言葉それ自体の意味は問題ではなく、離散事象システムの他の構成要素である活動との関係によって上のように定める。

エンティティやオブジェクトの例には、客、在庫引き当て待ちのの注文、与信調査終了を待つ顧客注文、組立ショップに指示された製造オーダ、乗りもの、人、病原菌、車、メッセージ、など多様なものがある。ある処理を待って蓄積することがありうるものである。

定義２　エンティティのタイプ（entity type, オブジェクトが属するクラス）

エンティティの種類を表わす名前。つまり、エンティティ を類似性に応じてまとめたものの名前である。エンティティ・タイプを管理実体型とも呼ぶ。

離散事象システム全体はエンティティの流れ道であると考えることができる。エンティティが蓄積する場所は、たとえば、コンビニのレジ，工場のペイントショップのスプレーブース，通信ネットワークにおける通信装置、受注処理係が受け付ける引き合い、あるいはWebサービス上でサプライヤとメーカのコラボレーションとして実現された自動化受発注相互プロセスの中で在庫引き当てを待っている注文のファイル、などである。

ビジネスプロセスで取り扱われるのは個々のエンティティであるが、モデルで考えるのは個々のエンティティではなくエンティティのタイプである。実は「客、注文、処理待ちの製造オーダ、乗りもの、人、病原菌、車、メッセージ」などはエンティティのタイプとして使われていることが多い。販売管理業務を考えるうえでは、顧客エンティティ・タイプの要素である「佐藤さん」がひとつのエンティティである。また、注文エンティティ・タイプの要素である「佐藤さんからの注文」もひとつのエンティティである。

定義３　活動（activity）

エンティティに対して加工やサービスという処理を行ない、大なり小なりエンティティを変換することを活動と呼ぶ。オブジェクトを変換することとも言える。活動の開始から終了までには時間がかかる。部署もひとつの活動としてとらえることができる。活動は、業務活動、業務、または、トランザクションと呼ばれることもある。

受注後の発送業務では，多くの商品の利用可能性に応じて作業を行うが，この場合，注文というエンティティの処理順位はいろいろな場合があって複雑となる。

定義４　待ち行列変数（queue）

活動による処理を待つエンティティが蓄積される変数を待ち行列変数と呼ぶ。単に待ち行列と呼ぶこともある。待ち行列の中には、エンティティ、もしくはオブジェクトがその後の処理を待って蓄積される。同一エンティティ・タイプに属するエンティティとひとつの待ち行列変数は同一視できるので、待ち行列変数の名前をエンティティ・タイプで表現することが行われる。

待ち行列の状況に応じて、活動の実行が開始される。銀行窓口で頼んだ仕事も、カウンター内のそれぞれの係員の前で、書類に記載されて待ち行列を作り、ひとつの仕事が終わると次の仕事が開始される。待ち行列は、具体的にレジやカウンターの前に人間が並んでいる場合もあるし、注文が記載された書類が積み上げられている場合もあるし、工場内に部品が在庫として散在していることもあるし、情報システムのデータベースのファイルの中に処理を待つデータとして蓄積されていることもある。

定義５　イベント（event，変化時刻）

システムの中で，どれかの活動を開始する時刻や終了する時刻のことである。

イベントの例として、レジにおける精算の開始や、ペイントショップでのオペレーションのスタートや作業終了がある。

2.5　アクティビティ・インタラクション・ダイアグラム（AID）

2.5.1　AID の構成要素

離散事象システムのモデル化はまず，システムの中を流れるエンティティに対する種々の作業のネットワークの「絵」を描くことからはじまる。その絵はアクティビティ・インタラクション・ダイアグラムと呼ばれる。システムに含まれるエンティティと活動の相互作用を表現している。離散事象システムの「骨格」を規定するともいえる。相互作用は通常は目に見えない。たとえば、工場の中にある部品が次にどこに行くのかは、部品そのものの物理的存在から決まるのではなく、生産プロセスの加工経路の定義とスケジュールによって定めてあり、その指示内容によって決まる。

図 2-1 や次の図 2-2 はアクティビティ・インタラクション・ダイアグラムの例である。

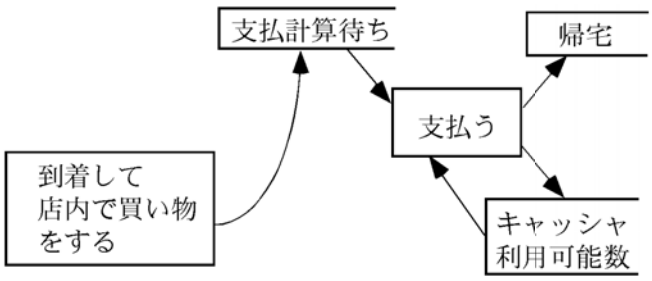


図 2-2　買い物離散事象システム

離散事象システムとしてビジネスプロセスを認識するために必要なものは，(1)エンティティ・タイプ，(2)それらと関係する活動，(3)エンティティが作る待ち行列の制御を含む活動と待ち行列の間の関係，である。

activity interaction diagram で用いる図形要素を３つ定義する。

定義６　活動図形

活動(activity)を表現する。外生的でシステムが制御できない外生活動と、システム内の活動である内生活動の２種類がある。内生活動には，入力の矢印があるが，外生活動では入力矢印がない。

図 2-2 の買い物離散事象システムでは，活動は３つあり長方形で表わされて，名前が付けられている。「到着して店内で買い物をする」が外生活動であり，「支払う」が内生活動である。いわゆる事務処理をイメージするような場合に、活動という代わりに、トランザクションといわれることが多い。

活動による作業にかかる時間がある分布に従っている場合は，その分布からのサンプリングによって必要時間を表現する。窓口でのサービスにかかる時間，自動車の工程の塗装にかかる時間などが，活動の実行終了に必要な時間の例である。

定義７　待ち行列図形（処理待ち行列）

待ち行列図形は，あるエンティティが活動による処理実行を待っている待ち行列をあらわす。

買い物離散事象システムでは，入り口で待っている客は「買い物をする」活動を待っていると見なせる。右開き長方形（open rectangle）で表現されている。また，自動車ボディーの塗装作業のプロセスを考えると，「塗装仕上げ作業待ち」のエンティティが待ち行列をつくる。いつ塗装仕上げのための装置が空くかに依存して決まる。

定義８　アクティビティ・インタラクション・ダイアグラム（AID）

３つの構成要素からなる図である。活動図形と 待ち行列図形とそれらをつなぐ矢印である。ただし，どのように矢印をたどっても，活動と 待ち行列が交互にあらわれる構成となっていることが必要である。

交互に活動と処理待ち行列が現れるということは，エンティティの側から見ると，活動の前後にあるエンティティの様子を変化させるのが活動だということである。ひとつの活動が実行できるのは，その活動への入力エンティティの待ち行列がある条件を満たすときである。つまり，活動開始条件は活動への入力の矢印でつながれたいくつかの待ち行列に対する条件で表現される。

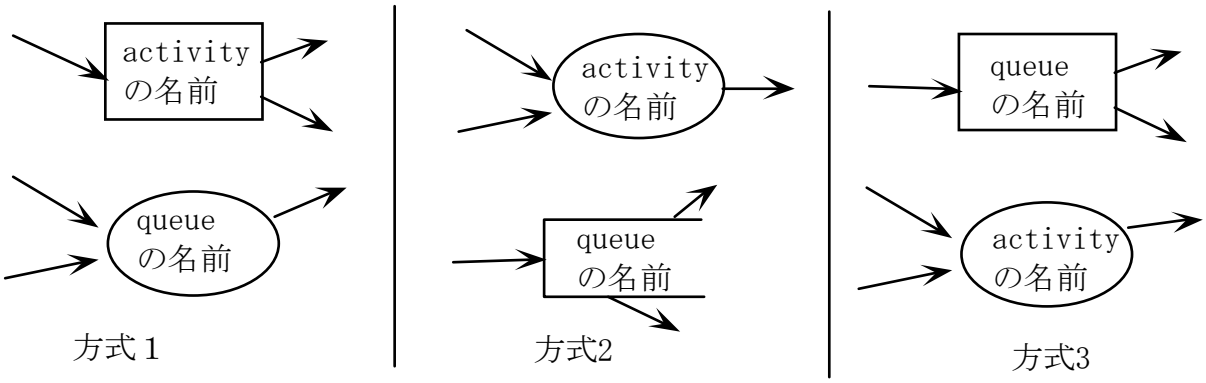


図 2-3　アクティビティと待ち行列（３とおりの例）

AID は，システムの中に待ち行列があるような，エンティティ間の相互作用を離散事象システムとしてモデル化するひとつの方法である。すなわち，AID はエンティティの各クラスが作る待ち行列と活動の関係（相互作用）をグラフで表示する。

例３　販売・請求業務

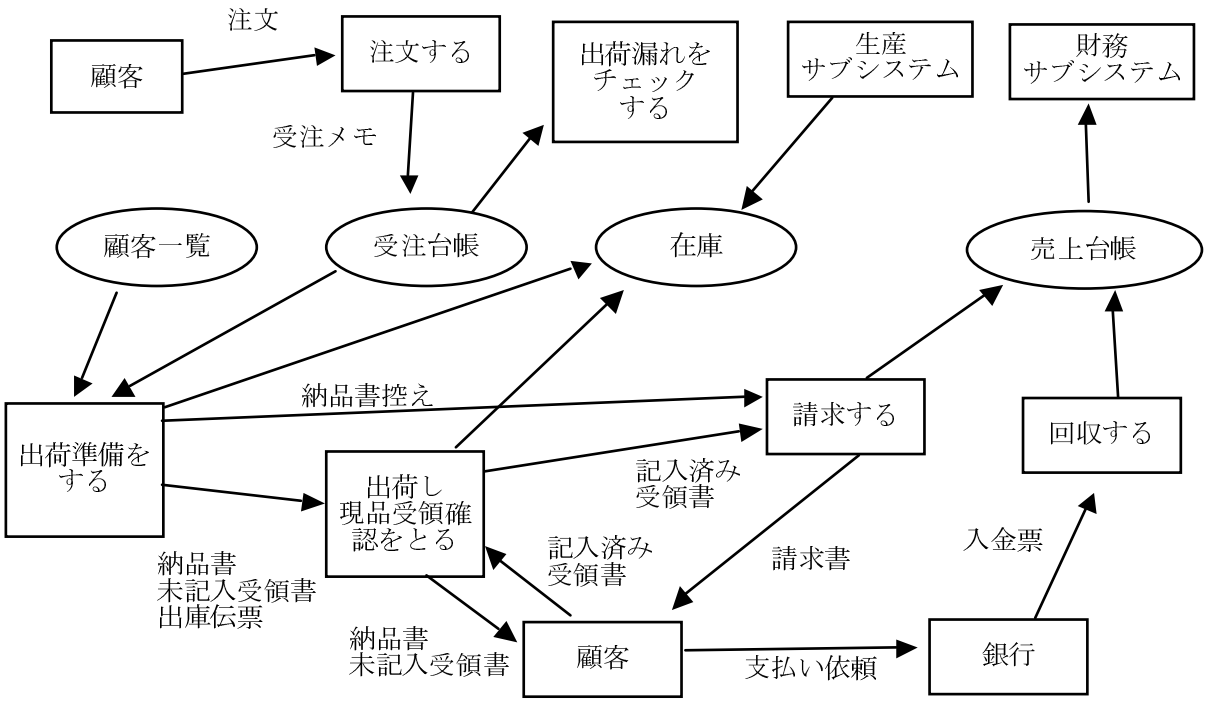


図 2-4　販売業務

問 2-1

図 2-4 販売業務は、このままでは AID としての条件を満たさない。矢印はデータの流れを表しているので、矢印に示された名前を待ち行列図形を経由するような入力と出力の矢印に直すことで AID にしなさい。

例４　組立ラインのカンバン・システム

２つの組立工程からなる組立プロセスを AID として表現したものである。カンバンを使ってコントロールされている。

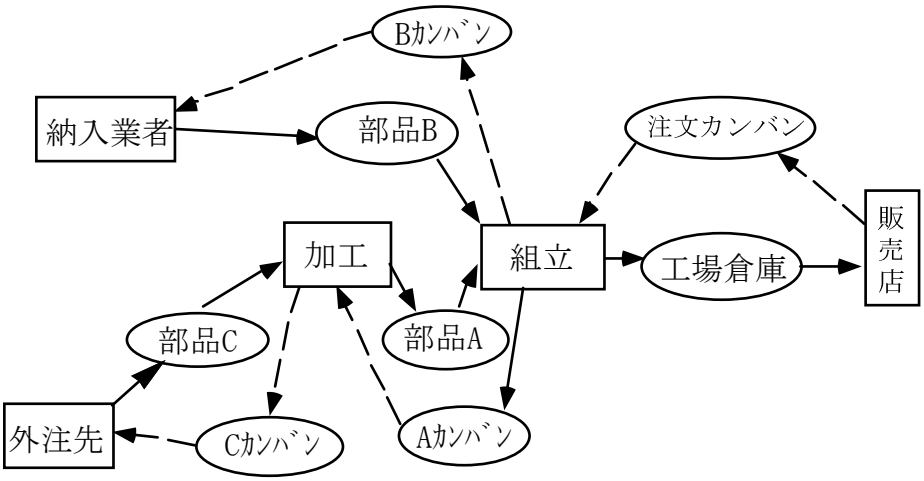


図 2-5　カンバン方式による組立プロセスの管理

カンバンは部品箱とともに工程間を移動するので、2 工程をつなぐ待ち行列変数である。たとえば製品を組み立てる組立工程では、作業開始時に部品 A の部品箱からカンバンがはずされ在庫する。カンバン置き場は後工程から前工程へ流れるカンバン・エンティティの待ち行列となる。あらかじめ決められた枚数のカンバンがカンバン置き場にたまると，前工程である加工工程へ生産指示として空の部品箱とともに運ばれる。運ばれたカンバンが生産指示となってカンバン枚数に応じた量だけのものを加工する。同様に部品Ｃのカンバンによって生産指示が前工程である外注先に伝えられる。空箱とカンバンの流れは点線で示されている。

こうして、後工程で部品が消費されない限りカンバンははずされず、必要のない生産は行われない。引き続く 2 工程の間には 1 定枚数のカンバンが使われているので、一定数量以上の在庫が生ずることはない。

カンバンシステムについての詳しい解説は門田安弘『新トヨタシステム』(講談社, 1991)や、同『トヨタシステム』(講談社文庫, 1989）にある。

例５　医療機関による診断・治療業務

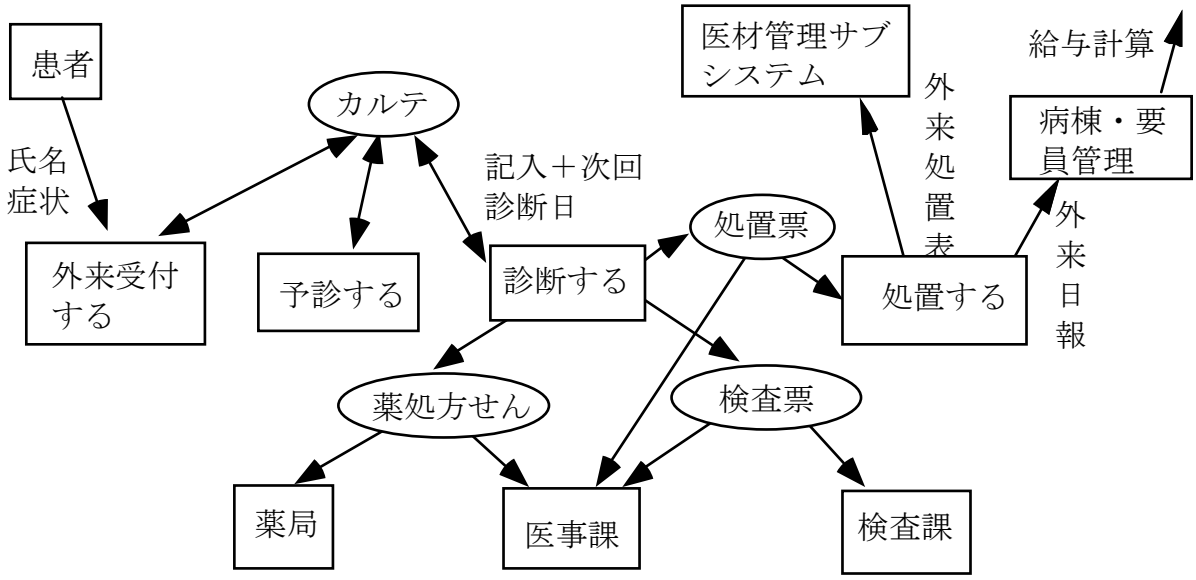


図 2-6　外来診断離散事象システム

2.5.2　動的なふるまい

アクティビティ・インタラクション・ダイアグラムは離散事象システムの構造を決めているが，そこで扱われるモノや要求は，時間的にどういう変化をたどるか。結論をいえば、表 2-1 のようにすればよい。このことを一般の AID で表わされるシステムに対して説明する。

アクティビティ・インタラクション・ダイアグラムが記述するものは，(1) 業務処理活動（ビジネスプロセス構成要素）の名前，(2)待ち行列の構造 ，(3) 待ち行列と活動の結合である。それらは時間的変化を記述しないという意味で，静的結合構造である。

組立プロセスの例では，注文がその後の処理を待って滞留する。つまり待ち行列を作る。時間の経過とともに、さらに注文が来たり、あるいは現在実行中の組立が終了したりする。

離散事象システムの時間変化

AID によってモデル化される離散事象システムは、図 2-7 のフローチャートのような時間的なふるまいをする。このふるまいを状態遷移と呼ぶ。まず、言葉を用意する。

活動の実行としては、人間によるものばかりでなく、コンピュータ化された情報システムや機械による活動の実行も含めて考えるので、活動は実際に分析が行われる場面では、トランザクションとか業務とも呼ばれ、活動による処理のことをトランザクション処理とか業務処理とも呼ばれることがある。

図 2-7 の外生トランザクションとは，システムの外で行われる活動のことであり、組立プロセスの例では、顧客（による「注文する」）という活動である。AID においては出力のデータフローだけしか持たない活動で表現される。また，業務とは，加工のような入力データフローを持つ活動である。活動には、外生トランザクションかあるいは業務の 2 種類しかない。 図では待ち行列を、より一般的に、ファイルと呼んでいる。

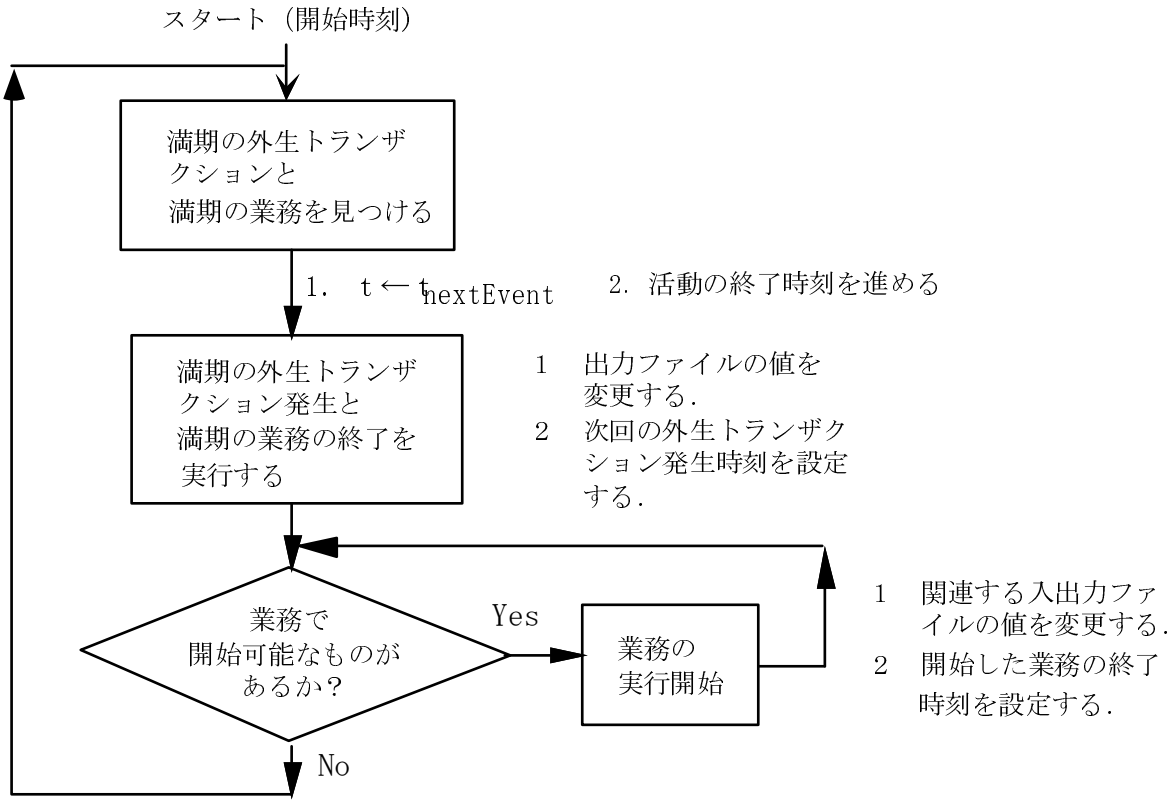


図 2-7　状態遷移（動的ふるまい）

また，「業務で開始可能なものがあるか」という条件判定がその業務の入力ファイルだけから判定できるように，入力を完全にそろえて保持されている必要がある。実際、たとえば、部品が揃わなければ組立を始めることは不可能だ。また、銀行の ATM で現金を引き出すのには、銀行カードに記録されている情報と暗証番号がなければ、銀行とし

てのその先の業務は進まないので、客はお金を引き出せない。

AID で表現される離散事象システムは、次の特徴を持つ

(1) 定型業務システムへの入力は、連続した変化ではなく、イベントがぽつりぽつりと起こるような変化である（図 2-8)。イベントとは、活動の開始か、終了のことである。

(2) 帳票とかデータストアに蓄えられたデータやモノがそれらの活動を結合している,

(3)プロセス内の活動の開始や終了が同時並行的に処理される。

(4) 活動が開始するための条件が、その活動への入力 queue の組の値だけから定まる。

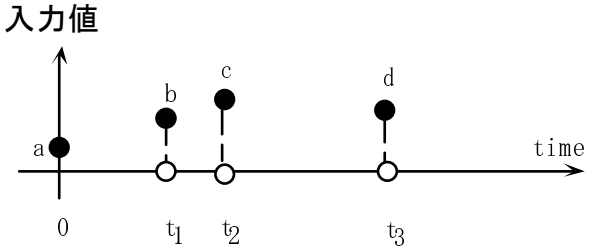


図 2-8　定型業務システムの入力空間

図 2-8 において、たとえば、時刻 0 におけるイベントである *a* とは品目 1 を 50 個という注文であり、*b*とは品目 2 を 3 個と品目 1 を 20 個という注文である。条件(4)は、AID によるモデルの条件である。これは、モデルを作成するとき、どういうデータやエンティティを各活動の入力とすべきかの判定についてのガイドラインともなる。

$t_{nextEvent}$とは、最近の将来に起こるイベント時刻のことである。

離散事象性の離散というのは、業務は「開始」したか,「終了」したかのイベントしか起こりえず，その業務の状況としては「処理中」か,「処理していない」かのどちらかしかないことである。もし連続的だと、42.9823%だけ処理したなどということになるが、離散事象システムでは「開始」の有無だけに注目するから、そういうことはあり得ない。また、電圧が連続的に変化しているような連続的な値を持つ入力も受け付けない。しかも業務処理は、同時並行的になされる。つまり、人間組織一般に見られるように、それぞれの仕事を各自がこなしていく。

図２―１の組立プロセスの離散事象システムが、図 2-7 のトランザクション処理フローチャートにしたがって，実際にイベント発生の時点だけでいくつかの活動が，時には並行して起こるようすを調べよう。

全体のふるまいを記述する情報は，組立プロセスの場合は下の(i), (ii), (iii) である。(i)は状態遷移図の中の 1 つの行であり、(ii)は各表示の意味を一般的に書き直したものである。また活動が行われていないようすを、(iii)の Ass（組立(assembly)の活動）や Isp（検査(inspection)の活動）ように Λ で表現することもある。ヌルと読むがギリシャ文字のラムダの大文字である。

(i)

time	(Cust)	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
12	odNo4(4)	1	1(2)	0	1(2)	0	1(3)	0	0	-Asm-	0	-Ins-	0

(ii)

time	(Cust)	OrdQ	Stg	MaSQ	AssS	…
時刻	オーダNo4到着(次の必要到着までの時間)	待ち個数	1個処理中(終了までの必要時間)	待ち個数	1個処理中(終了までの必要時間)	...

(iii)

time	(Cust)	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
12	odNo4(4)	1	1(2)	0	1(2)	0	1(3)	0	0	Λ	0	Λ	0

業務活動の処理終了までの残り時間を、材料用意の活動である Stg の場合を例として説明する。$ta_{Stg}(order3)$は、order3 のための材料用意の活動開始から終了までの所要時間（活動が満期となる時間ともいえる）を表すとする。order3 を処理中の任意の時刻において、活動 Stg の処理終了までの残り時間は

$ta_{Stg}(order3)$ - (開始からの経過時間)

で計算される。例ではどんな注文に対しても一定時間 5 分かかるので$ta_{Stg}(order3)$の値は５である。組立プロセスの例ではどの注文も同じ内容なので区別していない。実際には注文ごとに内容が異なるので処理時間も異なるが、同様に扱うことができる。

注文の到着も業務と同じように扱うことができる。次の注文である order4 を顧客が処理中であって、$ta_{Cust}(order4)=4$ として、業務活動と全く同じに扱える。

これらの情報をひとまとめとして、(離散事象システムの) 時間的変化を記述する仕組みが次の節の DEVS 記述である。離散事象システムでは上のような待ち行列ベクトルを状態とするので、おおまかなイメージでいえば、オートマトンの状態遷移関数にある待ち行列状態になってからの時間経過関数を付け加えたものが、離散事象システムの場合の状態遷移関数になる。

買い物離散事象システムの場合も調べてみる。

10 分に１人の客が到着し, 15 分かけて買い物し, 支払うのに３分間かかるものとする。

キャッシャー担当者は 2 人とする。
このとき，0 分に 1 人目の客が到着したとすると，図 2-7 のフローチャートにしたがって，図 2-2 に対応するデータは，表 2-2 のように変化する状態遷移表ととなる。

時刻	到着する	入り口での待ち	店内で買いものをする	支払い計算待ち	支払う	キャッシャ利用可能数	帰宅
0	(C2, 10)	C1	---	---	--	2	0
0	(C2, 10)	0	(C1, 15)	---	---	2	0
10	(C3, 10)	C2	(C1, 5)	---	---	2	0
10	(C3, 10)	0	(C2, 15) (C1, 5)	---	---	2	0
15	(C3, 5)	0	(C2, 10)	C1	---	2	0
15	(C3, 5)	0	(C2, 10)	---	(C1, 3)	1	0
18	(C3, 2)	0	(C2, 7)	---	---	2	C1
20	(C4, 10)	C3	C2, 5)	---	---	2	C1
20	(C4, 10)	0	(C3, 15) (C2, 5)	---	---	2	C1
25	(C4, 5)	0	(C3, 10)	C2	---	2	C1
25	(C4, 5)	0	(C3, 10)	---	(C2, 3)	1	C1
28	(C4, 2)	0	(C3, 7)	---	---	2	C1, C2
30							

(以下、永遠に変化する)

表 2-2　買い物離散事象システムの状態遷移表

問 2-2
状態遷移表 2-2 において、時刻 30 の行の内容を書きなさい。

2.6　DEVS 仕様

離散事象システム(discrete event system specification: DEVS)を以下のように定義する。定義の具体例は、定義の次にある。DEVS 理論はアリゾナ大学の B. P. Zeigler によって発展してきた。乱暴にいえば、離散変化する連続時間上のオートマトンを考えるために、満期時間関数 ta を付け加えたものといえる。

定義 9 :　Discrete event system specification :DEVS 仕様、離散事象システム仕様
DEVS 仕様Mは次の 6 つ組で与えられる :
$M =< A_M, S_M, B_M, \delta_M, \lambda_M, ta >,$

A_M　入力値の集合
S_M　疑似状態の集合（なぜ「疑似」と呼ぶかは後で説明する）
B_M　出力値の集合
δ_M　疑似状態遷移関数（なぜ「疑似」と呼ぶかは後で説明する）
λ_M　出力関数
ta　満期時間関数（the time advance function 時間進め関数とも呼ぶ。）

それらは以下を満たすものとする。

(1) $ta : S_M \to T^\infty$; ただし、T は時間軸集合（通常は実数）であり $T^\infty = T \cup \{\infty\}$ である。

(2) $\delta_M : Q_M \times (A_M \cup \{\Lambda\}) \to S_M$

$$\delta_M(s,e,\Lambda) = \delta_\Lambda(s) \quad \text{for all } (s,e) \in Q_M$$

ただし $Q_M = \{(s,e) \mid s \in S_M, 0 \le e \le ta(s)\}$, □ は空事象を表わす。（$A_M$ は空事象を含まない。）また $\delta_\Lambda : S_M \to S_M$ である。ここで e は疑似状態が s になってからの経過時間 (elapsed time) を表わす。関数 δ_Λ を（外部入力に依らない）内部状態遷移関数と呼ぶ。

(3) $\lambda_M : Q_M \to B_M$

δ_M が一般的に成立する有限時間消費条件（内部処理の必要時間がゼロでないという条件）のもとでは δ_M は状態遷移関数であり、したがって Q_M は状態空間であることを後で示す。

2.7　DEVS 仕様の例

図 2-1 の組立プロセスでは，離散事象システム仕様の定義の□□や□M などがどのような関数かを示す。

入力値は「(顧客の) 注文」だけである。つまり A_M ＝｛注文｝。もし「引合い」という入力も可能な場合には、A_M は２つの要素を持つ集合｛注文，引合い｝になる。

出力値を何にするかは分析の目的による。たとえば、各待ち行列の長さや、マシンが加工しないでいる時間とか、あるいは加工中の時間であったりする。

S_M はシステムの時間変化を記述するのに必要な情報である。まず、図 2-1 の AID に含まれる活動と待ち行列について、表 2-1 のようにそれらの名前を属性に持つようなベクトルを作り(ii')のようにする。ただし、外部入力である Cust の部分を取り去る。

(ii')

time	OrdQ	Stg	MaSQ	AssS	…
時刻	待ち個数	1個処理中(作業開始時刻)	待ち個数	1個処理中(作業開始時刻)	...

これに対応するデータは、

(I')

time	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
12	1	(1,9)	0	(1,9)	0	(1,9)	0	0	-Asm-	0	-Ins-	0

となり、次を意味する。

(i'のもつ情報)　活動 AssS の値が(1,9)ということは、

（ta_{AssS}(1)＋現在の材料 1 個の処理開始時刻＝５＋９＝）14 に処理終了。

なお、 -Asm-は作業を現在行っていないことを示す。こういう(i')の形式のデータの並び（ベクトル）がひとつの疑似状態 s であり、それらの疑似状態をすべて集めたもの（集合）が疑似状態集合 S_M である。

システム全体の満期時間関数 $ta(s)$は疑似状態 s に含まれる各活動のうちで最近の終了時刻までの時間となる。つまり、各活動の生起時刻と満期時間の和のうちで最小の時刻までの残された時間となる。形式的に次のように定める。

活動 k が現在の作業を開始した時刻を t_k とする。作業を一度も開始していない活動 k については $t_k=-1$とする。

$$t'=\max\{t_{Stg},t_{AssS},t_{AssB},t_{Ass},t_{Isp}\}$$

とする。作業を行っていない活動 k については $ta_k(\Lambda)=\infty$ と定める。たとえば、$ta_{Ass}(\Lambda)=\infty$ である。このとき

$$\min\{\ ta_{Stg}(j_{k1})+t_{Stg},\ ta_{AssS}(j_{k2})+t_{AssS},\ ta_{AssB}(j_{k3})+t_{AssB},\ ta_{Ass}(j_{k4})+t_{Ass},\ ta_{Isp}(j_{k5})+t_{Isp}\ \}$$

という最小値をもつ活動 a について

$$ta(s)=ta_a(j)+t_a-t'$$

と定める。したがって、疑似状態 s に対して $ta(s)$は最近に発生する内部イベントまでの残り時間であるとも言い換えられる。

内部状態遷移関数 δ_Λ は，内部イベントが起こったときの現ジョブの終了と次ジョブの開始によるファイルの変化を記述する。ある活動 k がオブジェクト m について作業中であり、 $ta(s)=ta_k(m)-t'$のとき：

$\delta_M(s,e,\Lambda)=\delta_\Lambda(s)=$ 次のようにする：

[（図 2.7 に示したように）活動 k の出力ファイルへ現ジョブ m を加え，活動 k での処理を待つものが k の入力にあればあれば k の入力ファイルからひとつ取り出して、処理作業

を開始する。満期になる活動がひとつの時刻で複数あれば、それぞれについてこの操作を行う。満期にならない他の活動については不変である。]

この新たな疑似状態 $\delta_\Lambda(s)$ の経過時間は０である。

$\delta_M(s,e,order_i)$=[注文iが入って来た瞬間にs の中の$OrdQ$の値を１つ増加する。それ以外の待ち行列と活動の状況は不変。経過時間を０とする。]

このように、状態遷移表はイベントがあったときのことだけを記述するため、状態遷移表の各行に示される疑似状態について、経過時間は０である。

出力は観察したい変数を自由に決めればよい。たとえば、すべての擬似状態を逐次出力すれば，動的ふるまいについて最も詳しい情報が得られる。

ここまでに示した関数や状態は、組立プロセスの例では次のようになる：

時刻 12 において注文が到着した時点で

s = [1　(1,9)　0　(1,9)　0　(1,9)　0　0　-Asm-　0　-Ins-　0]

であり、sになってからの経過時間 e は０である。このとき s の満期時間は、$ta(s)$ =14-12 = 2 である。なぜなら $ta_{Stg}(1)+9=14$，$ta_{AssS}(1)+9=14$，$ta_{AssB}(1)+9=15$であり、t'= max{12, 9, 9, 9, -1, -1} = 12 より $ta(s)=2$となる。ただし、各活動が何番目のオブジェクトを作業しているかは無視しており、すべての待ち行列が FIFO で（先着順で）扱われているものとしている。

$\delta_\Lambda(s)$ = δ_Λ([　1　(1,9)　0　(1,9)　0　(1,9)　0　0　-Asm-　0　-Ins- 0])

= [　1　(1,14)　0　(1,14)　1　(1,9)　1　0　-Asm-　0　-Ins-　0]

を得る。

問 2-3

上記の s について、 $ta(\delta_\Lambda(s))=1$となることを示せ。

上で示した疑似状態 s の表示方法は、表２-1 と同等の情報を持っている。表２-1 では時刻 12 の状況は次の(i)であった。

(i)

time	(Cust)	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
12	odNo4(4)	1	1(2)	0	1(2)	0	1(3)	0	0	-Asm-	0	-Ins-	0

(I')

time	(Cust)	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
12	(odNo4, 12)	1	(1, 9)	0	(1, 9)	0	(1, 9)	0	0	-Asm-	0	-Ins-	0

これら(i)と(i')で、たとえば、AssS の情報は次のように等価である。

(i)　時刻 12 において 1(2)ということは、あと２分後に処理終了

□

(i')　時刻 12 において小部品組立て活動 AssS の値が(1,9)ということは、

（満期 $_{AssS}$(1)＋現在作業の開始時刻)-現在時刻=(5+9)-12＝２分後に処理終了。

つまり、(i)があれば(i')を得られるし逆も可能である。この意味で、(i)と(i')の持つ情報は状態遷移を表現するという意味で等価であり、どちらを用いてもよい。

以上までで、次のことをできるようになった。

（１）離散事象システムを AID によってモデル化することができる。

（２）AID で表現されたシステムの時間変化を記述するためには、AID の中の活動と待ち行列を並べてひとつのベクトルのようにして擬似状態変数 s を表す。

（３）DEVS の内部遷移関数を念頭に置きながら、次に発生するイベントごとに s の変化を表現する。これによって状態遷移表を得ることができる。

問 2-4

図 2-5 の組み立てラインの状態遷移表を得るためには、どんな情報や条件を AID に追加すればよいかを述べよ。

2.8　状態表現：状態と一般状態遷移関数[1]

離散事象システムは activity interactionn diagram (AID)という図的モデルや、discrete event system specification (DEVS) による集合論を使ったモデルで表現した。これらが時間変化する仕組みは図 2-7 のフローチャートと、2.7 節の DEVS を使った例示で示した。精確な時間変化の記述は状態遷移関数（状態表現）によって表される。本節では、必ずしも AID の構成をもたない DEVS 仕様の離散事象システムも含めた一般に DEVS によって定義された離散事象システムに対して、状態遷移関数を得る一般的方法を示す。これによって、AID でモデル化された離散事象システムも、同じ構成方法で状態遷移関数という時間変化の構造を持つことになる。なお、2.7 節では例示によるやや直観的な説明によって、AID でモデル化された離散事象システムが DEVS として記述されることを述べた

[1] 本節は状態遷移の数学的 (集合論的) 記述でありはじめて読む場合には省略してよい。

が、第 4 章で一般化した形で精確に AID モデルを DEVS 仕様に変換する方法を述べる。

2.8.1　時間変化するシステムの状態遷移

一般に時間システムは３種類のいずれかになる。時間軸は連続か離散、変数の変化が連続か離散のどちらかがある。これの組合せで図 2-9 のようになる。

		時間	
		連続	離散
変化	連続	微分方程式	（不可能）
	離散	離散事象システム	オートマトン （離散時間システム）

図 2-9　時間システムの分類

状態遷移関数による記述は、他のモデル、たとえば、連続変化を記述する微分方程式とか、あるいは 1 秒、1 日おきや 1 年ごとの変化を記述する差分方程式とはどういう位置関係にあるだろうか。いずれも共通した状態遷移関数構造というべきものを持っていることが示される。

離散事象システムの時間軸 T は非負実数R^+か、または、非負整数Z^+とする。$r<s$であるような $r,s\in T$ によって定まる区間 $[r,s)$ とは、それぞれ$[r,s) =\{t \mid t\in T, r\le t<s\}$である。$x$ を T からアルファベットAへの時間関数とする（$x:T\to A$）。 シフト演算子σ^{-t}を$\sigma^{-t}(x_{tt'})(k)=x_{tt'}(k+t), 0\le k<t'-t$と定める。同様にして$\sigma^t$を$\sigma^t(x)(k)=x(k-t), k\ge t$と定める。 $t\le t'$なる任意の$t,t'\in T$について関数 x の定義されている範囲を$[t,t')$ に制限したものを $x|_{[t,t')}$とか$x_{tt'}$と書く。すなわち$t\le\tau<t'$である任意のτについて$x_{tt'}(\tau)=x(\tau)$。以下ではx_{0t}，$x_{t\infty}$をそれぞれx^t，x_tとも書く。

x とx'を同じアルファベットへの任意の時間関数とする。このとき、任意の$t\in T$を接続点として新たな関数 x''を次のように定義する：

$$x''(t'')=\begin{cases}x(t''), \text{if } t''<t\\ \text{x}'(\text{t}''), \text{if t}''\ge\text{t}\end{cases}$$

このx'' をx^tと x'_t の連接(concatenation) と呼び、 $x''=x^t\cdot x'_t$と表わす。

さて、状態表現はシステムの動的側面をとらえるための概念枠組みである。
連続時間システムの場合も、離散時間システムでも、離散事象システムでも共通するよ

うな時間変化を記述する共通の考えかたがある。状態空間表現（状態表現）とよばれる。システムを入力関数と出力関数の組みとしてとらえる。

連続時間システム　S

$(x,y) \in S \Leftrightarrow$ 任意の t において、

$$\frac{dz}{dt} = Fz + Gx$$
$$y = Hz$$

$\Leftrightarrow$ 任意の t において $y(t) = H[e^{Ft}z(0) + \int_0^t e^{F(t-s)}Gx(s)ds]$

離散時間システム　S

$(x,y) \in S \Leftrightarrow$ 任意の k において、

$$z(k+1) = Fz(k) + Gx(k), \quad y(k) = Hz(k) + Jx(k)$$

$\Leftrightarrow$ 任意の k において $y(k) = H[F^k z(0) + \sum_{n=0}^{k-1} F^{k-n-1}Gx(n)] + Jx(k)$

いずれの場合にも、初期状態 $z(0)$ と入力によって出力 y が決まることがわかる。さらによく見ると、状態 $z(t)$ は初期状態 $z(0)$ と x_{0t} によって決まっている。
このことを共通して状態遷移関数という関数で表現することができる。

$z(t) = \phi_{0t}(z(0), x_{0t}) = e^{Ft}z(0) + \int_0^t e^{F(t-s)}Gx(s)ds$ （連続システムのとき）

$z(k) = F^k z(0) + \sum_{n=0}^{k-1} F^{k-n-1}Gx(n)$ （離散時間システムのとき）

問 2-5

上の $z(t), z(k)$ がそれぞれ連続システム、離散時間システムの解であることを示せ。（それぞれの方程式の左辺を計算して、右辺の式と一致することを示せばよい。）

状態表現は、システムのクラスに共通な動的側面をとらえるための一般的な（抽象）概念枠組みである。

定義 10.　状態表現

時間システム $S \subseteq X \times Y$ が入力値集合 A と出力値集合 B を持つとする。

. $\Phi = \{\phi_{tt'} \mid \phi_{tt'} : C \times X_{tt'} \to C; \forall t, t' \in T, t \le t'\}$,

$\mu : C \times A \to B$

とするとき 組 $<\Phi,\mu>$ が S の状態表現であるとは(i), (ii)の条件を満たすことである：

(i) 関数集合 Φ が下の(a), (b) を満たす：

(a) $\phi_{tt"}(c, x_{tt"}) = \phi_{t't"}(\phi_{tt'}(c, x_{tt'}), x_{t't"})$. ただし $t \le t' \le t"$かつ $x_{tt"} = x_{tt'} \cdot x_{t't"}$

(b) $\phi_{tt}(c, x_{tt}) = c$

(ii) $(x, y) \in S$であることと、ある $c \in C$があってすべての $t \in T$ について

$$y(t) = \mu(\phi_{0t}(c, x_{0t}), x(t))$$

が成立することが同値である。

C を$<\Phi,\mu>$の状態空間と呼ぶ。Φを（状態）遷移関数族、 その要素を状態遷移関数、μを出力関数と呼ぶ。（図 2-10 参照）

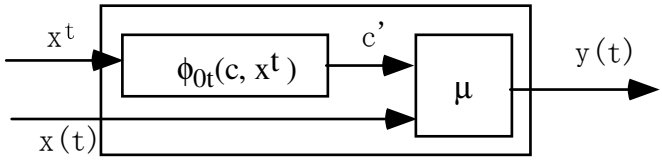


図 2-10 状態遷移関数と出力関数

問 2-6

連続システムと離散システムのそれぞれの状態遷移関数が半群性を満たすことを示す式を、具体的に書きなさい。

この定義で、たとえば、. $\phi_{tt"}(c, x_{tt"})$というのは現在 t の状態が c のときにt から $t"$ の間に入力列(セグメント) $x_{tt"}$ によって、時刻 $t"$でシステムが到達する状態が $\phi_{tt"}(c, x_{tt"})$であることを意図している。（図 2-11）

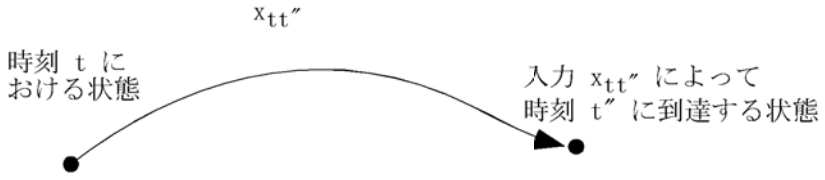


図 2-11 入力による状態遷移

状態遷移関数の中心的性質は、定義の(i) - (a) である。この性質を、状態遷移関数の半群性 semi-group property とか結合性 composition property と呼ぶ。半群性は図 2-12 のように図示される。

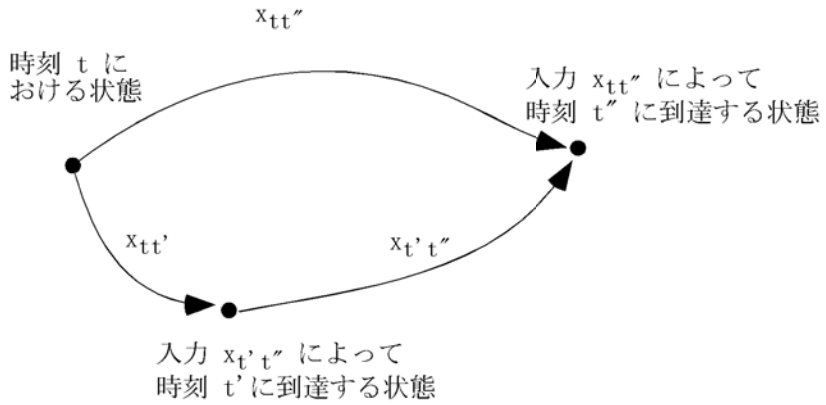


図 2-12 状態遷移の結合性

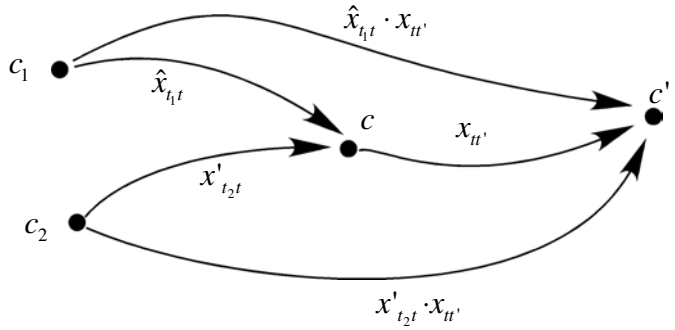


図 2-13 状態による過去の履歴の表現

結合性が意味することは、図 2-13 に示される。現在時刻が t で状態が c であるとする。状態 c_1 は過去の時刻 t_1 における状態であり、入力 $\hat{x}_{t_1t}$ によって状態 c に到達した。また、別の過去の時刻 t_2 において状態 c_2 にあるときは入力 x'_{t_2t} によって c に到達することができる。現在からの入力 $x_{tt'}$ によって時刻 t' において新たな状態 c' 到達するが、これを見直すと時刻 t_1 で状態 c_1 から入力 $\hat{x}_{t_1t} \cdot x_{tt'}$ によっても到達可能であり、また、時刻 t_2 で状態 c_2 から入力 $x'_{t_2t} \cdot x_{tt'}$ による到達も可能である。ここで、$(c_1, \hat{x}_{t_1t})$ や (c_2, x'_{t_2t}) はこのシステムの過去のいきさつであるとか過去の経緯と考えることができる。この際に、現在時刻 t で同一状態 c にあれば、そうした過去の経緯が異なっていてもそれ以降は同じ道をたどることが

結合性（半群性）によって保証されるのである。この意味において、状態はシステムの過去の履歴を代表している情報であるといえる。

これを言い換えると、結合性というのは「現時点での状態」という情報が持つべき性質であり、初期状態と入力列全体によって到達する状態と、入力列を任意に分割した前半の入力セグメントによってある状態に到達しそこからさらに残りの入力セグメントによって到達する状態が等しい、ということを表している。逆に、もし結合性が成立しなければ、入力列の分割のしかたで到達する状態が異なることになって「同じ状態」ということが入力列分割の情報なしには考えられなくなって、非常に煩雑で取り扱いが複雑となる。

微分方程式による連続システムや差分方程式による離散時間システムの解の性質として表現されるように、状態表現は時間システムの動的ふるまいを因果的に認識するために広く用いられている。 Mesarovic and Takahara(Abstract SystemsTheory, 1989)によれば時間システムが因果的であることと、その時間システムに対して状態表現が可能であることは等価である。

遷移関数族 Φと出力関数 μとの組 $<\Phi,\mu>$が定義する入出力システムを$\mathrm{Res}(<\Phi,\mu>)$で表わす。すなわち、(x,y) が $\mathrm{Res}(<\Phi,\mu>)$というシステムの入出力ペアであることを、ある $c\in C$ があってすべての $t\in T$ について $y(t)=\mu(\phi_{0t}(c,x_{0t}),x(t))$ であると定める。$<\Phi,\mu>$がSの状態表現であるときは、もちろん$\mathrm{Res}(<\Phi,\mu>) = S$である。

オートマトンの場合には、$[r,s)=\{t\mid t\in T, r\le t<s\}=\{r, r+1, r+2,..., s-2, s-1\}$であるので、$x_{0t}$は結局は入力値の並びそのものである。そして、一般状態遷移関数$\phi_{0t}(c,x_{0t})$はオートマトンの状態遷移関数δを$x(0), x(1), ..., x(t-1)$の t 個の入力によって順次、c から状態遷移したものとなる。状態遷移関数の半群性は、状態遷移関数の拡張の定義によって、もちろん成立する。

また、定係数微分方程式で記述される時間変化をもつシステムの状態遷移関数も、やはりこの一般状態遷移関数として記述される。（行列を使って表現される線形離散時間システムと線形微分方程式の場合の状態遷移関数の半群性は付録とした。）こうしたことから、一般状態遷移関数は、時間システムの時間変化を考える上での共通した認識の枠組みである。離散事象システムについては今のところ方程式表現は発見されていないが、我々は DEVS を使っている。当然、離散事象システムの時間的変化を表す DEVS についても、DEVS によって工学的に「正しく」時間変化をとらえるために利用できることを保証するためには、状態遷移関数が DEVS によって定まり、かつ、それが結合性を満た

している必要がある。それが示されなければ、離散事象システムを都合良くいい加減にとらえてしまっていて何か重要な情報を見落としていたりする可能性が残るのである。

2.8.2 DEVS 離散事象システムの状態遷移

任意の DEVS $M =< A_M, S_M, B_M, \delta_M, \lambda_M, ta >$を考える。以下では、DEVS の構成要素を用いて状態遷移関数を構成する。

入力：入力セグメント集合という集合Ω_Gを次で定める。

$\Omega_\Lambda = \{\Lambda_t : [0,t) \to \{\Lambda\}, t \in T\}$

$\Omega_A = \{a_t : [0,t) \to A_M \cup \{\Lambda\}, t \in T\}$、ただし$a_t$は次のような関数である:

$$a_t(t') = \begin{cases} a, & \text{if } t' = 0 \\ \Lambda, & \text{if } 0 < t' < t \end{cases}$$

$\Omega_G = \Omega_A \cup \Omega_\Lambda$

問 2-7

$t = 3$の場合の関数a_tのグラフを示せ。

状態遷移関数：疑似状態遷移関数 $\delta_M : Q_M \times (A_M \cup \{\Lambda\}) \to S_M$ の定義域を $Q_M \times \Omega_G$に拡張して$\delta_G : Q_M \times \Omega_G \to Q_M$にする：

$\delta_G : Q_M \times \Omega_G \to Q_M$

- $\delta_G(s,e,\Lambda_t) = \begin{cases} (s, e+t), & \text{if } e+t \le ta(s); \\ (\delta_G(\delta_\Lambda(s), 0, \Lambda_{e+t-ta(s)}), & \text{if } e+t > ta(s); \end{cases}$

  （$e+t > ta(s)$ とは$e+t$ 以内に s の内部処理が終了することを意味）
- $\delta_G(s,e,a_t) = (\delta_G(\delta_M(s,e,a), 0, \Lambda_t)$

上のδ_Gの定義は、$\delta_\Lambda(s)$という s の内部処理にはかならず一定時間以上の処理時間がかかるという条件のもとで well-defined（不明確でない）である。この条件を内部処理の有限時間消費条件と呼ぶ。δ_Gの定義より、これがないと無限に内部処理を続ける可能性がある。

有限時間消費条件：

任意の正の数Δと内部状態 s について適当な整数kがあって,$\sum_{i=0}^{k} ta(\delta_\Lambda{}^i(s)) \geq \Delta$
を満たすこと。つまり、sから何回かの内部遷移をすると必ず満期時間がΔ以上の値になること。

もし有限時間消費条件を満たさない DEVS があったとすると、ある特別な内部状態sについては内部遷移を何回行っても$ta(\delta_\Lambda{}^k(s)) = 0$のままで満期であり続け、いつまでたってもそれ以上の状態の変化をしないし時間が経過しないという異常なことになる。なお、有限時間消費条件は Zeigler 著で legitimacy 性（定義の妥当性・正当性）として定義された条件と同値である。

この「状態遷移関数」が本当に状態遷移関数であるためには、オートマトンの場合と同様に、$\delta_G^+ : Q_M \times \Omega_G^* \to Q_M$ と拡張されたものが、半群性を満たす必要がある。ただし、オートマトンは時間が離散的(discrete time = 0, 1, 2, ...) であったから文字列関数への拡張が自然に可能であったが、今は、連続時間の場合もありうるので、離散時間で連続時間でも同じ表現で扱える一般的な形を後で示す。

状態出力関数： $\mu : Q_M \times (A_M \cup \{\Lambda\}) \to Q_M$ を次で定義する：

$$\mu(s,e,z) = \begin{cases} (s,e), & \text{if } z = \Lambda \text{ and e < ta(s);} \\ (\delta_\Lambda(s),0), & \text{if } z = \Lambda \text{ and e = ta(s);} \\ (\delta_M(s,e,z),0), & \text{if } z \neq \Lambda \end{cases}$$

上の出力関数 μ は、単に状態を出力するものである。もし、たとえば、いくつかの待ち行列に関する瞬時値が必要な場合には、さらにこれからその部分を取りだせば（射影関数で）よい。

2.9.3　離散事象システムの状態遷移関数の定義と半群性

DEVS から定まる離散事象システムの入力時間関数Xを、値の集合を $A_M \cup \{\Lambda\}$とする離散事象入力空間とする。ここで、□は空イベントを意図するシンボルで、A_MはΛを含まないと仮定されている。また、離散事象入力空間とは、数学的には、Xの任意の要素と任意の区間$[t, t')$に対して $x|_{[t,t')} = x_{tt'}$の値のうちの$x(t'') \neq \Lambda$となるような時刻 t'' は有限個しかないことであると定める。（図 2-13 参照）

Xの任意の要素と任意の区間$[t, t')$ に対しては $x_{tt'}$の左最長分解が存在する。ここで左最長分解とは$x_{tt'}$の左から順に$\Omega_G = \Omega_A \cup \Omega_\Lambda$の可能な要素をとり、その結合で$x_{tt'}$を表わすことである。上の図の場合、 $x_{0t'} = a^1 \cdot b^2 \cdot c^3 \cdot d^4$ である。

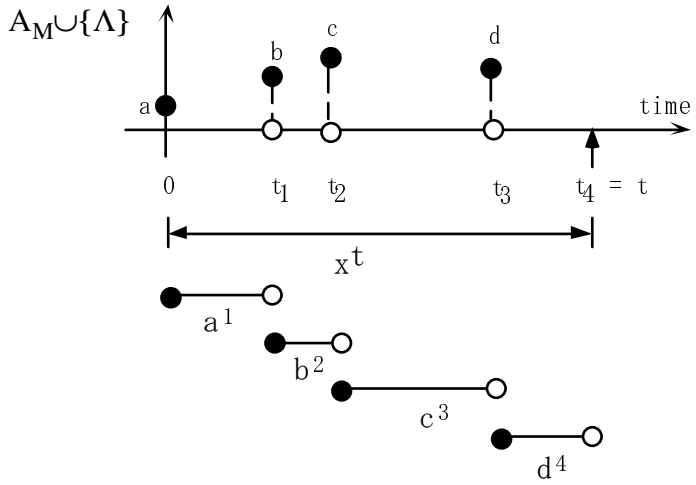


図 2-13 離散事象入力の左最長分割

任意の $x_{tt'} \in X_{tt'}$ についてその定義域を $D(x_{tt'})$で表わす。つまり$D(x_{tt'}) = [t,t')$である。 x_{0t}の左最長分解を $x_{0t} = a^1 \cdot b^2 \cdot ... \cdot c^n$ とする。

したがって $a^1(0) = a$、かつ$D(a^1) = [0,\ t_1)$のとき$\tau(0 < \tau < t_1)$について$a^1(\tau) = \Lambda;$

$b^2(0) = b$、かつ$D(b^2) = [t_1, t_2)$のとき$\tau(t_1 < \tau < t_2)$について$b^2(\tau) = \Lambda;$

などである。また、 $D(c^n) = [t_{n-1}, t),\ t_n = t$である。

関数の集合$\Phi = \{\phi_{tt'} \mid \phi_{tt'} : Q_M \times X_{tt'} \to Q_M; \forall t,t' \in T, t \le t'\}$を次のように定義する。

$\phi_{0t}(s, e,\ x^t) = \delta_G(...(\delta_G(\delta_G(\mu(s,e,a^1(0)), \Lambda^1), \sigma^{-t_1}(b^2)), ...), \sigma^{-t_{n-1}}(c^n))$

$\phi_{tt'}(s,\ e,\ x_{tt'}) = \phi_{0\tau}(s,\ e,\ \sigma^{-t}(x_{tt'}))$

ここで $\tau = t' - t$と定める。

問 2-8

関数$\sigma^{-t_1}(b^2)$のグラフを示せ。

以下でこの関数が半群性をもつこと、したがって Φ が（時間不変な）状態遷移関数の集合であることを示す。

いま、例としてx_{0t}の左最長分解を$a^1 \cdot b^2 \cdot c^3 \cdot d^4$として、半群性を示す。$D(a^1)=[0,\ t_1)$, $D(b^2)=[t_1,t_2)$, $D(c^3)=[t_2,t_3)$, $D(d^4)=[t_3,t_4)$, $t_4=t$とする。

時間不変性が定義によって成立しているので（つまり$\phi_{tt'}(s,\ e,\ x_{tt'})=\phi_{0\tau}(s,\ e,\ \sigma^{-t}(x_{tt'}))$であるので)、半群性を示すには
$\phi_{0t}(s,e,\ x^t)=\phi_{t't}(\phi_{0t'}(s,e,\ x^{t'}),\ x_{t't})$が任意の$t'$ について成立することを示せば十分である。

$x^t=x^{t'}\cdot x_{t't}$であるが、t' が$[0,t)$のどこに位置するかによって２とおりに場合分けする。
（１） $x(t')\neq\Lambda$のとき。
（２） $x(t')=\Lambda$のとき。

（１） $x(t')\neq\Lambda$の場合。$t'=t_2$のときを一般に仮定する。したがって今の場合$x(t')=c$である。

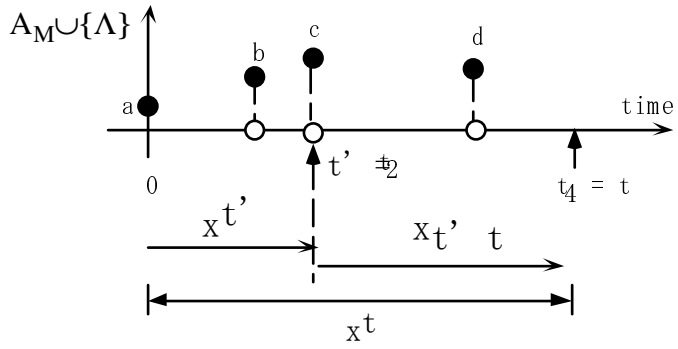


図 2-14　場合（１）

$x_{t't}$の左最長分解は$c^3\cdot d^4$である。
$\phi_{0t_2}(s,e,\ x^{t_2})=\phi_{0t_2}(s,e,\ a^1\cdot b^2)$
$=\delta_G(\delta_G(\mu(s,e,\ a^1(0)),\Lambda^1),\sigma^{-t_1}(b^2))=(s',e')$
とおく。
$\phi_{0t}(s,e,x^t)=\delta_G(\delta_G(\delta_G(\delta_G(\mu(s,e,a^1(0)),\Lambda^1),\sigma^{-t_1}(b^2)),\sigma^{-t_2}(c^3)),\sigma^{-t_3}(d^4))$
$=\delta_G(\delta_G(s',e',\sigma^{-t_2}(c^3)),\sigma^{-t_3}(d^4))$
$\phi_{t't}(\phi_{0t'}(s,e,x^{t'}),x_{t't})=\phi_{0t''}(s',e',\sigma^{-t'}(x_{t't}))$　　($t''=t-t'$とおいた)
$=\phi_{0t''}(s',e',\sigma^{-t'}(c^3\cdot d^4))$
$=\delta_G(\delta_G(\mu(s',e',\sigma^{-t_2}(c^3)(0)),\Lambda^3),\sigma^{-t_3}(d^4))$
よって$\delta_G(s',e',\sigma^{-t_2}(c^3))$と$\delta_G(\mu(s',e',\sigma^{-t_2}(c^3)(0)),\Lambda^3)$が等しければよい。

$\sigma^{-t_2}(c^3)(0) = c^3(t_2) = c$ より、

$\delta_G(s',e',\sigma^{-t_2}(c^3)) = \delta_G(\delta_M(s',e',c),0,\Lambda^3) = \delta_G(\mu(s',e',\sigma^{-t_2}(c^3)(0)),\Lambda^3)$ が成立する。

（２） $x(t') = \Lambda$ のとき。

1° $t_1 < t' < t_2$ の場合を示す。

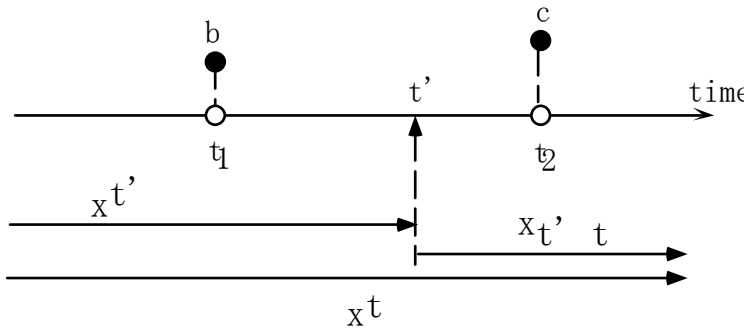


図 2-15　場合（２）

2° それぞれの入力の左最長分解は

$$x^t = a^1 \cdot b^2 \cdot c^3 \cdot d^4$$
$$x^{t'} = a^1 \cdot b_{t_1 t'}$$
$$x_{t't} = \Lambda_{t't_2} \cdot c^3 \cdot d^4$$

3° このとき

$$\phi_{0t'}(s,e,x^{t'}) = \phi_{0t_2}(s,e,a^1 \cdot b_{t_1 t'})$$
$$= \delta_G(\delta_G(\mu(s,e,a),\Lambda_{0t_1}),\sigma^{-t_1}(b_{t_1 t'}))$$
$$= \delta_G(\delta_M(\delta_G(\mu(s,e,a),\Lambda_{0t_1}),b),0,\sigma^{-t_1}(\Lambda_{t_1 t'}))$$

ここで、 $s_0 = \delta_M(\delta_G(\mu(s,e,a),\Lambda_{0t_1}),b)$ とおくと、

$$\phi_{0t'}(s,e,x^{t'}) = \delta_G(s_0,0,\sigma^{-t_1}(\Lambda_{t_1 t'}))$$

となる。

4° 一方、$[t_1,t_2)$ において、図 2-16 のように内部遷移がおこるとする。つまり、

$$t_1 + ta(s_0) + ta(\delta_\Lambda(s_0)) + ta(\delta_\Lambda^2(s_0)) < t_2 \le t_1 + ta(s_0) + ta(\delta_\Lambda(s_0)) + ta(\delta_\Lambda^2(s_0)) + ta(\delta_\Lambda^3(s_0))$$

が成立していて $[t_1,t_2)$ においては内部遷移は３回だけ起こるとする。このように有限回の内部状態遷移で t'以上の時刻まで満期時間が経過することを仮定してよいのは、有限時間消費条件が成り立っていることが前提であることに注意されたい。

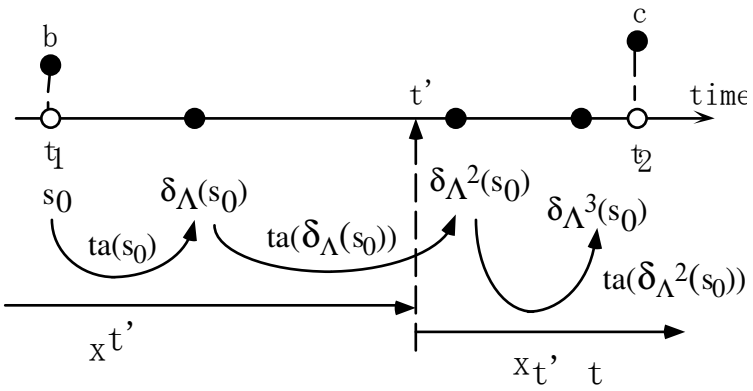


図 2-16　３回の内部遷移

5°　この時、

$$\delta_G(s_0,0,\sigma^{-t_1}(\Lambda_{t_1t_2})) = \delta_G(\delta_\Lambda(s_0),0,\sigma^{-t_1-ta(s_0)}(\Lambda_{[t_1+ta(s_0),t_2)}))$$
$$= \delta_G(\delta_\Lambda^2(s_0),0,\sigma^{-t_1-ta(s_0)-ta(\delta_\Lambda(s_0))}(\Lambda_{[t_1+ta(s_0)+ta(\delta_\Lambda(s_0)),t_2)}))$$
$$= \delta_G(\delta_\Lambda^3(s_0),0,\sigma^{-t_1-ta(s_0)-ta(\delta_\Lambda(s_0))-ta(\delta_\Lambda{}^2(s_0))}(\Lambda_{[t_1+ta(s_0)+ta(\delta_\Lambda(s_0))+ta(\delta_\Lambda{}^2(s_0)),t_2)}))$$
$$= (\delta_\Lambda^3(s_0), t_2-[t_1+ta(s_0)+ta(\delta_\Lambda(s_0))+ta(\delta_\Lambda{}^2(s_0))])$$

である。

6°　同様にして、3°の計算を続けると

$$\phi_{0t'}(s,e,x^{t'}) = \delta_G(s_0,0,\sigma^{-t_1}(\Lambda_{t_1t'})) = (\delta_\Lambda(s_0), t'-[t_1+ta(s_0)])$$

を得る。

7°　よって、Φの定義を使って次式を得る：

$$\phi_{t't}(\phi_{0t'}(s,e,x^{t'}),x_{t't}) = \phi_{t't}(\delta_\Lambda(s_0), t'-[t_1+ta(s_0)],\sigma^{-t'}(\Lambda_{t't_2}\cdot c^3\cdot d^4)) \quad (\because 6^\circ)$$
$$= \delta_G(\delta_G(\delta_G(\delta_\Lambda(s_0),t'-[t_1+ta(s_0)],\sigma^{-t'}(\Lambda_{t't_2})),\sigma^{-t_2}(c^3)),\sigma^{-t_3}(d^4))$$

8°　この式の内部遷移を計算すると、

$$\delta_G(\delta_\Lambda(s_0),t'-[t_1+ta(s_0)],\sigma^{-t'}(\Lambda_{t't_2}))$$
$$= \delta_G(\delta_\Lambda^2(s_0),0,\sigma^{-[t_1+ta(s_0)+ta(\delta_\Lambda(s_0))]}(\Lambda_{[t_1+ta(s_0)+ta(\delta_\Lambda(s_0)),t_2)}))$$
$$(\because\quad t'+ta(\delta_\Lambda(s_0))-(t'-[t_1+ta(s_0)]) = t_1+ta(s_0)+ta(\delta_\Lambda(s_0)) \text{ より})$$
$$= \delta_G(\delta_\Lambda^2(s_0),0,\Lambda_{[0,t_2-t_1-ta(s_0)-ta(\delta_\Lambda(s_0)))})$$
$$= \delta_G(s_0,0,\sigma^{-t_1}(\Lambda_{t_1t_2})) \qquad (\because\ \Omega_G \text{の定義より})$$

9°　よって、

$$\phi_{t't}(\phi_{0t'}(s,e,x^{t'}),x_{t't}) = \phi_{0t''}(s',e',\sigma^{-t'}(x_{t't}))$$
$$= \delta_G(\delta_G(\delta_G(s_0,0,\sigma^{-t_1}(\Lambda_{t_1t_2})),\sigma^{-t_2}(c^3)),\sigma^{-t_3}(d^4)) \quad (\because 7^\circ, 8^\circ)$$

$= \delta_G(\delta_G(\delta_G(\delta_M(\delta_G(\mu(s,e,a),\Lambda_{0t_1}),b),0,\sigma^{-t_1}(\Lambda_{t_1t_2})),\sigma^{-t_2}(c^3)),\sigma^{-t_3}(d^4))$

（$\because 3^\circ$ による s_0 の置き換え）

$= \delta_G(\delta_G(\delta_G(\delta_G(\mu(s,e,a),\Lambda_{0t_1}),\sigma^{-t_1}(b^{t_2})),\sigma^{-t_2}(c^3)),\sigma^{-t_3}(d^4))$　（$\because$　δ_Gの定義）

$=\phi_{0t}(s,e,x^t)$　（$\because$　ϕ_{0t}の定義）

t' が $p=0,1,2$ のいずれかについて $t_1+\sum_{k=0}^{p} ta(\delta_\Lambda^k(s_0))$と等しい場合についても同様の計算によって半群性が成立する。

以上の（１）、（２）の場合よりΦが半群性を満たし、その結果状態遷移関数の集合であることがわかった。全体的な構成を絵で描くと図 2-17 となる。

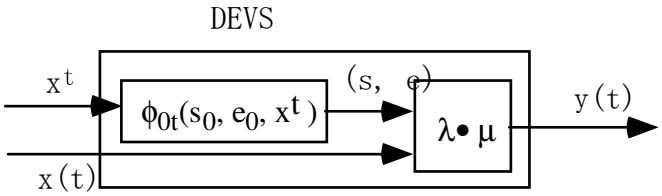


図 2-17　DEVS から構成される状態表現

DEVS 仕様Mは６つ組で $M=<A_M,S_M,B_M,\delta_M,\lambda_M,ta>$で定義されるが、これまでのことから、有限時間消費な DEVS については、状態遷移関数の拡張が常に可能であり、それがあれば DEVS が定める離散事象システムの状態遷移関数$<\Phi,\mu>$が常に定まる。

また、状態をそのまま出力する$<\Phi,\mu>$という状態表現を考えるときには、もともとの出力関数 λ_M は不必要なので、DEVS を単に$<S_M,\delta_M,ta>$とかくことも、さらに添字 M を省略して$<S,\delta,ta>$とかくこともある。DEVS に対して定まる$<\Phi,\mu>$という状態遷移メカニズムを DEVS の状態システムと呼び、S_Dとかく。ダイナミックスを表すという意味で D という添字を使っている。 $S_D=\mathrm{Res}(<\Phi,\mu>)$であって、

$(x,z)\in S_D\subseteq X\times(Q_M{}^T)\Leftrightarrow$

（ある $(s,e)\in Q_M$ があってすべての $t\in T$ について$z(t)=\mu(\phi_{0t}(s,e,x_{0t}),x(t))$）

によって定義されている。

第２章末問題

問題１．クリスマスカードゲーム

ウォマックとジョーンズの図 2–1 のゲームを、クリスマスカードを作るゲームに変形してカンバン方式の練習にする。クリスマスカードゲームと呼ぶことにする。

クリスマスカードを作ろう！

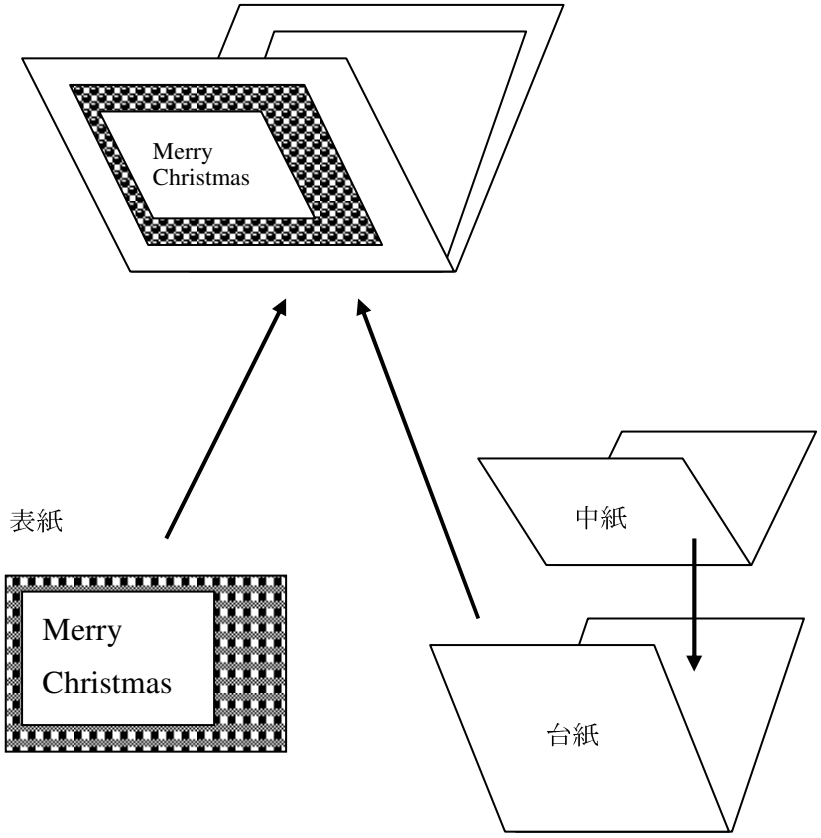


図 2-18 クリスマスカードゲーム

１．概要

６人一組となってひとつの工場となり，多品種（８種類）の製品を作る。いくつかの生産方式を体験する。

２．部品構成表 (派生 BOM：bill of material)

製品：クリスマスカード。表紙に４種類あり，台紙の中のメッセージに２種類ある。表紙１枚と台紙１枚から，ひとつの製品ができる。よって８通りの製品がある。製品名

を X1 から X8 までとする。
部品：表紙が４種類ある。中紙が２種類ある。

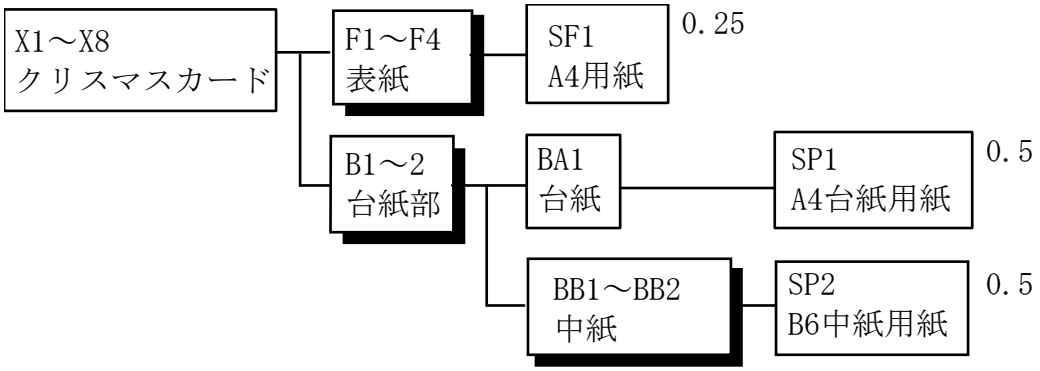


図 2-19　クリスマスカードの部品構成表

３．製造手順 (routing)

カンバンを使用しない場合の作業手順（生産活動の手順で routing と呼ばれる）は、図 2-20 のようになっている。

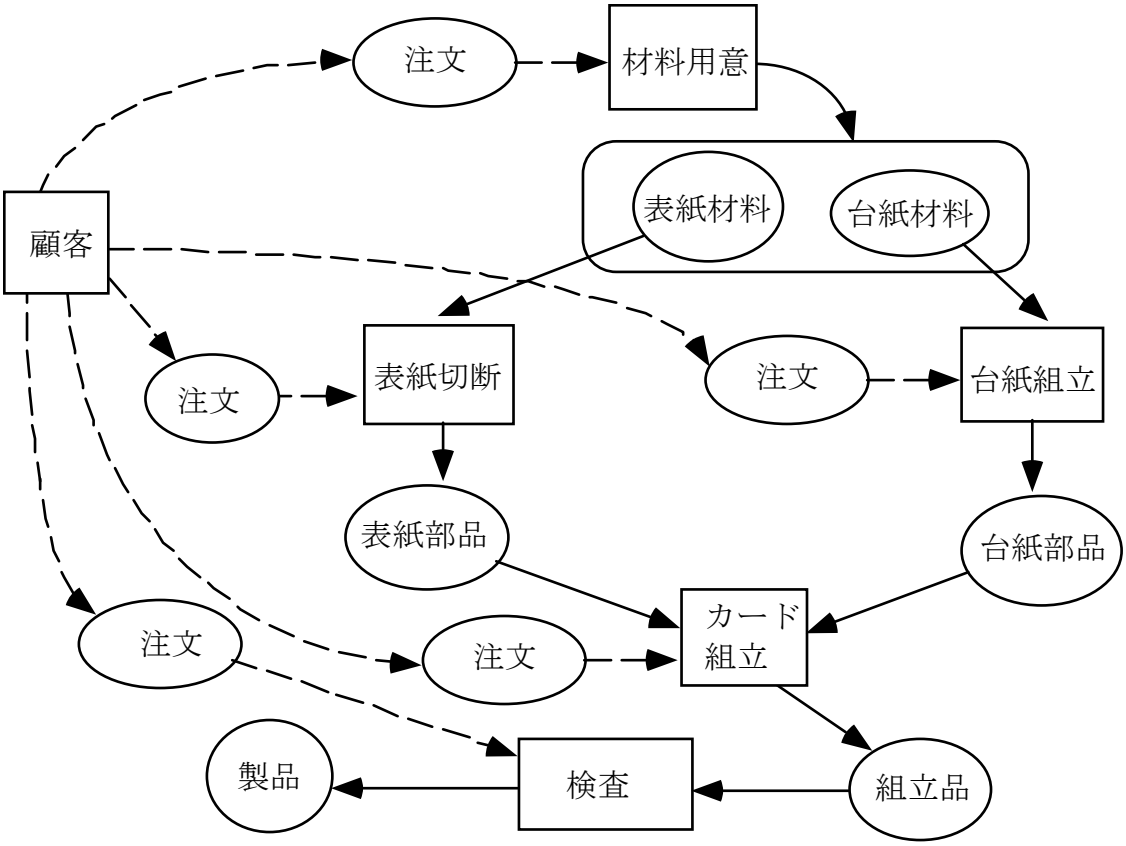


図 2-20　作業手順

４．作業内容

（１）最初の人　No.1---原料供給の倉庫係

まだ折り畳んでいない 3 種類の箱に折る紙を 2 ヶ所の組立準備工程へ供給する。

供給量は顧客からの受注量を考慮して決めるが，できるだけ余分な材料を教官倉庫から取りださないことを目標とする。

（２）No.2 の人----　表紙の切断

倉庫係から表紙の用紙を受け取り，４つの表紙を作成する。

ジグザグバサミできれいに切り取る。（ジグザグバサミは切り口がジグザグの模様になるはさみで、DIY ショップ等で販売している。）

できたものを表紙部品と呼ぼう。表紙部品をカード組立係のところに置く。

（３）No.3 の人----　台紙の組立

倉庫係から台紙と中紙を受け取り，台紙部品を組み立てる。中紙には２種類ある。

台紙と中紙をそれぞれ２等分してカッターで切り離す。台紙にちょっとだけのりずけして，そこに中紙を貼る。

できあがった台紙部品を，カード組立係のところに置く。

（４）No.4 の人----　製品の組立（カード組立）

表紙と台紙部品から最終製品であるカードを作る。

表紙を台紙部品に貼り付ける。それから，製造日報にカードの種類と数を記入し保持する。できたカードは，検査出荷係のところに置く。

（５）No.5 の人----　製品の検査と出荷

製品となったカードについて，いくつかの点を検査して顧客に出荷する。

次の点をチェックする：

　　紙がきちんと折れていること。

　　折り線が中央にあること，

検査日報にカードの種類と数を記入し保持する。

製品を注文ごとにまとめて顧客のところに運ぶ。

（６）No.6 の人---- 顧客

注文を発する。

あらかじめ決められたように，いろいろな製品を注文する。

注文した各製品を受け取った時刻と個数を記録する。

５．使用する用具と材料

はさみ（ジグザグ７つ）
カッターナイフ（７つ）
のり（14 個）
クリップ
付箋紙 （完成品を束ねて，オーダー番号を付箋紙に書いて顧客へ渡す）
集計用紙（材料受け取り記録用。材料係）
各種原材料（BOM を見ること）表紙，中紙，台紙
カンバンの紙

６．生産方式１：カンバンなしの生産

6-1. 実行手順の概略

(1) 練習に，初期に各材料ををを５個ずつ作って置いておく。これを初期在庫とする。
(2) 注文は生産指示書としてすべての工程に一度に伝えられる。
(3) 納品は顧客からの注文単位に行う。つまり，分納しない。
(4) 顧客への納品の実績は，中断時に計算する。
(5) できるだけ早く納品するためにどういうふうにやるかを 5 分間程度相談する。その後，実習本番に入る。

6-2. 実習

初期在庫をそろえ、開始する。顧客は，教官からの指示によって，注文を検査係の注文表に書く。各工程係は，生産指示カンバンによって注文量を製造する。10 分から 15 分間程度に渡り作業を黙々と続ける。

客からの発注についていくつかのパターンを試みる。

適当に，各作業のスピードを上回る程度に，いろいろな製品の注文を全行程に一斉に出す。発注される製品総数を予め決めておく。たとえば、X1ーX5 は各 15 個、X6ーX8 は各６個等とする。最後の注文が終了したところで終了する。なお、どの程度の注文頻度でどこかのボトルネックに在庫が積み上がるかの実験は，状態遷移表を出力するようなプログラムを作ってコンピュータシミュレーションでも観察できる。

各作業者は，注文書＝製造指示書であるとして扱う。

終了時に，各班の結果（数）を黒板に書く。

班名，台紙材料 SP1, SP2，台紙部品 B1，B2，表紙材料，表紙部品 F1, F2, F3, F4，組立品 X1, X2, X3, X4, X5, X6, X7, X8，完成オーダー番号列）

6-3. 分析

何についての差を観察できるか、あらかじめ各グループごとに話し合う。

１．納期,

２．製造リードタイム（生産開始時刻と終了時刻を記録しておく）

３．工程中の中間在庫（終了時）

４． ボトルネックとなる工程

５．全体的なビジネスプロセス管理のしやすさ

７．生産方式２：カンバンによる生産管理

図 2-21 のようなプロセスによって生産する。

顧客は，検査係だけに注文を出す。

製造は２個単位の固定ロットで，製造を下流から製造指示カンバンで指示されたときにのみ製造する。つまり，作るときは，一度に２個だけを作成する。カンバンが２枚くれば４個作ることになる。

自分の製造する材料を使用するときに，２個なくなったら，前工程にカンバンを出す。（注文は２個単位ではなく１個もあれば５個もある。）

なお、生産指示カンバンだけを用いる。ひとつのテーブルで実施し、作業と作業の間が空間的に近いため、いわゆる引き取りカンバンは使わない。

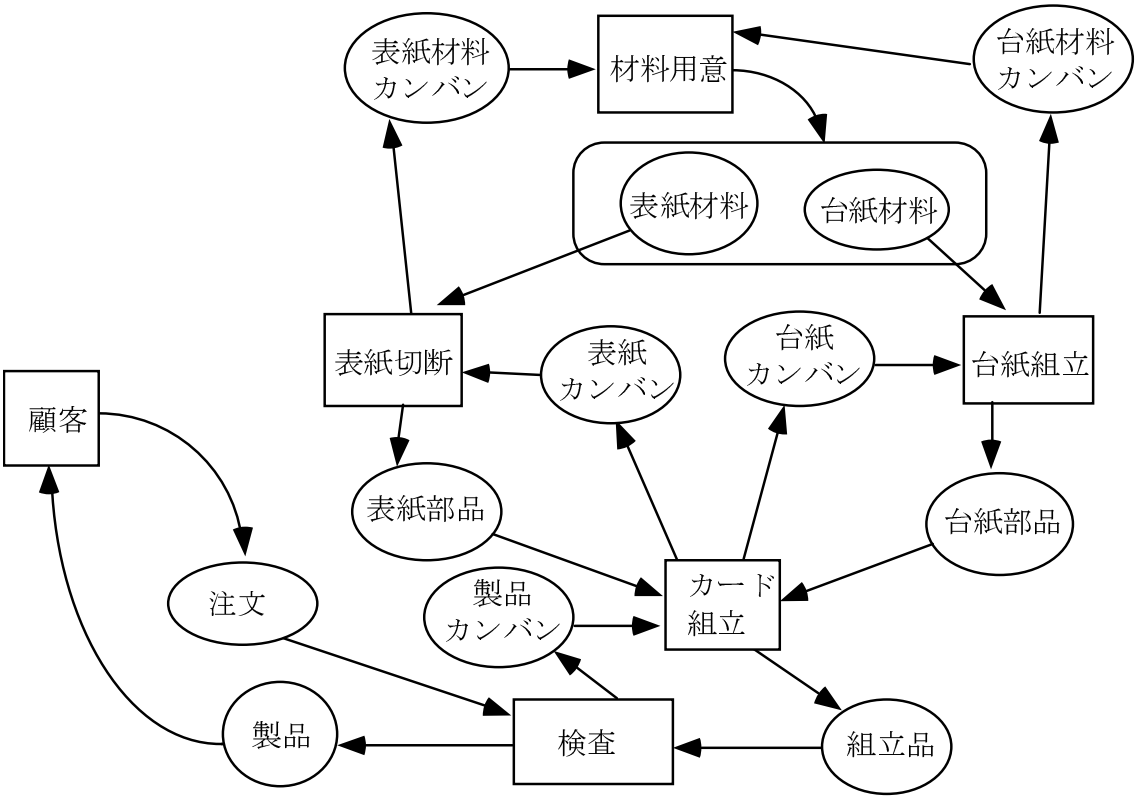


図 2-21 カンバンによる生産管理

7-1. カンバンを準備する

各工程で用いるカンバンを作成する。品目名、作業名、個数＝２、カンバン番号（たとえば、３枚中の２枚目のときは、2/3 とかく）。A4 用紙半分の大きさで、下図のようにつくる。各項目は作業前に作業者が書き込む。

品番　　B１ 品名　　台紙 収容数　　２ 発行番号　１／２	工程名 台紙組立 後工程 カード組立

注文は図 2-21 の検査係にだけ送られる。倉庫係から材料という形で上流から下流に製造指示がでていく。

7-2. 実習

５個単位の固定ロットで，製造を下流から指示されたときにのみ製造する。つまり，作るときは，一度に５個の整数倍だけを作成する。

注文は５個単位でなくてよい。

（カンバンケース１） ５個の中間在庫を常にもつ。

（カンバンケース２）固定ロットによる５個未満の在庫以外は，中間在庫を５個持つ。２個作るごとに看板にカンバンによって，前行程に２個分の製造指示をだす。

全製品だと５×８＝４０個の在庫を持つことになるが，部品で考えると，どうか。

もし作ろうとして，材料がないときは，上に製造指示カンバンを出す。納品は，５個単位でおこなう。

カンバンを使わないときと同様の分析を行い、各パフォーマンス指標を比較する。

問題 2.

次の説明は、購買手続きについての概要である（H. Pickle and R. Abrahamson 著, Small business management Wiley, 1990.）。企業等の組織の社員や部署が外部から購入する際には、基本的に購買を担当している部署を通して購入する。経理部が購買担当になっている場合も多い。この説明をもとにして、購買業務の AID をモデル化しなさい。モデル化の過程において、説明にはないが活動を行う上で必要なモノやデータが明確になる場合も多い。必要なデータは必ずどこかの誰かが作成している必要がある。

ある社内部署から，購買業務担当へ，購買依頼 が送られる。購買依頼は標準化された文書であるべきである。その内容は，品目，数量，希望納期，当方担当者である。品目は，間違いのないように完全に記述される必要がある。複数のベンダーに対して，購買希望品目についての内容を問い合わせて，購買担当者の決定にもちいることにも用いられる。購買のきっかけとなる。

・ベンダーの評価

あらかじめ購買方針を設定しておき，それに基づいて行う。３社はそろえることとか，価格についての検討指針とか，さらにデータを要求するか，など。納期見積りの正確さ，たとえば製品設置のサービスの有無，提供される割引の種類，ベンダーの立地，複数のベンダーが可能か，現金割引の可能性，販売員。

・価格決定

いくつかの価格決定方法がある。最近のカタログで価格入のものを参考にする，ベンダーと交渉して決める，入札。入札を行うときには，購入希望品目の仕様を完全に規定する必要があるという手間がかかる。

・購買発注

この文書によって，購買が法的に確定する。書面になっている必要があり，内容は，購買品目（またはサービス），数量，価格，配送，その他のこと（届け先とか）である。

・購買管理

ベンダーが購買注文を受け取ったことを確認し，指定期日に品物を届けるように，進捗管理を行う。

たとえば，4 枚のワンライティング・コピーによる発注伝票を作る。白２枚，ピンク１枚，青１枚。青をファイルして発注済みで未入荷の品目一覧として用いる。白２枚をベンダーに送り，そのうちの一枚の受注確認用伝票は返送してもらう。ピンクをファイルしたものは１週間おきにチェックして，まだ確認が来ていないものは必要に応じて手を打つ。ベンダーから確認を受け取った時点で，ピンクを発注順に発注記録ファイルとして追加する。納品時に青を取り除き，支払い記録に使う。

・受け取り検収

受け取り時には，品目の記述，価格，数量，納品日について，発注通りかどうかを突き合わせる。何らかの違いがある場合には，それに応じたアクションをベンダーとの間で行う。

・発注の記録と支払い

納入が終了されたら発注データは記録され，購買管理のためのデータとなる。また，同時に支払いが行われる。

第 3 章　データモデルとデータモデリング機能

問：自分が見たり考えうるどんな文書や現象も、かならずデータとして整理して表現できるようになりましたか。これが本章を読んでよくデータモデリング機能を理解したかどうか判断するときに、自問すべき問いである。

3.1　データモデル（TH データモデル）

データモデルは現実世界を、データを使って記述するための概念と表現方法を述べたものである。つまり、現実世界をデータの世界に写像する・写し取る・モデル化する概念枠組みがデータモデル／データモデリング機能である。データモデリング機能を使って得られたデータモデルがデータベースの設計である。

データモデリング機能にはいろいろなものがこれまでに提案されてきた。本書では TH データモデリング機能を使う。TH は穂鷹良介著「データベースシステムとデータモデル」や椿正明著「概念データモデル」においてオリジナルなアイデアが提案されている。本書では､基本的部分のみを取り出した簡略版 TH を使うので、もとの本格的 TH と区別するために DAE データモデルと呼ぶ（ドメイン-アトリビュート-エンティティ）。

なお、TH は別のポピュラーなデータモデルである ER モデル（Entity-relationship モデル）の P. Chen（陳）博士オリジナルのアイデアの本質部分を取り出しているともみなすことができる。ER の説明書は数多くそれぞれに細かな差違があるが、チェン先生の論文の根本的アイデアをつかまえにくくなっているように見えるものもある。たとえば、Chen 博士が例に出す関係にはほぼ必ず主キーがあるので概念的にもデータベースで使う場合にもに意味があるが、ER の類書では主キーのない多対多関係などいうものを無造作に作成している場合もある。これに対して、結果として ER の本質部分のみを取り出したような、データについての独自の深い概念化と簡潔な表現形式を整え、そうすることで実用的なデータモデルを得つための概念枠組みとなったのがデータモデリング機能が TH（DAE）データモデルであるといえる。また、現実の宅配便の伝票を見てそのデータモデルを作成したり、さらに、亜流 ER によって作られたデータモデルの完成度を評価できる目を持てるようになるという意味で、TH や DAE は実用的である。SAP R/3 のような統合基幹情報システム(ERP - enterprise resource planning パッケージ）はデータベース管理ソフトウエアの上に、実務向けレディーメイドの部品表や注文表等の主要なデータのテーブルを持つ。全部で 10,000 以上のテーブルを持ち､主要なものだけでも 3,000 から 4,000 ものテーブルから成る。R/3 のデータモデル図をエンドユーザが見ることができ､そのデータモデリング機能は拡張 ER と呼ばれていたが、その際に ER から「不自然な」概念である relationship

を除いてあり、結果としてエンティティ・タイプとそれらの間の TH 的な関係概念（本章で後述）によって表現されていた。このように、大規模な全社的情報システムのデータモデルや、さらに、他のデータモデルもかなり程度「読みこなせる」ための基本概念からなっている。また、渡辺著（業務別データベース設計のためのデータモデリング入門、日本実業出版社）のように ER のリレーションシップを TH と同じ立場から限定して使い、その結果、実質的に TH と同様のデータモデルを得られるような本が出て、かなり成功した本となっている。なお、TH（DAE)とデータモデリング機能である ER や UML のクラス図との相互関連は４節で説明する。

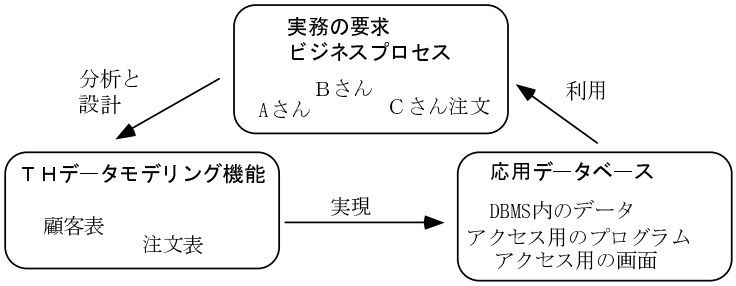


図 3-1　データモデルとその実現

データモデリング機能は，データを記述する概念の集まりである。データモデルとそれに関連する概念の関係が図 3-1 に示されている。データモデリング機能を使って，たとえば、顧客管理システムで用いられるデータを格納し利用するためのいくつかのテーブル（表）の設計図ができる。データモデルはテーブルの設計図であるので、DBMS（データベース管理システム）というソフトウエアを使って，実際にコンピュータの上に作り上げることができる。これを，データモデルを実現するという。

DBMS は多くのソフトウエア製品として利用可能である。Oracle, SQL server, DB2, Access, MySQL, PostgreSQL などが有名で実際によく利用されている。歴史的に，また現在でも，COBOL などのプログラム言語を使って実現されたデータモデルを扱うこともある。データベースは大量のデータを扱うのでずいぶん前に制作されたソフトウエアや応用データベースが長期間使用されていることがあり、レガシーシステム（既設システム、遺産的ソフトウエア／データ、ちょっと遅れた技術を使っているシステム）と呼ばれる場合がある。また、データ倉庫等のデータの分析には通常のビジネス取引を記録し参照できるようにする DBMS とは異なる仕組みが必要で、その用途に特化した DBMS も作られている。

現代のデータベースは、インターネットからアクセスすることが普通の形態となっている。そのためには、ウエブサーバと DBMS を連係させて構成される分散システムがよく用いられる。さらに、Web サービスのようにいくつかの分散システムどうしがネットワークを介して連係するために、XML と SOAP が用いられる。XML で表現されたひとまとまりのデータの定義は XML スキーマとよばれるが、これも

ネットワーク上を流れるデータのデータモデルである。また、電子商取引(EDI-electric data interchange)においては、もともと紙ベースでチェーンストア等の業界で標準化されていた注文伝票や納入伝票などが、電子的データとして表現され標準化される際にデータモデル化された。建設とチェーンストアでは異なる商品を扱う場面もあるというように各業界で独特の伝票やデータ項目を用いる。業界ごとに標準伝票の種類やデータ項目を定めた規定書があり、それをダウンロードできる。ただし、それらの EDI のためのデータモデルは、図 3-2 の応用データベースのテーブル定義の部分に対応する表現となっており、伝票で用いる属性を表すデータ項目と桁数などが規定されている。

データモデルという概念を理解する上で､図 3-2 の３つの世界を区別することが役立つ。データモデルとデータモデリング機能は、現実世界をモデル化する概念の集まりである。それは DBMS を使って実際に作られるソフトウエアとは独立である。したがって、顧客管理のために設計されたデータモデルを、いろいろな DBMS で実現できるし、また、データの量や処理速度要求に応じた実現上の工夫を DBMS ごとに行う。伝票保存に対応して DBMS ではログと呼ばれる取引記録を確実に保存することや、分散システムでのセキュリティの保証の仕組みも実現の時の重要事項だが、データモデルとは独立である。DBMS を取り替える場合も、わが社が扱うデータの設計図であるデータモデルが必要となる。

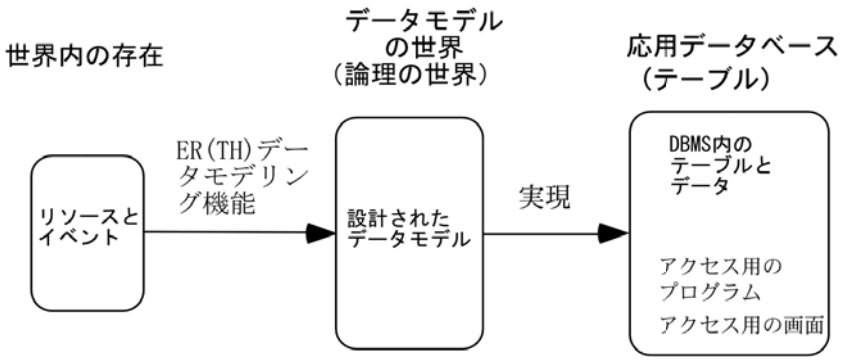


図 3-2　データモデルはデータベースの設計図

3.2 DAE データモデリング機能

DAE データモデリング機能は、わずか 5 種類の概念だけを使ってビジネスで扱われるすべてのものをデータモデル化する。5 種類の概念とは、管理実体型、属性、ドメイン、参照関係、継承関係である。

（1）管理実体型

現実世界の存在をデータとして表現する場合、存在をイベントとリソースの 2 種類のグループに分けてとらえる（椿、1997）。

「イベント」グループには、ビジネス上の売り買いのような取引という存在が含まれる。イベントの代わりにトランザクションと呼ぶこともある。イベント実体の例としては、顧客の佐藤さんから西暦 1975 年 8 月 10 日にコーラ 30 ダースとビール 55 箱を受注したという個別の注文が存在する。注文、支払、購買、クレーム、生産計画は、ビジネス上重要であるので個々に管理される必要がある。また、すべて現実世界で一つずつ発生するので、DBMS で実現されるときにテーブルの中に個々のレコードとして記録される必要がある。一方、「リソース」グループの存在には、顧客、製品、学生、運賃といった、個々の取引ごとには発生しないが、個々の取引の意味を確定する際に必要となる存在が含まれる。モデルが DBMS で実現される際には、顧客一覧表や製品マスター、学生一覧表や運賃一覧表のように、マスターテーブルとか一覧表となるものである。

2 種類のグループの個々の存在を実体と呼び、意味によって分けた実体の集まりを実体型という。「実体型」とは実体の型ということであるが、佐藤さんや John さんという人が顧客である時、彼らは個別の顧客実体であり、顧客実体をまとめた顧客という概念が存在する。このような実体をまとめた実体（存在）がである。言い換えれば、実体のグループの名称が実体型である。顧客はリソースグループの実体型である。また、取引発生ごとに注文実体が発生するが、注文実体をまとめた「注文」という概念が存在する。注文はイベントグループの実体型である。

ビジネス遂行のために管理し保持すべきであると認識された実体型を、管理実体型という。注文や顧客などは、ほとんどのビジネスにとって管理実体型である。

（2）属性と主キー

顧客や注文はビジネス遂行上必要であるから管理しなければならないので、管理実体型である。管理するためには、個別実体を過不足なく明確に記述・表現することが必要となる。同じ管理実体型に属する個別実体を区別するのには、個別実体の違いを表現できるようないくつかの側面を利用する。観察や記述のためのこうした側面を属性（attribute）と呼ぶ。

たとえば、個々の顧客を人間社会において区別するために、会社名、住所、電話番号、担当者名、担当者メールアドレス等を属性として用いる。他の属性として体重や身長、あるいは職業とか年令、入社年月日、既婚未婚区別、借家持家区分など、どこまで記述を細かくするかは、自然的必然的に決まるのではない。

管理実体型という概念の管理というのはビジネス遂行のために管理し保持すべきことであるが、後述するように、ビジネスプロセスは活動の集まりであり、離散事象システムと呼ばれる全体構造を持っているので、管理するということは管理実体のどれかの属性データをある活動が使用するかどうかによって決まる。どの活動も使用しないようなデータを属性とする必要はない。たとえば、個別の人間顧客を構

成する水素原子の個数を顧客管理実体型の属性にしていることは、少なくとも現代ではない。測定できるかという以前に、そのデータを使うビジネス活動がないからである。個人認証に使うようにでもなれば、ひとつの属性にすることになる。電話番号とか、メールアドレスも昔はなかったそうした属性と考えられる。

データモデルの原型は、コンピュータよりはるか以前に人間が言語を使い紙に記録できるようになったころから存在したと思われる。属性の選択は、広い意味のビジネスや記録テクノロジーに深く関係する。

管理実体型の中で、ある属性について同一値を持つ実体がただひとつしかないときに、その属性はその管理実体型の主キー属性であるという。TH や DAE では、すべての管理実体型がそれぞれの主キー属性という特別の属性を持つことを要請する。つまり、DAE を使ってデータモデルを設計する時に、主キー属性を持てない管理実体型があれば、それは、考え方やとらえ方が不十分であったり勘違い等の何かの間違いがある。

主キー属性を単に主キー（primary key）ということもある。主キーの例は、顧客管理実体型の顧客番号、学生管理実体型の学籍番号、宅配便伝票の伝票番号等である。顧客管理実体型で、顧客の名前は、同姓同名の可能性がありうるので主キーにはなり得ない。また、顧客の電話番号を主キーに使えるかどうかはビジネスによるが、多くの場合は無理がある。基本的に、主キー属性は人工的に逐次的にふっていく番号を使う。「顧客番号」のように「管理実体型の名称＋番号」を主キー属性にする。他の可能性は何か特殊事情がないかぎり考えない。DBMS でテーブルとして実現されている場合、顧客番号は顧客コードと言い換えられていることも多い。

データモデルを作成するとか設計する場合に、属性はどのように決まるだろうか。顧客管理実体型を例として説明する。新たな顧客実体を顧客に加える時、主キー属性はこれまでの最大番号に１を加えたものにする。顧客管理実体型のすべての属性が定まらないか、あるいは、記入しきれないとすると、不必要な属性が紛れ込んでいる可能性がある。

たとえば、ひとつの顧客会社が本店と複数の支店を持っていると、支払先が本店ひとつでも顧客への発送先住所が複数あるのが普通だから、顧客管理実体型の属性として「発送先住所」があると記入できない。顧客自体を表現し管理する場面において、複数の発送先を持っている事実までも含めようとする概念的混乱が、属性に記入できない／記入しきれないという現象としてあらわれている。つまり、発送先住所は顧客管理実体型の属性ではない。では何の管理実体型の属性であろうか。この場合は、発送するという活動を想定すると、顧客が複数の発送先を持っているという事実は、顧客が存在するという事実とまったく別の事実であるということに思い至る。よって、「発送先」管理実体型が存在する。主キーは発送先番号で、属性の中には顧客番号と住所を用意し、もし一つの会社が５つの届け先を持つなら、一つの会社と住所の対応という５つの実体がある。この実体の各々は、ひとつの会社

が発送先住所を持つという５つの事実を表現し、それぞれが異なる事実として利用されうるので、個別に管理されるのである。

主キーという概念は重要である。データモデル設計において「すべての管理実体型がそれぞれの主キー属性を持つことを要請する（主キー制約）」ことによって、後述するような形式的な正規形の議論が実質的に不要になる。つまり、結果として正規形になっているデータモデル設計が得られることが多い。逆に言えば、データモデルの定義として主キーをもたない管理実体型の存在を許すと、理論的に無駄に複雑になるばかりであるし、また、実用からかけ離れた使えない定義になる。顧客と発送先の状況のように、実体を集めたが主キー番号をつけられないという状況は、いくつかの事実を混ぜこぜにしてしまって属性を適当に並べていくつかの属性値に空欄があったりしながら、複数のことがらを記述しようとしていることが多い。それでは、意味が多様で定まらないから使えないのである。

主キー制約がないと、現実に使われている伝票１枚に表されたデータをつきつけられても、あまりに多くの分析の方向が可能であるために、結果として分析できず設計もできないという事態に陥ってしまう。とっても繁盛しているケーキ屋さんには、多くのお客様から予約注文が来る。ほとんどのお客さんは、いちどの予約で、たとえばモンブランやミルフィーユ、イチゴショートなどを数個ずつ注文する。ケーキ製造のために、それらの注文をバラバラにしてケーキごとの製造個数にまとめたのはよいが、どのケーキがどのお客様向けのものだったか。注文番号をつけて注文ごとにケーキの個数を管理しておかないと、できあがったケーキを穴があくほどじーっと見つめても、行き先のお客様の顔が浮き上がってくるわけではないのである。なお、特定業務に限らず、受注のデータは後出の図 3-10 の「受注」と「受注細目」の２つの管理実体型を使って用意して管理する場合がほとんどである。

管理実体型を DBMS を使って実現する場合、一つの管理実体型は DBMS のひとつのテーブルとなる。図 3-3 は学生一覧表のテーブルのイメージである。図で最上段の「学生」が管理実体型の名称であり、その下段の学籍番号、氏名、氏名よみが３つの属性、学籍番号が主キー属性の実現である。例として、武田信玄という人間存在が、（228019, 武田信玄, たけだしんげん）というデータの並びで表現されている。属性(attribute)はテーブルの欄として実現される。

学生		
学籍番号	氏名	氏名よみ
228019	武田信玄	たけだしんげん
228021	武田信義	たけだのぶよし
…	…	…

図 3-3　テーブル（表）による管理実体型の実現（学生一覧表）

y			
a_1	a_2	a_3	a_4
v_{11}	v_{12}	v_{13}	v_{14}
v_{21}	v_{22}	v_{23}	v_{24}
v_{31}	v_{32}	v_{33}	v_{34}

図 3-4　表による管理実体型の実現

管理実体型 y という表の属性が$\{a_1, a_2, a_3, a_4\}$であることを $A(y) = \{a_1, a_2, a_3, a_4\}$ とかく。図 3-3 の例では、A(学生) = {学籍番号, 氏名, 氏名読み}である。A(学生)という属性の並びは、学生という存在を、これらの属性に対応した 3 種類のデータで表現するということを示している。

表の中のデータはひとつの行である。ひとつの行は各属性に対応する値を持つものであり、図 3-4 では 1 行目のデータは$(v_{11}, v_{12}, v_{13}, v_{14})$という並び(list) で表される。また、y 表の主キーが a であることを key(y) = a と表す。いくつかの属性の組みが主キーになることもある。管理実体型名 y や属性 a_1, a_2, a_3, a_4 は、テーブルに格納されるデータの構造を規定するデータであり、この意味でデータのデータであるので、メタデータと呼ばれる。データベースの設計とはメタデータの設計ということである。

（3）ドメイン

属性を決めると同時に、属性が取りうる値の種類を決める必要がある。TH や DAE データモデリング機能では、各属性に対応する値の集合としてドメイン (domain) という集合を指定する。たとえば、氏名に対応するドメインを名前という集合とするときに dom(氏名) = 「名前」と表す。属性の取りうる値は、現実世界を記述するときに人間にとって意味を持つようなある集合の要素である。その集合がドメインということである。

大学の学生一覧表（図 3-3）の例のように学生管理実体型をテーブルとして DBMS で実現するときには、ドメイン「名前」は string のようなデータ型となる。

異なる属性が同じドメインを持つことも当然ありうる。たとえば、教科書管理実体型に著者名という属性が使われていて、dom(著者名) = 名前 ということがありうる。

ドメインの代わりにデータ型と呼べばよいように見えるかもしれないが、図 3-2 のように、データモデルは DBMS とは独立な論理的世界として存在する必要がある。それによって、データモデルの設計を検討することができるし、また、わが社のデ

ータモデルを特定の DBMS で実現するときや移行するときにも利用できる。

ここまでで説明された、管理実体型、属性、主キー、ドメインの概念を使ってひとつひとつの管理実体型を記述できるようになった。

（4）参照関係（Rキー）

現実社会には、物理的制約や人間社会の取り決めに由来する制約や条件が存在する。これらの制約や条件を、複数の管理実体型の間に２つの条件を課すことにより、たくさんの管理実体型からなる集合全体が満たすべき一貫性条件として記述する。２つの一貫性条件とは参照整合性と継承整合性である。直感的には、参照整合性条件の例をテーブルの実現形であらわしたものが図 3-5 である。

経営工学メールアドレス	
メールアドレス	学生
jkato@tsukuba.ac.jp	229025
…	…

学生		
学籍番号	氏名	氏名よみ
228019	武田信玄	たけだしんげん
228021	武田信義	たけだのぶよし
…	…	…

図 3-5(a) 経営工学科の学生は本学の学生だ

顧客			
顧客番号	会社名	会社名よみ	代表連絡先
1025	MedServe	メッドサーブ	029-345-1001
…	…	…	…

受注			
受注番号	顧客＃	受注日	受付従業員
3022998	232901	1999/12/23	8091
…	…	…	…
…	…	…	…

図 3-5(b) 受注は顧客から来る

図 3-5　参照整合性（参照キー）

学生管理実体型は属性として学籍番号、氏名、氏名よみ、所属を持ち、主キーは

学籍番号である。経営工学科メールアドレスは、経営工学科の学生に発行されたメールアドレスの一覧である。経営工学科の学生は大学の学生だから学生IDカードを所持して自分の学籍番号を割り当てられている。したがって、経営工学メールアドレスの中に表れる学生とは、かならず、大学の学生である。いい換えれば、経営工学メールアドレスを持つ学生をすべて集めても、大学の学生の集合に含まれている。

この事実をコンパクトに表現するには、図 3-5(a) にテーブル表現されたように、経営工学科メールアドレスの方で学生を指示するために学生実体型の主キーである学籍番号を、経営工学メールアドレス実体型の学生属性として用いる。つまり、経営工学科メールアドレス実体型の中で、ある学籍番号を指定すると、その学生は学生実体として必ず存在し、しかも唯一であって、かつ、氏名・氏名読み・所属を分かることができる。２つの属性である、学籍番号と学生とは同じドメインを持つ。また、経営工学メールアドレスの属性には、学籍番号が定まると該当する学生の氏名や所属は学生管理実体型を見れば分かるので、学生の氏名や所属は属性としては不要となる。参照関係の構成規則として、主キー以外の属性を「参照」することは禁止される。この例では、経営工学メールアドレスが学籍番号以外の学生実体型の属性を持つことは、冗長性を防ぐために禁止される。

受注の管理実体型における顧客属性として、顧客管理実体型の主キーである顧客番号を使うことについても事情は同じである。当然、dom（経営工学メールアドレス.学生）= dom（学籍番号）と、dom（受注.顧客）= dom（受注.顧客番号）が成り立つ。ここで、「受注.顧客」などは、属性 A が管理実体型 B のものであることを明示するための B.A という表現の例である。

一般に、ある管理実体型 A の主キー属性の値だけを取りうるような属性を持った管理実体型 B があるときに、**B は A を参照する**と定義する（穂鷹著, 1989）。同じことをいい換えていくつかの言い方がされている：

- **B のその属性は A（の主キー）を参照する。**
- **B のその属性は、A からの参照キーである。**
- **B 管理実体型は A 管理実体型を参照する。**
- **A と B には参照関係がある。B と A には参照関係がある。**

参照関係で重要な点は、受注の顧客のように、B が参照するのは、必ず他の管理実体型 A の主キー属性に限るということである。管理実体型 A の主キー属性でない属性と同じドメインを持つ属性を、別の管理実体型 B が使っていても、参照関係はない。主キーでない属性間の関係は意味がないのである。主キーでない属性間の関係性は、データモデルとしては重要性がないばかりか、それを参照関係と見なすようなデータモデルを使うと分析が無意味に複雑化し無方向化する。

つまり、ビジネス上の重要性があって意味を持たせる必要があれば、かならず、

その属性が主キーになっている管理実体型を認識して取り出して、データモデルを設計する。いわば、関係を解きほぐすわけである。

（５）単純属性と複合属性

属性には単純属性と複合属性がある。複合属性はいくつかの属性が組み合わされてできているものである。たとえば、販売業務システムにおいて、受注明細管理実体型の主キーは、受注番号と商品番号という２つの属性の組みから成っている場合がある。このときにこれら２つの組みを新たにひとつの属性として、たとえば受注明細コードと認識し定義するならば、受注明細コードは複合属性である。また、入社年月日という属性が年、月、日という三つから成る場合も、複合属性である。

少し細かい説明を追加する。受注明細管理実体型の主キーとして、原則に従って受注明細番号を使うこともできるし、（受注番号, 商品番号）という組を使うこともできる。違いをケーキ屋さんの例で述べる。（受注番号, 商品番号）を受注明細の主キーにできるということは、イチゴショートを２個注文したら、もはやイチゴショートは同じ注文の中には書かない（書けない）ということである。もしどうしてもイチゴショートを３個追加したいときは、先のイチゴショート２個を訂正して、イチゴショート５個に変更するか、または、別の注文伝票を使って同じ顧客からの新たなイチゴショート３個の注文とする。後者の場合、引き取り可能時間が２個分と３個分でずれることはある。

一方、受注明細番号を主キーとしているときには、同じ注文書の異なる明細として、イチゴショート２個の注文の何行か下にイチゴショート３個が書かれてもよいことになる。

一般の注文でも、同じ商品だが２個は緊急的に必要だが、３個の納期は２週間以内でよいことを、同じ注文伝票で指定したいこともあれば、それは別の受注として扱うものとしてビジネスを行うことも可能である。逆に、何らかのビジネス上の理由で、（受注番号, 商品番号）の組み合わせをただ一つに限りたい時にはこの組を主キー属性にする。

（６）継承関係（汎化と専化、派生関係）

管理実体型の間の一貫性条件である継承整合性条件の例を、TH の表記法である右下に向かう矢印を使って図 3-6 に示す。経営工学学生管理実体型と学生管理実体型との間の継承関係であり、それぞれの管理実体型をテーブルで実現した場合の図である。あらっぽく言うと、経営工学学生が学生を詳しくした実体型であるときに、それぞれ継承関係にあるという。経営工学学生の方が学生実体型よりも詳しく記述されており、この場合、経営工学管理実体型は学生管理実体型の専化（特殊化）であるという。同じことを、学生管理実体型は経営工学管理実体型の汎化（一般化）であるという。専化する方法は，属性を加えていくことだから、ひとつの表に対し

て無数の専化表がありうることに注意しよう。

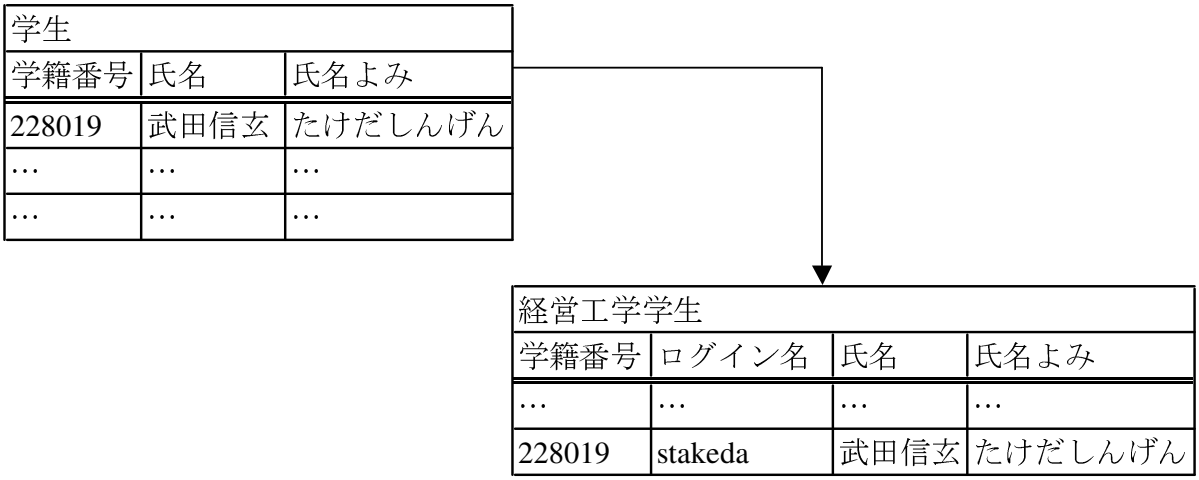


図 3-6　継承整合性（継承関係）

経営工学学生管理実体型が学生管理実体型の属性を継承する (inherit)とは、以下の４つを満足することであると定義される（穂鷹, 1989）。

１）それぞれキー属性が一致していること。つまり同じ主キーを持つ。

２）経営工学学生管理実体型は学生管理実体型のキー以外のすべての属性も含む。つまり、経営工学学生管理実体型はキー以外の属性を継承する。

３）経営工学学生管理実体型の任意の実体に対して、その行のキー値と等しい値を持つ学生管理実体型の実体が存在する。

４）キー属性値での値が一致すれば、学生管理実体型からのキー値以外の継承属性値もそれぞれ一致する。

図 3-6　のときに、管理実体型を実現したテーブル（表）に対しても同じ言葉を使って関係を表現する。学生表は経営工学学生表の汎化(generalization)であるとか、経営工学学生表は学生表の専化 (specification)であるとか、経営工学学生表は学生表の属性を継承する(inherit)という。汎化は図の場合のようにいくつかの表を観点を抽象化し、したがって属性数を減らして、共通化して全体をとらえたいときに用いられる。個人顧客も法人顧客も顧客に汎化されるし、さらに、顧客もサプライヤも取引先として汎化されうる。また、オブジェクト指向の構造をもつプログラム言語やデータベースの場合では、属性だけでなくメソッドも継承し、重要な概念となっている。

なお、データベース用プログラム言語 SQL では、ひとつの表から、同じ主キーを持ついくつかの汎化表を得る場合に、それをビュー表 (view table)、または派生表という。継承関係の条件の 2)を緩めて、派生表の属性として汎化表の主キーだけを持てばよいものとして使う。

参照キーや継承関係を TH で図 3-7 のようにも描くことがある。

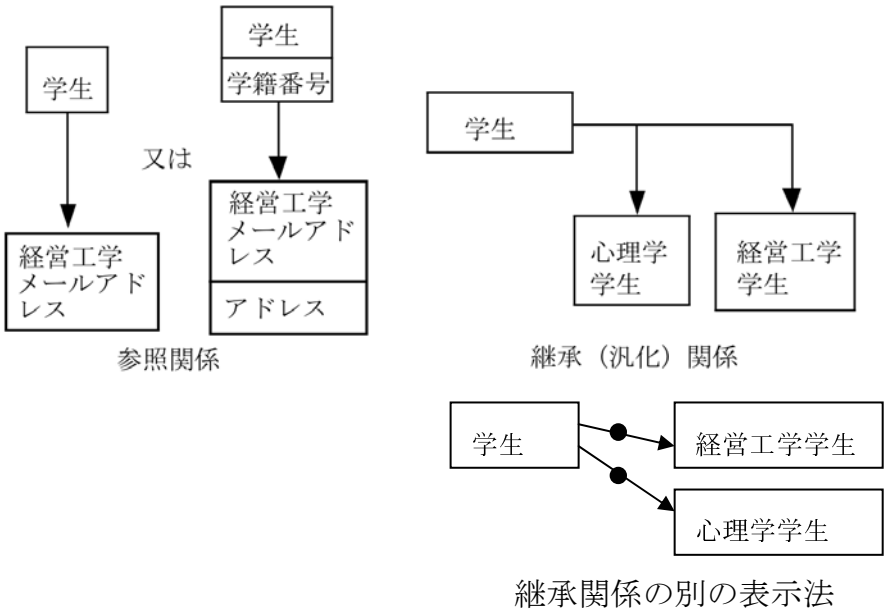


図 3-7　参照キーと継承関係の図示

3.3　メタデータベース

（1）データモデリング機能

DAE データモデリング機能によるデータモデル化とは、管理実体型、属性と主キー、ドメイン、参照関係、継承関係という５つの概念を使って対象領域のデータの分析を行いその結果を表現することである。表現されたものはメタデータベースと呼ばれるものであった。それは、対象とするデータについて、データモデリング機能を使って得られたデータの設計図である。TH と同様に DAE データモデリング機能はあらゆるデータをモデル化できるので、DAE データモデリング機能を使ってメタデータベースを表現できる。DAE で DAE 自体をモデル化できるわけである。得られたデータモデルは、データベース管理システム(DBMS)を使って作られるファイルシステムの設計図であって、それによって、ファイルシステムの論理構造が完全に定義される。穂鷹著で示された方法を紹介する。

DAE データモデリング機能のデータモデルを５つの管理実体型によって表現できる。メタデータベースの表による表現とデータモデルは次の表のようになり、図 3-8、図 3-9 と図示できる。図 3-8 では、メタデータの例として図 3-5(a)と図 3-6 の表のデータを示してある。

メタモデルにおける管理実体型の記号	DAE の概念
¢FTS	管理実体型
¢AS	属性
¢DS	ドメイン
¢REFER	参照関係
¢INHERIT	継承関係

¢FTS のテーブル表現を表表と呼ぶ。同様に¢DS、¢AS、¢REFER、¢INHERIT、¢ISA の表を、それぞれ、ドメイン表、属性表、参照表、継承関係表、汎化関係表と呼ぶ。ある具体的なデータモデルの設計結果は、これらの表の中に格納されたデータとして表されることになる。この５つのテーブルをメタデータベースという。

表表は情報システム（あるいは業務システム）として、どれほどのテーブルを持っているかを表す。表識別子は表表の主キーである。ドメイン表のドメイン識別子は主キーである。ＫＳ性は、そのドメインがＫＳであるときにＹ（yes の意図）を書く。属性表の表属性識別子は主キーである。属性表の表識別子の値は、その属性がどの表の属性なのかを示す。同様に属性表のドメイン識別子の値は、その属性のドメインを示す。キー性は、属性がその表の主キーかどうかを表し、主キーであるときＹの値をとる。参照表は属性表に書かれた属性間の参照関係を表し、(a,b) ε REFER のとき a を subAS の値とし b を superAS の値とする行が参照表のなかにある。継承関係表、汎化関係表についても参照表と同様である。

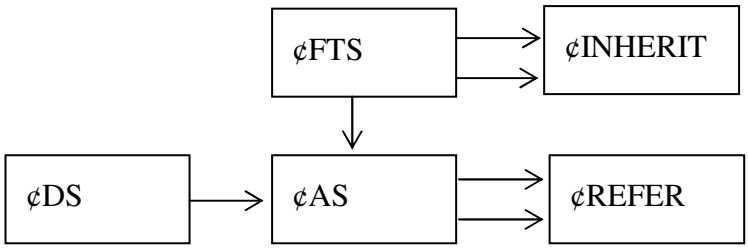


図 3-8 DEA のデータモデル

表表	
表識別子	表名
50001	学生
50002	経営工学メールアドレス
50003	経営工学学生

ドメイン表		
ドメイン識別子	ドメイン名	KS性
11	学籍番号	Y
12	氏名読み	
13	氏名	
14	メールアドレス	Y
15	ユーザID	

属性表				
属性識別子	属性名	管理実体型	ドメイン	キー性
1	学籍番号	50001	11	Y
2	氏名よみ	50001	12	
3	氏名	50001	12	
4	学生	50002	11	
5	メールアドレス	50002	14	Y
6	学籍番号	50003	11	Y
7	氏名よみ	50003	12	
8	氏名	50003	13	
9	ログイン名	50003	15	

参照関係表		
参照識別子	subAS	superAS
61	4	1

継承関係表		
継承識別子	subFTS	superFTS
201	50003	50001

図 3-9 DAE メタデータのテーブル表現のイメージ

（2）メタデータの図示法

椿正明著「データ中心システム入門」の記法をつかって、より分かりやすくデータ設計図を表現する。バックマンダイアグラムと呼ばれる表記法の拡張である。

以下に説明する描き方のルールにしたがって、ある営業支援データベースのメタデータを描くと、あとのページの図 3-10 のようになる。

描き方のルール：

（1）一つの表は、たとえば、受注表のように一つの長方形ボックスとして描く。
（2）ボックスの中には、表の名前と、その属性を書く。たとえば出荷先住所表は、出荷先番号を主キーとして、（出荷先名、出荷先住所１、出荷先住所２、出荷先都道府県、出荷先郵便番号、会社番号）という属性をもつ。このとき、主キーはカギカッコ［ ］で囲み、他の属性はその後に並べる。
（3）さらに参照キーは、ボックス間を矢印→でつなぐと同時に、受注表の中の会社番号のように、会社番号と下線をかく。参照関係が明らかな場合は下線は省略も可とする。
（4）管理実体型の箱の配置は、参照キーの矢印が、上から下、左から右になるようにする。

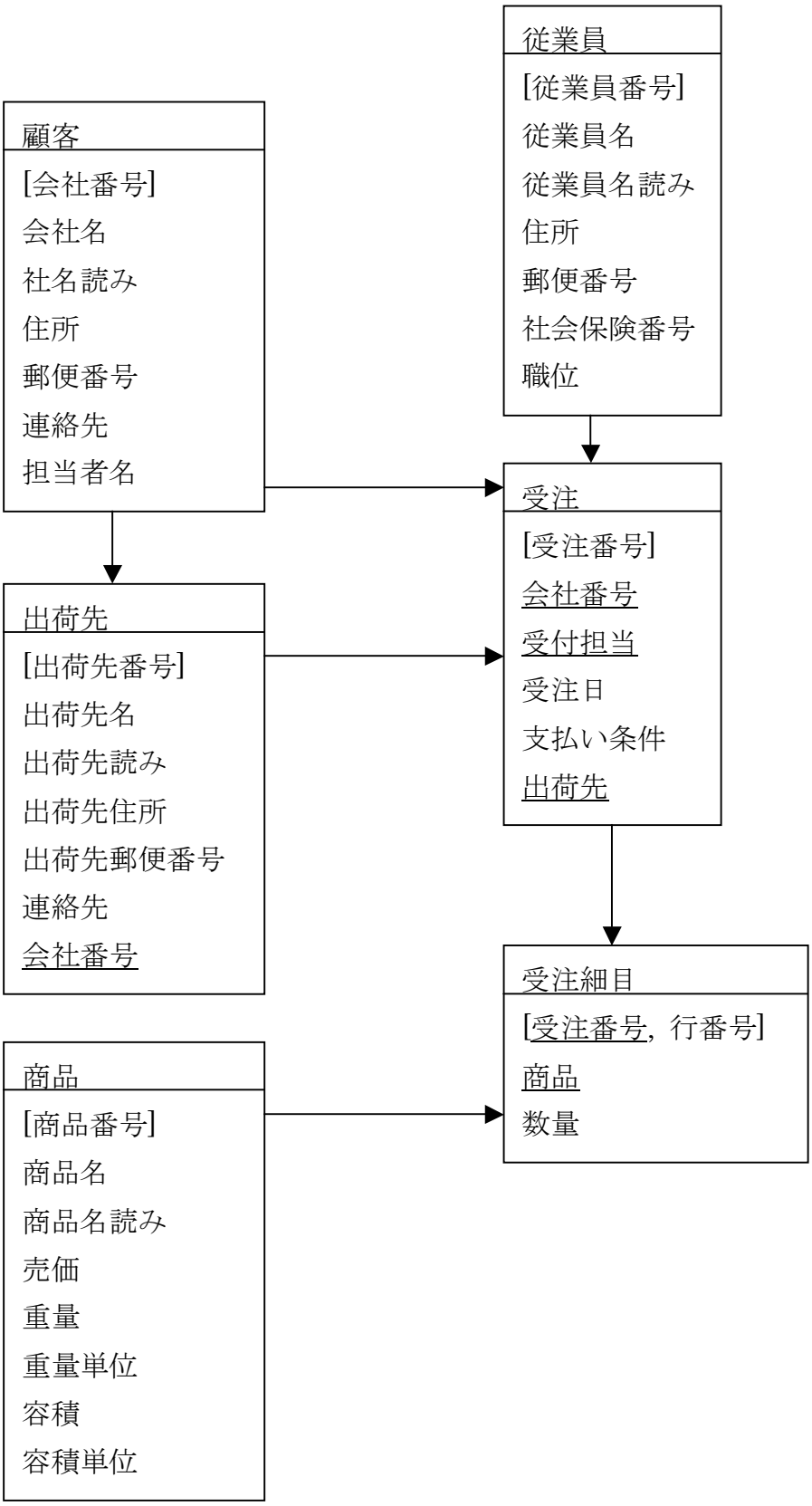


図 3-10(a)　営業支援データベースの例

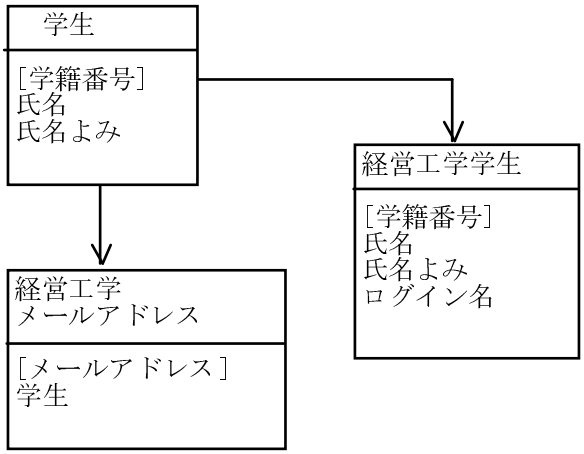


図 3-10(b)　学生表データベース（図 3-9）の例

なお、データモデルの一般的パターンがある。ビジネスで用いられる伝票には多様なものがあるが、大多数が、注文書とか納入書のような概要部分と細目部分を持っている。図 3-10 においては、受注と受注明細の組み合わせがそれである。

さて、ここまで説明された概念を使って、世の中に存在するデータのデータモデルを作成することが既にできるようになった。以下の練習問題にチャレンジすることで自己のデータモデル化能力の養成することができる。なお、物理学や電子工学などの物理に基盤を持つ工学では

現象や対象→よく定義されたモデル→分析や最適化

ということを行う。たとえば、

ふりこ振動→微分方程式で近似→周期を計算→だいたい合っていることを確認；

という感じである。現象や対象のモデルを作るときに思考を進める方法は、データのモデルを作る場合にもあてはまる。定義された概念をよりどころとして、いろいろな可能性の中から満たすべき条件を確認しながら、つまりいくつかのモデルを試しに作りながら、現実からの要請と対応させて正解のひとつを探しだす。言いかえれば、気楽にモデルを描き始めては練り上げていくつもりでよい。

問 3-1

図 3-11 のデータは，図書館の蔵書検索システムの出力例である（筑波大学で以前に使っていたものである）。

（１）一般に市販されているすべての本に必ず書かれている ISBN とは何か。ウエブなので調べてみなさい。また、NCID も同様に調べてみよう。

（２）図書館には多くの種類の本がある。図で所蔵の部分がないものとして無視し，多くの本の書誌のデータがもともとはどういう表に入っていたかをモデル化しなさ

い。つまり，属性，ドメイン，管理実体型と主キー，参照関係，汎化関係を発見してメタデータの図にしなさい。

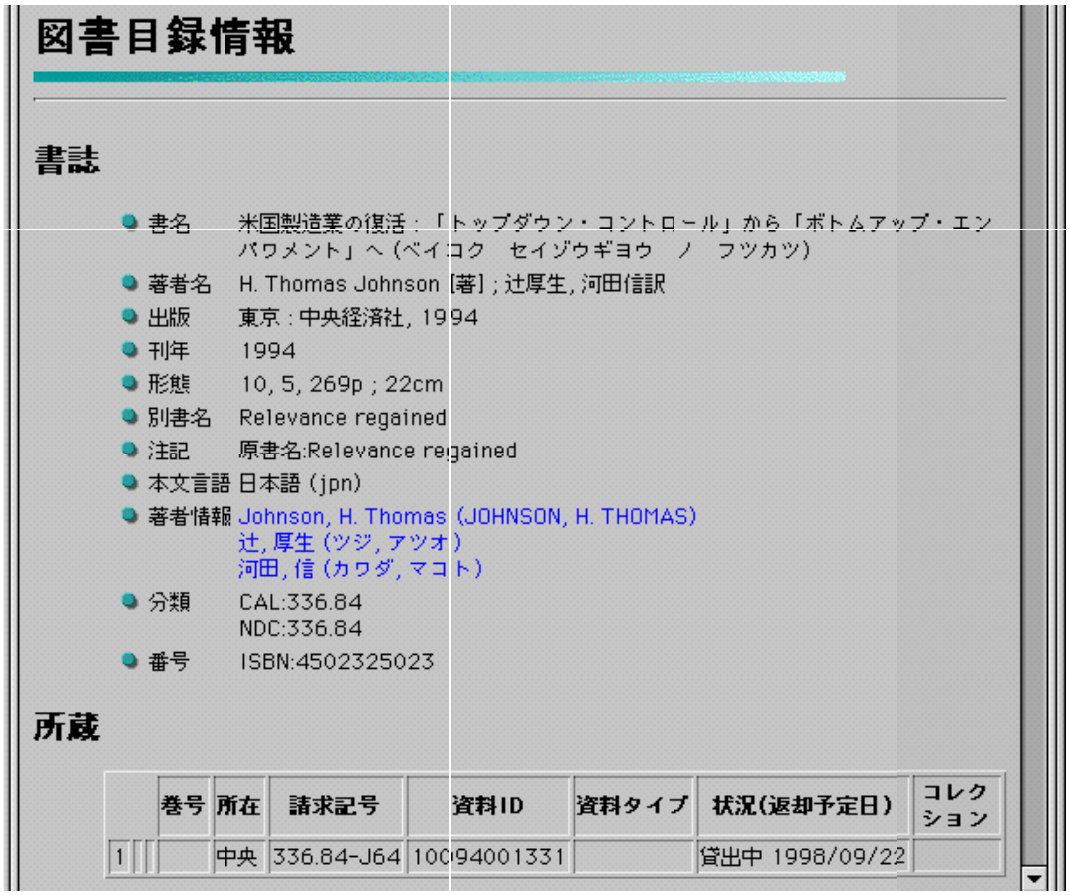


図 13-11　図書検索システムの出力例

（３）大学の図書館には同じ本でも数冊あることが多い。よい教科書は特にその傾向がある。図書館としてはモノとしての１冊１冊の本を管理していく必要がある。次の図 13-12 は同じ本を 3 冊保有している場合の検索出力例である。蔵書数の部分も含めたデータを保持するためのデータモデルを作成しなさい。

［書　　名］経営情報システム
［著 者 名］島田達巳，高原康彦著
［出 版 社］東京 : 日科技連出版社，1993
［形　　態］xvi，297p ; 22cm
［シリーズ名］シリーズ・経営情報システム:第1巻
［分　　類］336.17
［著者情報］島田，達巳(シマダ，タツミ) ; 高原，康彦(タカハラ，ヤスヒ
［ＩＳＢＮ］4817161574
［ＮＣＩＤ］BN08922568
［蔵 書 数］3冊

(所在)	(請求記号)	(資料ID)	(状況)
1: 中央	336.17-Sh36	10094002332	教員特別貸出中
2: 中央	336.17-Sh36	10093004493	
3: 大塚	336.17-Sh36	10093280367	

図 13-12　同じ本が 3 冊ある場合の出力例

データモデルのチェック！

既に３冊所有しているとき，同じ本をもう 2 冊購入したとする。このとき，あなたのデータモデルでは，1 冊づつを区別してきちんと新たな 2 冊分のデータを加えることができますか？

問 3-2　伝票のデータモデル

図 13-13 は，リーンサプライ（株）からの納品時に受け取ったある納品書伝票である。小口の取引でよく見かける伝票である。同様のものは手書き用伝票として文房具売り場で売られている。この伝票の元となっているデータが蓄積される表がある。さらにそのデータの元となる現実世界での納品という行為があり、納品を認識するための概念を人間が持っている（文化や言語や歴史を超越してなお、納品という概念がある）。この納品関連の概念をデータモデルを作成し図で表しなさい。言いかえれば、納品書伝票を作成する元となっているデータを蓄積するためのいくつかの表を設計しなさい。

（ビジネスとして現実世界で受注に対応して納品されるので、つまり納品伝票が発生するので、図 3-10(a)の受注を納品と読み替えたようなデータモデルになるはずである。また、納品は請求のために使われるが請求についてはこの問では無視してよい。）

データモデルのチェックとしては、新たに別の顧客への納品があったときに、それを我が社の（リーンサプライ社の）データモデルのデータとしてきちんと記録できるかどうかを調べる。主キーがふれるか、参照関係はくずれないか、データから正確に納品書を再現できるか。

リーンサプライ株式会社
埼玉県加須市

納品書

納品書No.　3233289

お客様名　筑波大学
住所　茨城県つくば市天王台１ー１ー１
お届け先　社会システム工学事務室
ご担当者　佐藤様
TEL　029-yyy-kkkk
お客様ご注文番号

	品名	商品コード	納品数	単価	金額
1	再生ペーパー白WA4	2300750	2	2,250	4,500
2	SuperSoftware J	J-4531	1	18,000	18,000
3					
4					
5					
6					
7					
8					
9					
10					

納品に関するお問い合わせは 0120-BBBB-CCCC までお願いします。

図 13-13　納品書

問 3-3　階層図のデータモデル

図 13-14 の組織階層図がある。これを２つの管理実体型（表）で表したい。一つは各組織単位の一覧を表し，もう一つはそれらの組織単位の上下結合関係（親部署と子部署）を表すものである。 組織階層図のデータモデル図を完成させよ。

自分のモデルが階層図を正しく表しているかどうかは，表のデータを使ってこの図を再現できるかどうかを考えるとよい。

（注意：図を表すというのは、ここでは図として表現された階層関係・上下関係・全体部分関係の認識を表現することであり、図形の大きさなどについては無関心である。CAD では図形情報に関心がある。）

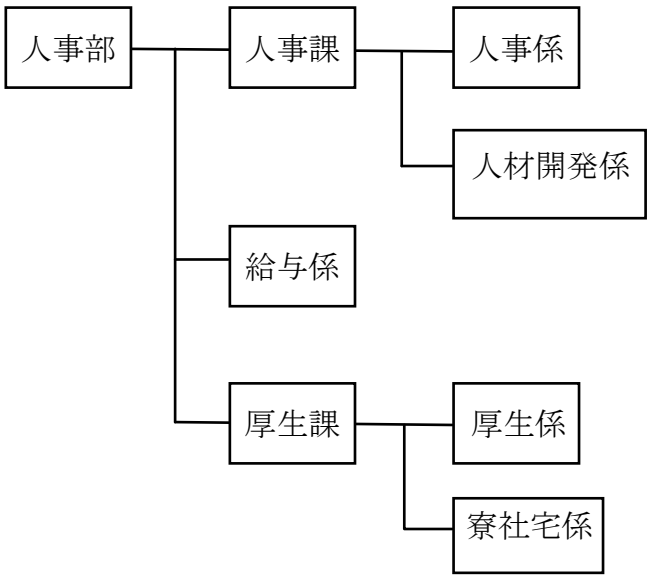


図 13-14　組織階層

問 3-4

市販のデータベース管理ソフトウエア(DBMS)は、無料で使える試供版や統合的ソフトウエアの機能の一部として使えるものがある。たとえば Access、Oracle、DB2、MySQL、PostgreSQL などがある。いずれかを用いて、これまでの問の解のデータモデルを DBMS のテーブルとして実現してみよう。その際には属性ごとにデータ形を適当に設定する。たとえば学生番号の属性は 10 桁の数値（numeric）にするとか、氏名は文字列形にするなどである。

問 3-5　データモデリング武者修行

この問は少しチャレンジングである。自分の身の回りにある請求書や納品書や、宅配の伝票を見て、それらのデータを格納するためのデータモデルを作成せよ。また、何かを集計したり計画データを記載しているドキュメントについても同様に考えてみよ。データモデルを作成することは（メタ）データ設計とも呼ばれる。伝票のデータ項目について、ビジネス上の必要とか利用方法を考えながらモデリングする。

さらに進めて、人類の歴史上まだ文書化されてはいないが、社会システムを動かすためとかビジネスに使えそうなことがらを見つけ、それを蓄積・記録するためのデータモデルを作成しなさい。

3.4　データモデルの正規形（normal form）

データモデルはデータベース管理システム（DBMS）というソフトウエアによって実現でき、管理実体型はテーブル（表）になるのであった。本節で、管理実体型という概念化をほとんど行うことなく、テーブルの設計図に焦点を当てて、テーブル定義の形式的な完成度を追求する方法として、正規形の理論を説明する。

経験的には、すべての管理実体型が主キーを持つべしという主キー制約を考えるTH や DAE の概念枠組みを用いてデータモデルを設計すると、完成したデータモデルはおのずと正規形になっている。つまり、実現される世界であるテーブルの正規形の議論は、概念の世界であるDAEデータモデルにおいて解決されるといえる。しかし、このことは数学的に証明されたわけではない。そのためにも、また、完成度の低いテーブル群の定義を設計しなおす場合もありうるから、DAE と正規形の関連を説明する。

テーブル正規形は、歴史的には関係データモデルというデータモデルの発展過程で提案された概念である。TH や DAE と異なり、関係データモデルの表（テーブルとも呼ぶ）とは属性集合の直積の部分集合であって、主キー属性を持つこと要求されない。つまり、以下の正規形の説明での表は主キー属性を持たなくてもよい。

3.4.1 第 1 正規形

表または表の属性が第 1 正規形であるとは、行が定まって、かつ、どの属性の値も普通の集合であること。

属性の値が普通の集合でない場合の例を上げると、図 3-15、図 3-16 である。 どちらも第 1 正規形でない。図 3-15 では「趣味群」のドメインが集合値になっているのでおかしい。また主キーとなる属性がない。図 3-16 の表では（顧客番号，顧客名）という組みと、残りの属性の値が図のように対応しているので、この表からは「行」をうまく取り出せない。

人の趣味	
人	趣味群
田中	碁，将棋，テニス
大田	碁，水泳
木村	テニス，水泳

図 3-15　人と趣味の関連　[穂鷹 1989]

受注控え				
顧客番号	顧客名	品目番号	数量	品名
1150	川田商店	I025	50	サンドペーパ 25 番
		P321	23	P 社油性ペイント赤 1L
		S153	15	S 社水性ペイント黄 3L
2030	青木工業	P321	35	P 社油性ペイント赤 1L
		S166	23	S 社水性ペイント緑 6L
3040	トミカワ	N216	155	N 社殺虫剤小

図 3-16　受注の控えメモ　[島田・高原(1993)の例を変更]

問 3-6

では、どうすれば第 1 正規形になるか？

図 3-15 を第 1 正規形にする：図 3-15 は主キーがないので、新たに「人の趣味＃」属性を主キーとする。図 3-17 のようになる。

人の趣味		
人の趣味＃	人	趣味
1	田中	碁
2	田中	将棋
3	田中	テニス
4	大田	碁
5	大田	水泳
6	木村	テニス
7	木村	水泳

図 3-17　（図 3-15 を）第 1 正規形に直した表

図 3-16 を第 1 正規形にすることを考える。主キーは、誰がどの品目を注文したかということを表すための属性の組 [顧客番号，品目番号] として，単純にそれらが各行で明確になるように作り直せばよい。すると図 3-18 を得る。

受注控え				
顧客番号	顧客名	品目番号	数量	品名
1150	川田商店	I025	50	サンドペーパ 25 番
1150	川田商店	P321	23	P 社油性ペイント赤 1L
1150	川田商店	S153	15	S 社水性ペイント黄 3L
2030	青木工業	P321	35	P 社油性ペイント赤 1L
2030	青木工業	S166	23	S 社水性ペイント緑 6L
3040	トミカワ	N216	155	N 社殺虫剤小

図 3-18（図 3-16 を）第 1 正規形に直した表

3.4.2 第 3 正規形

図 3-15, 3-16 の表は複雑であって都合の悪い性質をもっている。たとえば図 3-15 では田中さんが将棋を趣味にしていないときには、もはや「将棋」という趣味が存在することが表現されない。また、図 3-16 では、トミカワさんからの注文を消してしまうと、顧客番号 3040 のお客の名称が分からなくなる。また、同時に、トミカワさんが（可能な）顧客であることもその顧客番号も分からなくなる。

だから、こういうことがないような、スッキリした構成で、おなじ事実を記述することが考えられた。それが（関係データベースという分野の）正規形という概念である。第 1 正規形だけでは、表としての操作はできるようになるが、上の不都合は直っていない。

次の、関数従属という概念で、その不都合をなくすることを考える。

関数従属

表において、ある属性 X の値が決まると、対応する別の属性 A の値が一意にきまるとき、A は X に関数従属であるという。

X が 1 個の属性ではなく、いくつかの属性の組みである場合も関数従属という。

主キーが 2 つ以上の属性の組みから成るときには、第 1 正規形と第 3 正規形の間に、次の第 2 正規形を考えることができる。したがって、主キーが複合属性のときも考えうる。しかし、第 2 正規形は第 3 正規形への整理作業の中間形態にすぎない。

第 2 正規形

第 1 正規形であるような表において、K を任意の主キー候補であって属性の組み合わせか複合属性とする。K が次を満たすとき、表属性は第 2 正規形である：

K に属さない任意の属性 A が、K の真部分集合に関数従属でない。

図 3-17 は、主キーがただ一つの属性だけからなるので、第 1 正規形であると同時に第 2 正規形でもある。一方、図 3-18 の表は、第 1 正規形であるが、顧客番号によって顧客名が唯一に決まり、また品目番号によって品目名が唯一に定まるので、第 2 正規形でない例である。

問 3-7

図 3-18 の表を第 2 正規形にするとどうなるか。その結果、いくつの表に分かれるか。

第 3 正規形

第 1 正規形であるような表が、さらに次のこと満たすこと：

> 表の任意の属性の組みを X とするとき、X に含まれない属性のうちで X に関数従属になるようなものが存在するとき、他の残りのすべての属性も X に関数従属であること。

言い換えると、「主キー以外の属性の中で、他の属性を関数従属的に定める属性が存在しないこと」である。第 1 正規形から、この性質を満たすように、表を分割して行くと、たいていはいくつかの第 3 正規形の表で、参照関係をもつものに分けられる。

次の図 3-19 のデータモデルであらわされる表は、第 2 正規形であるが、第 3 正規形ではない表の例である。ただし属性のドメインは省略した。

宿泊予約
[顧客番号]
顧客名
顧客住所
ホテル番号
ホテル名
ホテル住所
ホテル部屋数
宿泊開始日
宿泊日数

図 3-19　宿泊予約 (カッツ：情報システムの分析と設計)

問 3-8

図 3-19 の表をどうすれば第 3 正規形になるか？

問 3-9

実は図 3-19 の表は何を記述したいのかよくわからないところがあるので、図 3-19 のデータモデルを批判してみよ。つまり、何の側面を考えるのを怠ったために、こういう変なモデルを最初に考えてしまったかを述べてみましょう。そのために、この図の基になったであろう伝票のイメージを書いてみよう。あるいは、これらの表が表わそうとしていることを翻って考えて、何か変なところやおかしいと思われる点を述べてみよう。そして、図 3-19 に替わるデータモデルを提示してみなさい。

3.4.3 データモデル化の方法論

第 3 正規形までくると、指摘したような不都合はすべてなくなっているかというとそうではない。たとえば、前出の図 3-17「人の趣味」のデータモデルを考える。

ここで、田中さんが 3 人も 4 人もいる場合には、図 3-17 は正確な記述を与えない。また、第 2 行目のレコードを削除すると、趣味として将棋がありうることがこの表からはわからなくなる。同時にまた、どれだけの趣味があるのかもわからないようなデータモデルになっている。

そこで、もし、2 人以上の田中さんを区別するために、人間に番号を振って識別することにしたとする。これが図 3-20 である。こんどは第 2 正規形ではなくなっている。

人の趣味			
人の趣味＃	人＃	人名	趣味
1	1	田中	碁
2	1	田中	将棋
3	2	田中	テニス
4	3	大田	碁
5	3	大田	水泳
6	4	木村	テニス
7	5	木村	水泳

図 3-20. 同姓同名を識別できるようにした「人の趣味」表

結局は、図 3-21 のように、趣味についても人名と同じようにあつかって，合計で 3 つの表で事態を表現すれば，問題は解決する。結局突き詰めて考えて，第 3 正規形まで持っていけば，かなり満足できるデータモデルを作れる。

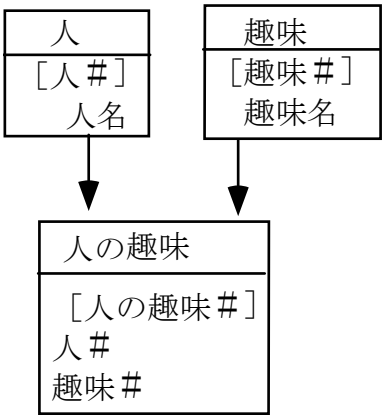


図 3-21　人の趣味のデータモデル

では，どうやって，仕事に必要ないろいろなデータから，第３正規形という整理されたテーブル設計にもっていけばいいのか。

方法は２つある。

（方法１）とりあえず，表のような記述を行い，それから，第１正規形，第２正規形，第３正規形の順に，機械的な手順を踏んで正規化する。現実に使われるデータモデルでは、情報システムの規模によるが500個以上から5,000個程度の管理実体型を定めることになるので、このような手順でデータモデル化することは絶望的である。このやり方で多少の大きさ・複雑さのデータのモデル化をしようとしたら、生きている間に終わらないほど非効率である。

（方法２）「データモデルとは，現実に発生するデータを紛れなくスッキリと記述する記述法・概念の集りであるという」データモデルの本来の問題意識にたちかえると，そもそも表として記述したいことは，商品とか，受注のような，仕事上で共同作業を可能にするために管理する必要がある実体である。そのために管理実体型と呼ばれる。仕事を進めるうえで管理すべきモノとコトが何であるかを見定めながら，

- 新規に商品（または受注）を記入するときに必要な属性はなにか，
- ある商品（または受注）の記録を削除しても不自然に情報は消えて他に管理したいデータが消えることはないか，

について考える。いわば，その表のデータに関連する書き込みや削除のイベントが発生したときに，関連を持つ表の集合全体として，ビジネスを実行するための必要性から考えて、不必要にデータが変化しないかをチェックして，表と属性を設計していく。そうして，第３正規形かどうかをチェックし，そうでなければ、DAE の概念に立ち返って検討する。つまり、何を管理すべきなのか、現実世界でおこる事柄

ごとに対応して管理実体型を分けているか、個別に実体を区別しかつビジネスに必要十分になる記述の細かさを定義するような属性はなにか、ということであす。こうした観点から理由を見つけて，たいていはいくつかの属性をまとめてひとつの表にし，元の表との間には参照関係ができる。

以上の方法２は、DAE データモデリング機能の概念を使って管理実体型としてデータモデルを作り、その実現としてテーブル設計を得ることに他ならない。

データモデルの「正解」はただ１つではない場合が多い。ビジネスの都合や利用可能技術によって、一つの管理実体型についてどの細かさまで記述すべきかということが異なるからである。属性概念の説明で述べたように、データモデルの細かいところは異なったモデルができることが有りうる。

自分のテーブル設計が「正しいか」どうかのチェックポイント：
（１）主キーを持つか。
（２）表にデータを書き込むイベントが発生したとき，1 行（1 レコード）の全属性に対するデータが定まるか。
（３）第３正規形になっているか。

経験則：上の（１）と（２）を満たすように表が作られていると、第３正規形になっている。また、データモデルの本によっては多対多関係を許すかのような記述を見ることがあれが◀、それは間違いであると言ってよい。多対多関係は、もし必要ならば、主キーと R キーを使ったいくつかの１対多の関係に分割する。

3.5　TH と ERM と UML の関係

データモデルとしては Chen（陳）氏の ER データモデリング機能が世界的によく知られており解説書も数多い。TH や DAE と ER との違いは、表面的には、ER が実体型（entity type）以外にも関係（relationship）を使うことである。Chen 氏のオリジナルな論文を読むと、２つの実体型をつかって関係を作るときには、各々の実体型の主キーを使っている。そのように作られた関係を新たに実体型と見なすと、その２つの主キーの組を関係実体型の実体型の主キーとすることができ、DAE との本質的なちがいはほとんどなくなる。

ER を使用する際に、そのような主キー制約をつけて、かつ関係を実体型として主キー制約を要求することは有効である。たとえば、統合基幹情報システム(ERP - enterprise resopurce planning ソフトウエアパッケージ）である SAP R/3 のメタデータ

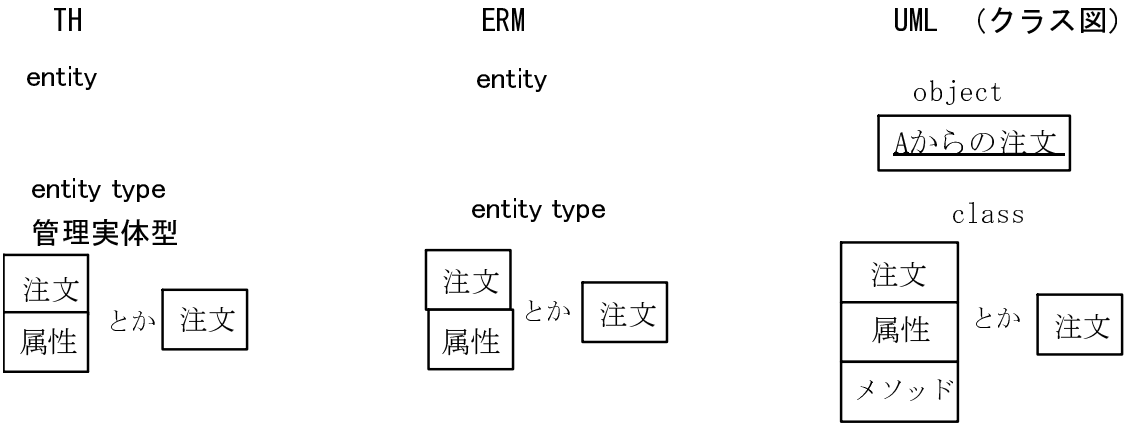
の表示に使われている。また、渡辺幸三著「業務別データベース設計のためのデータモデリング入門（2001）」においても、ER モデルの使用方法として、関係を構成するときに主キーを用い、かつ関係自体が必ず主キーを持つようにモデル化している。

こうして、データの世界について本質的な構造を取り出すと、主キーを持つ管理実体型を利用することになる。

UML は、GUI（グラフィカル・ユーザ・インターフェイス）の作成を伴うオブジェクト指向ソフトウエア開発のためのソフトウエアのモデル化の概念と図的ツールの集まりである。UML ではソフトウエアを扱う。一般にアルゴリズムを実現するものがプログラムやソフトウエアである。プログラムはデータ構造とその構造を持つ変数を一連の計算で変換するアルゴリズムから成るので、UML もデータ構造をモデル化する図的ツールを備えている。クラス図がそれである。

以下の図 3-22 には、TH(や DAE)と ER と UML のクラス図の相互的対応を直感的に示してある。

UML ではクラスは大きく 2 種類ある。データモデルが対象とする管理実体型を記述するクラスと、ユーザインターフェイスと関連する変数データ（オブジェクト）を記述するクラスである。TH や ER と対応するのは前者のクラスである。



THでは、業務上管理する必要のあるものの集合名を、管理実体型の名前とする。ERM でもUML でも、モノや事象を何でもentity type やobject やclassにするという。UML はソフトウエアの分析と設計に使いたいので、特には主キーを要求しない。

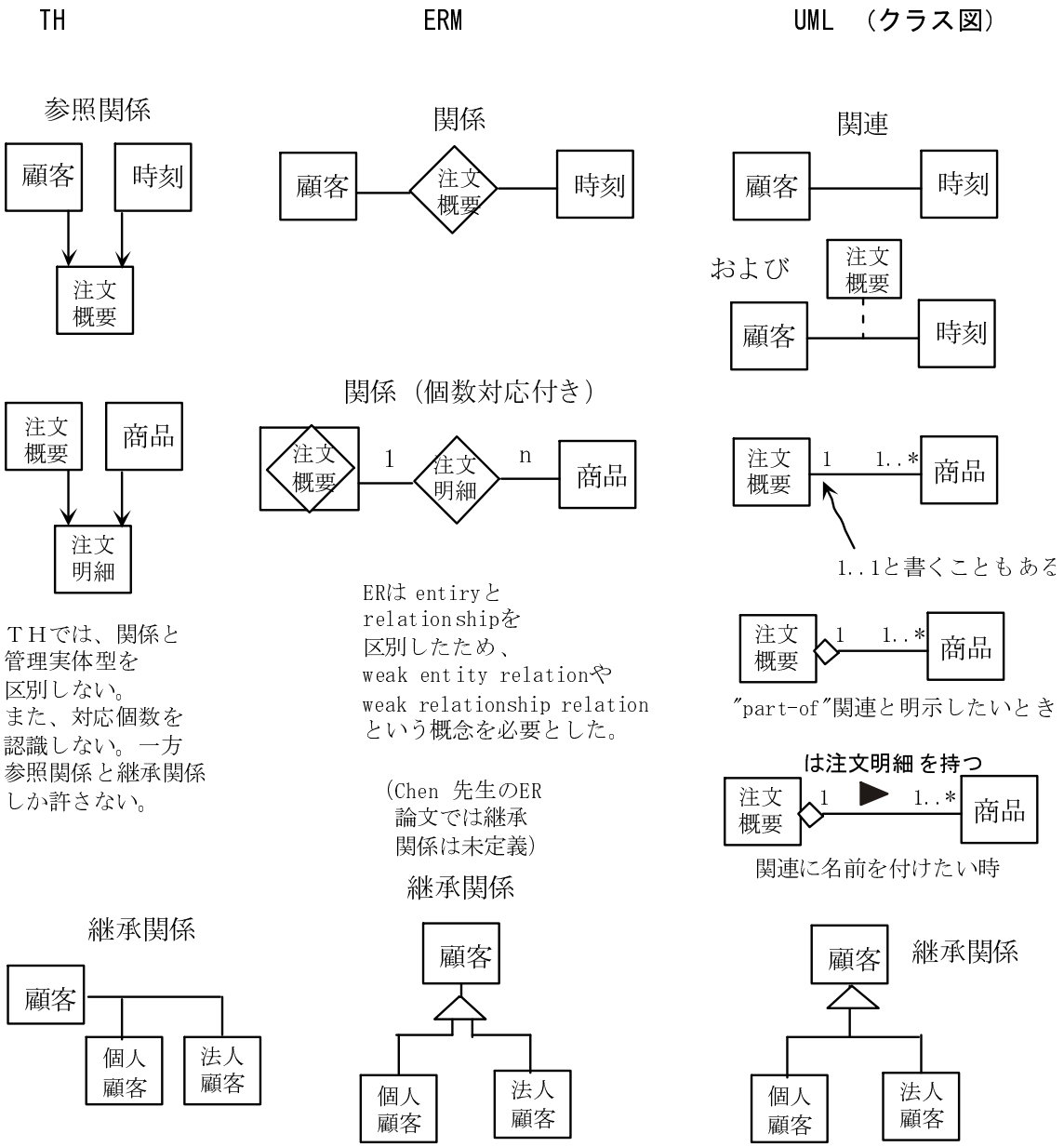


図 3-22　TH や DAE、ER、UML のクラス図の対応関係

3.6 データモデルの定式化

DAE データモデリング機能は TH データモデリング機能を変形し簡略化したものであった。

データモデルを語るときに、表やテーブルとしてファイルを例示することだけでは、結局概念規定のあまさから議論が混乱する場合がある。そこで本節では穂鷹著の TH データモデルの定式化を参考にして、DAE データモデリング機能の概念を集合論的に定式化する。

以下に、集合論的定式化を示しておく。

DAE データモデリング機能は，いくつかの集合，それらの間の関数，それらが満たすべき２種類の制約からなる。順に説明する。

１）DAE で用いる集合：

ES：実体（エンティティ）を要素とする集合で，実体（の）集合と呼ぶ。

ETS：実体型(entity type) を要素とする集合で，実体型集合と呼ぶ。

DS：ドメインを要素とする集合で，ドメイン集合と呼ぶ。

KS：キー実体型(key entity type)を要素とする集合で，キー実体型集合と呼ぶ。キー実体型は管理実体型とも呼ばれる。KSはDSの部分集合。

FTS：ファイル型を要素とする集合で，ファイル型集合と呼ぶ。FTSはETSの部分集合。

AS：属性を要素とする集合で，属性集合と呼ぶ。

LES：リスト実体を要素とする集合で，リスト実体集合と呼ぶ。

２）DAE で用いる関数

$S : DS \to \mathrm{P}(ES)$

ここで$\mathrm{P}(ES)$はESの全ての部分集合を要素とする集合でべき集と呼ばれる。

$FT : KS \to FTS$ ；ファイル型関数と呼ぶ。全射的とする。KSの要素xに対し，$FT(x)$をxのファイル型と呼ぶ。

$A : FTS \to \mathrm{P}(AS)$ ；属性関数と呼ぶ。$A(FT(x)) = \{a_1, \cdots, a_n\}$とするとき、各$a_i$を$FT(x)$の属性と呼ぶ。

$dom : AS \to DS$ ；ドメイン関数と呼ぶ。$dom(a)$をaのドメインと呼ぶ。

$File = \{f : FTS \to LES \mid f$はファイル内容関数$\}$。ここで$f$ がファイル内容関数であるとは、任意のKSの要素xに対して、$A(FT(x)) = \{a_1, \cdots, a_n\}$とするとき $f(FT(x))$ が $f(FT(x)) \subseteq S(dom(a_1)) \times \cdots \times S(dom(a_n))$ を満たすことである。$f(FT(x))$は実質的にある時点における$FT(x)$の中身を表現するものである。

$V_f : f(FTS) \times AS \to S(DS)$ という関数を各ファイル内容関数 f に対して次のように定義する。$A(FT(x)) = \{a_1, \cdots, a_n\}$とする。$f(FT(x))$の任意の要素 y と$A(FT(x))$の任意の要素$a_i$に対して、$V_f(y, a_i)$は$S(dom(a_i))$の要素であること。この関数$V_f$ はfが文脈から明らかなときには、単にV とかく。

なお、以下の条件が成立しているものとする；

$ETS = DS \cup FTS$（直和とする。つまり、$DS \cap FTS$は空集合。）

$ES = ETS \cup LES \cup AS \cup S(DS)$

LES=∪ ｛P($S(dom(a_1))\times\cdots\times S(dom(a_n))$)｜ KSのある要素 xについて $A(FT(x))=\{a_1,\cdots,a_n\}$｝。

以上の集合と関数の関係を図示すると、図 3-23 のようになる。

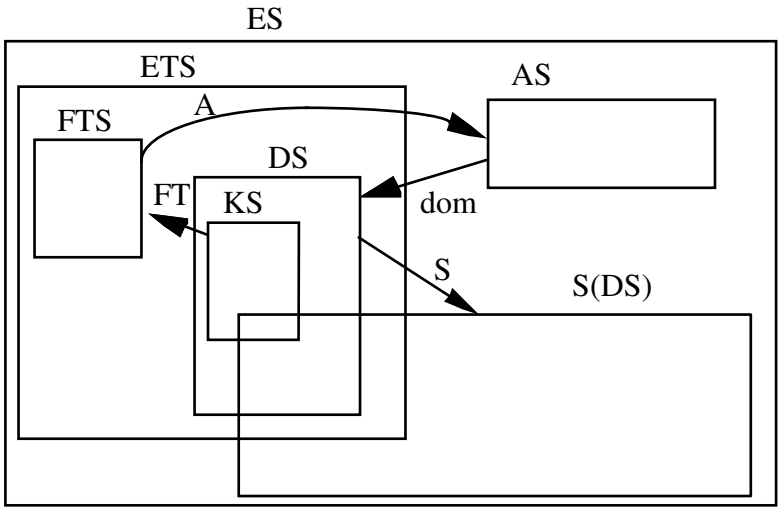


図 3-23.　The DAE データモデル)

３）　２種類の一貫性保持条件

定義：主キー(primary key)

$key:FTS\rightarrow AS$は、次の２条件を満たす関数である。

1）　$dom(key(FT(x)))=x$を満足する、

2）ファイル型$y=FT(x)$に対してキー(または主キー)を与える関数である。

ここで$key(y)$がファイル型yのキーであるとは；

任意のファイル内容関数fと、任意の$z_1,z_2\in f(FT(x))$に対して

$[V_f(z_1,key(y))=V_f(z_2,key(y))\Rightarrow z_1=z_2]$

定義：参照キー(参照整合性)

任意の$x_1,x_2\in KS$をとる。

$a\in A(FT(x_2))$が、$FT(x_2)$の$FT(x_1)$からの参照キー(Ｒキー、または外部キー)であるとは、

1）　$dom(key(FT(x_1)))=dom(a)=x_1$　かつ、

2）任意のfに対し、

$f(FT(x_2)).a\subseteq f(FT(x_1).key(FT(x_1))$　　つまり

$V_f(f(FT(x_2)),a)\subseteq V_f(f(FT(x_1)),key(FT(x_1)))$

このとき　$FT(x_2)\leftarrow FT(x_1)$とか$FT(x_1)\rightarrow FT(x_2)$とかく。

参照キーは、データベースのファイル間の整合性条件としての参照整合性を表現す

るのに使われる。

定義　参照関係

$REFER \subseteq AS \times AS$

$(a,b) \in REFER$とは、ある$x_1, x_2 \in KS$があって次の２条件を満たすこと：

1）$a \in A(FT(x_2))$ かつ$b = key(FT(x_1))$

2）aが$FT(x_2)$の$FT(x_1)$へのＲキーである。

定義　継承(inheritance)整合性

$x_1, x_2 \in KS$をとる。

$FT(x_1)$は$FT(x_2)$の部分ファイル型、あるいは$FT(x_2)$は$FT(x_1)$の汎ファイル型、あるいは、$FT(x_1)$は$FT(x_2)$の属性を継承するとは、

1）任意の$f \in File$に対して、

$V_f(f(FT(x_1)), key(FT(x_1))) \subseteq V_f(f(FT(x_2)), key(FT(x_2)))$；

ファイル内容$f(FT(x_1))$の任意の組yに対して、yのキー値と等しいキー値を持つ$f(FT(x_2))$の組zがある、かつ、

2）$A(FT(x_1)) - \{key(FT(x_1)\} \subseteq A(FT(x_2)) - \{key(FT(x_2)\}$；

$FT(x_1)$は$FT(x_2)$のキー以外の属性を含む、つまり、$FT(x_1)$はキー以外の属性を継承する、かつ、

3）任意の$a \in A(FT(x_2)) - \{key(FT(x_2))\}$，$y_1 \in f(FT(x_1))$，$y_2 \in f(FT(x_2))$に対して、

$V_f(y_1, key(FT(x_1))) \subseteq V_f(y_2, key(FT(x_2)))$ならば$V_f(y_1, a) \subseteq V_f(y_2, a)$；

$FT(x_2)$のキー値以外の継承属性値は一致する。

上のとき、$FT(x_1) \leftarrow\!\!\bullet\!\!- FT(x_2)$，$FT(x_1)$ inherits $FT(x_2)$とかく。

定義　継承関係

$INHRT \subseteq FTS \times FTS$を　z_1 inherits z_2で定める。

命題

任意の$x_1, x_2 \in KS$に対して

$FT(x_1) \leftarrow\!\!\bullet\!\!-\ FT(x_2)$ ならば、$FT(x_1) \longleftarrow FT(x_2)$。

$x_1, x_2 \in DS$をとる。x_1がx_2の部分型(あるいはx_2はx_1の汎型)であるとは、

$S(x_1) \subseteq S(x_2)$であること。

上のとき、x_1 isa x_2　とか、$x_1 \leftarrow\!\!\bullet\!\!-\ x_2$　とかく。

x_1, x_2がFTSの要素の時は、x_1 ←● x_2を$FT(x_1)$ ←● $FT(x_2)$で定める。

$ISA \subseteq DS \times DS$

$(x_1, x_2) \in ISA$ を x_1 isa x_2 で定める。

定義　排他的専化

y_1, y_2がy の互いに排他的な専化であるとは

1) y_1 ←● y ，　2) y_2 ←● y ，　3) $f(y_1) \cap f(y_2) = \phi$

問 3-10

EDI（電子データ流通）では、手書き伝票に代わるものとして電子的にデータをやり取りしている。インターネットが出現する以前から実用されていた。電子化とは独立に、まず、ビジネスを実行するために伝票が必要であり、それが標準伝票として確立して業界全体の効率向上に役立ってきた。標準伝票が存在すれば、EDI を実施するには原理的には標準伝票ごとにそのデータ項目のデータモデルを作ることで、データモデルの実現方法が紙なのかデジタルデータなのかの違いだけで、ビジネス上の意味は同じである。

チェーンストアや建設業界や電子機器販売業界など、EDI のデータの標準はみんなが尊重し従わないと、互いに利用メリットがないため、インターネットなどで公開されている。どれかの業界の EDI のデータの標準の文書をダウンロードし、多くの伝票の中から意味の分かりそうな伝票の規約を取り上げて、データモデル化してみよ。

（ネットワーク化、デジタル化がこれからも広がっていくので、データモデルの知識やデータモデリング能力はますます常識として要求されるようになる。）

4 章　結合離散事象システムと業務取引システム

4.1　結合離散事象システム（マルチコンポーネント DEVS）

ビジネスプロセスの多くの機能とプロセス内で大量に用いられるデータが多様に結合してできあがる現実のビジネスプロセスは、全体としてどういったシステムになるだろうか。

本章で示す答えは、ビジネスプロセス全体がひとつの離散事象システムとなるというものである。第 2 章において離散事象システムは DEVS によって表現できることがわかった。ビジネスプロセスは多くの離散事象システムが相互に関連し、全体として機能する。したがって、ビジネスプロセスを合成する立場からいえば、目標とする全体特性を持つようにするには、部品となる離散事象システム用意し、ビジネス環境下で目標を達成するようにどのように組み合わせればよいかということになる。

結合方法は基本的には activity interaction diagram (AID) による方法である。AID の中のひとつの活動がひとつの離散事象システムすなわち DEVS であり、２つの DEVS を結合する場合はかならずひとつの変数を間に置く。DEVS を結合する変数は、待ち行列やファイルに対応する。

本章では、ビジネスプロセスが、作業やワークセンターや部署や工場や企業といった大小の離散事象システムの結合として構成されていること、その中で情報システムとデータがどのような機能を持っているかが明らかになる。

4.2　結合離散事象システムの構成規則

AID の構造において､活動のいくつかが(または全部が)DEVS になっていて、さらに DEVS を開始することに関する条件を課したものが、結合離散事象システムである。つまり､AID モデルの作成方法として以下の規則３が追加される。

規則１：

２つ以上の DEVS を結合するとき、必ず待ち行列変数を介して結合する。それはどれかの DEVS の出力であり、どれかの DEVS の入力である。

規則２：

出力だけを持ち入力のない DEVS も存在する。ジェネレータと呼ばれる。

規則３：

入力変数を持つ DEVS は、その DEVS へのすべての入力変数によって定められた条件が成立するか否かによって、その DEVS が活動を開始できるかどうかが定まる。入力変数を持つ DEVS を内部 DEVS と呼ぶ。

結合離散事象システムを DEVS 結合システムとも呼ぶ。規則２の入力変数のない DEVS はジェネレータと呼ばれ、外部入力のモデルとして、DEVS 結合システムに入れられる。規則１と規則３によって、結合された DEVS 全体の動きが決まってくる。その動きを説明するために、図 4-1 の買掛金決裁プロセスで各規則が何にあたるかを見ると：

・ひとつの構成要素をひとつの DEVS としてモデル化する。

取引先（で請求書発行)、仕入れ（を記録)、請求書照合、決裁、手形作成といった活動が、DEVS としてモデル化される。

・入力のない DEVS:　取引先、仕入れ

・各 DEVS 間の結合には、照合作業を待っている請求データと仕入れデータを表現する変数が存在している。しかし、このままでは、「請求書照合」という DEVS が活動開始するには、作業を待つ請求関連データ以外に、照合作業者が稼働可能かについての情報も必要である。また、請求書照合の作業者が複数人いるときでも、作業者の空き状況を示す変数を用意する必要がある。これらを追加したものが、図 4-2 のプロセスである。

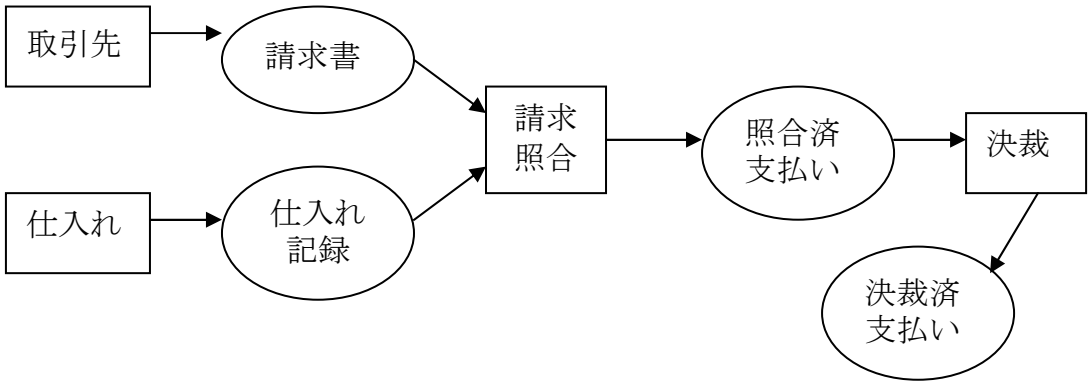


図 4-1. 買掛金決裁プロセス（作業者情報なし）

また、機械加工のようなプロセスでも、図 4-3 のように作業者の空き状況をも入力とするように追加することで、DEVS で表された活動の開始可能性を入力変数の値だけで決定できるようになる。

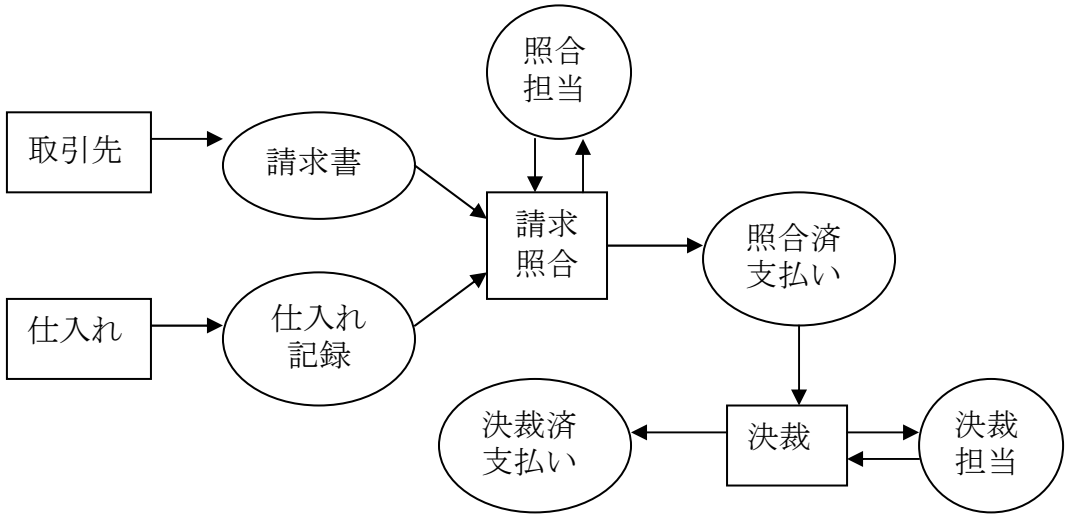


図 4-2.　買掛金決済プロセス

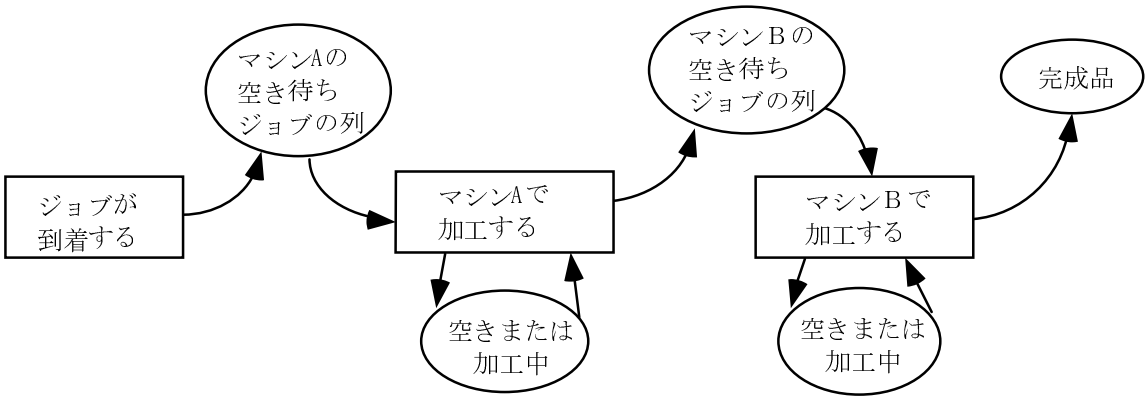


図 4-3.　機械加工工程

図 4-2 のAIDで表現された買掛金決済プロセスの遷移表は表 4-1 となる。各活動はDEVSであり、満期時間は表のtaの欄にあるとおり、取引先による請求書発行、仕入れ、請求書照合、決裁について、それぞれ 15, 7, 10, 29 である。また、第 2 章での議論との類似性をこの状態遷移表に当てはめると、買掛金決済プロセスの擬似状態変数は図 4-4 であることが予想できる。図 4-2 の結合変数を、たとえば請求書を請求書Q（Qは待ち行列(queue)の略）として表記している。図で、

たとえば、-照合-とは、照合活動が現在何も行っておらず、空であることを示す。

時刻 ta	取引先 15	請求書 Q	仕入係 7	仕入記録 Q	照合担当	照合 10	照合済請求 Q	決裁担当 Q	決裁 29	決裁済請求 Q
220	(1,210)	0	(1,217)	115	1	-照合-	10	0	(1,203)	7
224	(1,210)	0	(1,224)	116	1	-照合-	10	0	(1,203)	7
225	(1,225)	1	(1,224)	116	1	-照合-	10	0	(1,203)	7
225	(1,225)	0	(1,224)	115	0	(1,225)	10	0	(1,203)	7
231	(1,225)	0	(1,231)	116	0	(1,225)	10	0	(1,203)	7
232	(1,225)	0	(1,231)	116	0	(1,225)	10	1	-決裁-	8
232	(1,225)	0	(1,231)	116	0	(1,225)	9	0	(1,232)	8

表 4-1. 買掛金決裁プロセスの状態遷移表の一部

取引先	請求書Q	仕入係	仕入記録Q	照合担当	照合	照合済請求Q	決済担当Q	決済	決裁済請求Q
(1,210)	0	(1,217)	115	1	-照合-	10	0	(1,203)	7

図 4-4. 買掛金決裁プロセスの疑似状態

また、機械加工工程の離散事象プロセスの場合も全く同じ方法で表現することができて、製造指図であるジョブ以外に、図 4-3 の AID モデルに対応して次のようになる。図 4-5 では機械 B が 3 台ある場合を示しており、「空き」のところに空き台数を示し、加工中のジョブの欄に複数の機械の稼働状況を列として表現している。

ジョブの到着	マシン A 待ちジョブ列	マシン A で加工する	マシン A の空き	マシン B 待ちジョブ列	マシン B で加工する	マシン B の空き	完成品
(j_{n+1}, 到着時刻)	j_k, j_{k+1} … j_n	(j_{k-1},作業開始時刻),	0,	Λ	(j_{m-1}, 作業開始時刻), (j_{m-3}, 作業開始時刻),	1,	j_1 … j_{m-2}

図 4-5. 機械加工プロセスの疑似状態

買掛金決済における請求書などのデータは、顧客の連絡先や口座などの他の

データと連動している。また、仕入れ先に支払う金額は、一括して手形によって支払われたり、あるいは逆に一件の仕入れに対して何回かに分割して支払ったりもする。こうしたデータを保持する仕組みは単純な待ち行列ではないので、上記のように単に請求書を 1 件ずつカウントして待ち行列ととらえると単純すぎる。データを無駄なく一貫して保持する機構としてデータモデルがあり、結合変数として待ち行列のかわりに管理実体型を用いたファイルシステムを使用することで十分な表現力をもったビジネスプロセスモデルとなる。なお、データモデルとは、データ記述に使うことが目的だがそのために人間が注目する世界のことがらを記述するための概念枠組みであった。

また、生産管理では上の情報に加えて、作業者についてのシフトと呼ばれる稼動可能日のデータも用いられる。たとえば、朝８時３０分から午後 5 時３０分まで、１２時からの昼休み 1 時間をとって働くのはひとつのシフトである。ひとつのビジネスプロセスの中にいくつものシフトが設定されうる。図 4-2 の中の「空きまたは加工中」のデータにシフトのデータを持ち込めば、より詳しい構造を表現できる。この場合もファイルシステムを使用することでそれが可能になる。

以下では、まず、DEVS 結合システムを定義し、その後で、ファイルシステムを結合変数とする DEVS 結合システムである業務取引システムを定める。少し抽象化すれば、離散事象システムとデータモデルという２つの概念を結合したモデルを提示することになる。

4.3　DEVS 結合システムの定義

ビジネスプロセスのモデル作成や分析の精確な議論を可能にするために、まず、DEVS 結合システムの AID の形式的な定義を行う。

Q_{id}：待ち行列を構成するエンティティやオブジェクトの名前の集合。Q_{id}の各要素を管理実体型とかエンティティ・タイプと呼ぶ。

$Q = \{q : Q_{id} \to NaturalNumber\}$：各待ち行列の状況はひとつの関数で表現できる。ここで、$NaturalNumber = \{0, 1, 2, ...\}$ である。Qを待ち行列ベクトルとか待ち行列システムと呼ぶ。

活動や部署は DEVS として扱われる。つまり、それぞれが個別の DEVS である。ひとつの DEVS を表現する AID モデルの場合と同様に、活動や部署はトランザクションとも呼ばれる。トランザクションには外生的なものと内生的なものがあり、各々が待ち行列システムにアクセスする。内生トランザクションは必要な定型処理を開始させる割り当てトランザクションと、それを終了するトランザクションがある。

定義１　外生トランザクション

E 外生トランザクションを要素とする有限集合。

$f_{EK}: E \to P(Q_{id})^{+}$ 外生トランザクションが起こると、あらかじめ決められた管理実体型の待ち行列が更新される。外生トランザクションが発生したときに影響する待ち行列を表わす関数。

上記の定義で $P(Q_{id})^{+}$ は Q_{id} のべき集合 $P(Q_{id})$ から空集合を除いた集合を表わす。したがってすべての外生トランザクションは f_{EK} によって影響しうる管理実体型を必ずいくつか持つ。待ち行列の更新とは、モノの移動や追加であったり、データの記録や書き換えをモデル化している。

問 4-1

$Q_{id} = \{q_1, q_2, q_3\}$ とするとき、べき集合 $P(Q_{id})$ はどうなるか。

定義２　内生トランザクション

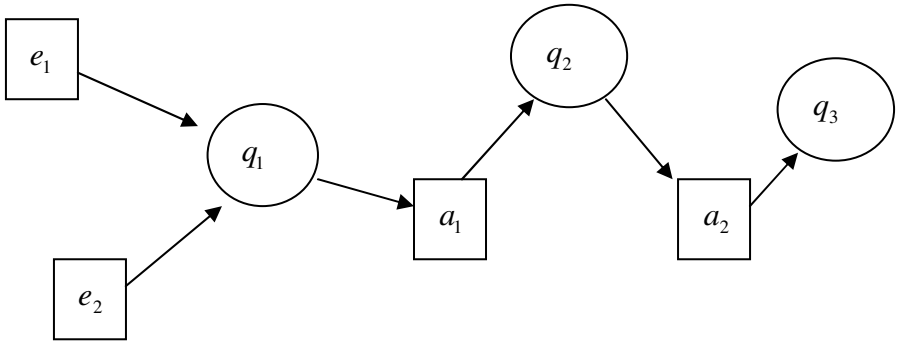
A 内生トランザクションを要素とする有限集合。

$f_{AK}: A \to P(Q_{id})^{+}$　内生トランザクションが終了したり開始したりしたとき、あらかじめ決められた待ち行列が更新される。内生トランザクションが開始や終了したときに影響する待ち行列を表わす関数。

$f_{KA}: A \to P(Q_{id})^{+}$　内生トランザクションが開始するための条件が書かれている待ち行列を指定する。

f_{AK} は活動から出て行く矢印を表現し、逆に、f_{KA} は活動に入ってくる矢印を表わす。

E, A は互いに素とする。



$f_{KA}(a_1)=\{q_1\}, f_{AK}(a_1)=\{q_2\}, f_{EK}(e_1)=f_{EK}(e_2)=\{q_1\}$

図 4-6　DEVS 結合システムの静的表現の例

問 4-2

図 4-2 において f_{AK}(請求照合)の値は何か。また、f_{KA}(請求照合)の値は何か。

定義 3　DEVS 結合システムの静的表現

待ち行列システム、外生トランザクション、内生トランザクションからなる $<Q,E,A,f_{EK},f_{AK},f_{KA}>$ という組を DEVS 結合システムの静的表現という；

DEVS 結合システムの静的表現は AID であり（図 4-6)、その AID を集合表現したものが定義３である。図 4-2 の買掛金決済プロセスの AID においては、ジェネレータからの出力となっている結合変数は、図 4-2 の「取引先」の出力にある「請求書」のように関数 $f_{EK}:E\rightarrow P(Q_{id})^+$ で表現される。たとえば f_{EK}(取引先)={請求書}である。同様に、内部 DEVS の出力となる結合変数は関数 $f_{AK}:A\rightarrow P(Q_{id})^+$ で表現される。たとえば f_{AK}(請求書照合)={照合担当、照合済請求書}である。さらに、内部 DEVS への入力となる結合変数は関数 $f_{KA}:A\rightarrow P(Q_{id})^+$ で表現する。たとえば、f_{KA}(請求書照合)={請求書、仕入れ記録、照合担当} である。結合待ち行列ベクトル Q は Q_{id} を結合変数の名前の集合とするとき、$Q=\{q:Q_{id}\rightarrow NaturalNumber\}$ であった。図 4-2 の例では、Q_{id}= {請求書, 仕入れ記録, 照合担当, 照合済請求書, 決裁担当者, 決裁済請求書}である。

次に DEVS 結合システムの動的構造の定義をしていく。DEVS 結合システムは規則２によって外部入力もジェネレータ DEVS としてモデル化されているので、全体として外部入力を持たない DEVS となる。まず、DEVS 結合システム

全体としての動きを記述する状態情報を図 4-2 の例で考えていく。

買掛金決裁のビジネスプロセス（図 4-2）のDEVS結合システムは、任意のある時点で図 4-4 のような具体的な疑似状態になっていた。この情報は、一般的には、図 4-7 のように表現される。ここで、$t_{取}$は取引先が次の請求書を発行するための活動開始時刻である。言い換えると、取引先DEVSが$g_{取}$という疑似状態を開始した時刻が$t_{取}$である。

取引先	請求書Q	仕入係	仕入記録Q	照合担当	照合	照合済請求Q	決済担当Q	決済	決裁済請求Q
$(g_{取},t_{取})$	0	$(g_{仕入},t_{仕入})$	115	1	-照合-	10	0	$(s_{決済},t_{決済})$	7

図４- 7 .買掛金決裁プロセスの疑似状態

この例では、DEVSは４つ存在する。取引先DEVS $M_A = < S_A, \delta_A, ta_A >$、仕入係DEVS $M_B = < S_B, \delta_B, ta_B >$、照合DEVS $M_C = < S_C, \delta_C, ta_C >$、決裁DEVS $M_D = < S_D, \delta_D, ta_D >$の４つであり、取引先DEVSと仕入係DEVSはジェネレータDEVSである。上では$g_{取}$はS_Aの要素、$g_{仕入}$はS_Bの要素である。また、$s_{決裁}$はS_Dの要素である。

取引先DEVSの値である$(g_{取},t_{取})$ は請求書の到着をモデル化している離散事象システムであるから、$(t_{取}+ta_A(g_{取}))$ という時刻が来たら「請求書」のイベントをおこし、すぐにまた次回の到着の予定を $g_{取}'$ にセットし開始時刻 $t_{取}$ をセットする。

図 4-2 の結合DEVSシステムでは、任意の時点で、

$(t_{取}+ta_A(g_{取}))$，$(t_{仕入}+ta_B(g_{仕入}))$，$(t_{照合}+ta_C(s_{照合}))$，$(t_{決裁}+ta_D(s_{決裁}))$

の４つからどれが早く満期になるかがわかる。これら４つの値の最小値を $t_k + ta_k(s_k)$とし、さらに、$t' = \max\{t_{取}, t_{仕入}, t_{照合}, t_{決裁}\}$とするとき、

$$ta(s) = t_k + ta_k(s_k) - t'$$

によって、sの次にイベントが起こるまでの時間を得ることができる。

これらを形式的に DEVS としてまとめると、結合システム全体 $M = < S, \delta, ta >$は次のように定義される。$E \cup A$の要素それぞれに対応する DEVS がある。Eの要素に対応するものを外部 DEVS やジェネレータ，Aのそれを内部 DEVS とよぶ。

疑似状態は$S = (S_A \times T^{\infty}) \times (S_B \times T^{\infty}) \times Q \times (T^{\infty})^2$であり、その任意のひとつの要素は$s = (g_A, t_A, g_B, t_B, q, t_C, t_D)$ である。ここで時間集合$T^{\infty} = T \cup \{\infty\}$、$t_A, t_B$はそれぞれジェネレータ M_A, M_Bの開始時間であり、t_C, t_Dは内部 DEVS M_C, M_Dの開始時

間である。ジェネレータ M_A の開始時間 t_A とは M_A が疑似状態 g_A になった時刻のことである。また、たとえば M_C について $s_C = q|_{f_{AK}(C)\cup f_{KA}(C)}$ が成立しており、s_C は結合変数 q の中で、C という内部 DEVS に関係する入力・出力結合変数だけを取りだしたものである。

t_C が内部 DEVS M_C の開始時間であるとは、M_C が疑似状態 s_C になった時刻のことである。

システム全体の時間進め関数　$ta : S \to T^{\infty}$ の定義は

$s = (g_A, t_A, g_B, t_B, q, t_C, t_D)$ とするとき、

$$ta(s) = ta_k(s_k) + t_k - t'(s)$$

ただし $ta_k(s_k) + t_k = min\{ta_A(g_A) + t_A, ta_B(g_B) + t_B, ta_C(s_C) + t_C, ta_D(s_D) + t_D\}$,

$t'(s) = \max\{t_A, ..., t_D\}$ である。

問 4-3

表 4-1 の time = 231 の行の状況を s とするとき $ta(s), t'(s)$ をそれぞれ示せ。

結合 DEVS システム 全体の状態遷移 δ は、下図のフローチャートのように遷移する。つまり結合 DEVS もひとつの DEVS となる。そして、それ自身がまた、他の DEVS の構成要素となってさらに大きな DEVS を構成できる。こうしていくらでも階層的な構成で結合 DEVS システムをモデル化できる。

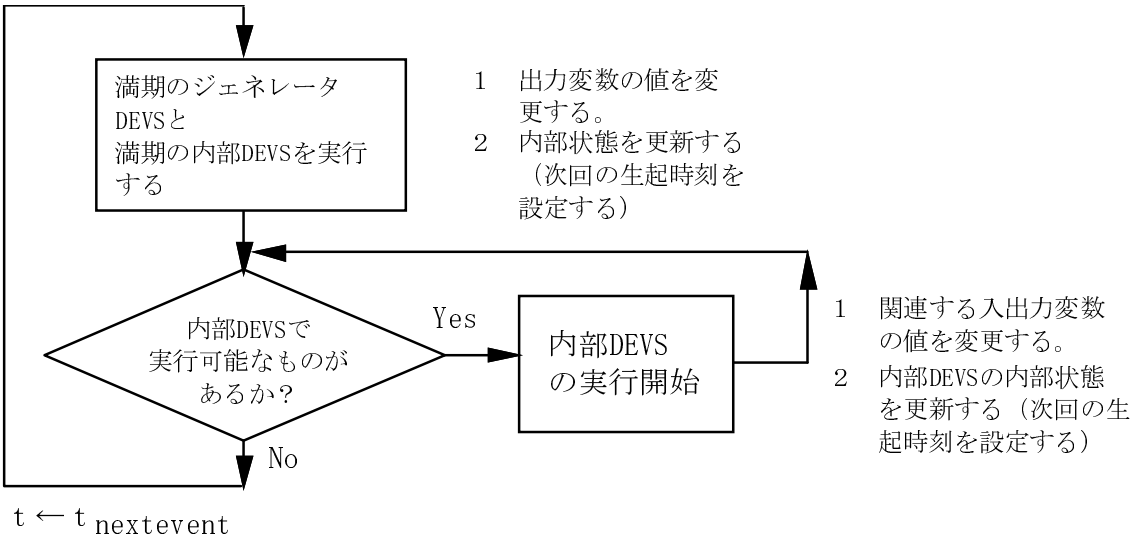


図 4-4　DEVS 結合システム全体の状態遷移を示すフローチャート

以上の例示をもとにして、一般の結合 DEVS システムの静的構造 $<Q,E,A,f_{EK},f_{AK},f_{KA}>$に対応する動的構造 $M=<S,\delta,ta>$を定義する。

全体システム中にあるジェネレータの集合が E である。Eの要素数は、つまりジェネレータの個数はm とする。内部 DEVS の個数をn とする。結合変数全体からなるベクトルが $Q=\{q:Q_{id}\to NaturalNumber\}$であり、$Q$は待ち行列のつくるベクトルである。$m$ 個のジェネレータそれぞれは$e\in E$を添え字として、$G_e=<S_e,\delta_e,ta_e>$とかける。n 個の内部活動 DEVS は $a\in A$ を添え字として $M_a=<S_a,\delta_a,ta_a>$とかける。

疑似状態　$S=(\underset{e\in E}{\times}(S_e\times T^{\infty}))\times Q\times(T^{\infty})^n$

Sの要素は$s=((s_1,t_1),...,(s_m,t_m),q,e_1,e_2,...,e_n)$とかける。 ここで各 t_e はジェネレータ G_eが疑似状態 s_eになった時刻であり、e_aは内部 DEVS　M_aが疑似状態 s_aになった時刻である。

時間進め関数　$ta:S\to T^{\infty}$

$s=((s_1,t_1),...,(s_m,t_m),q,e_1,e_2,...,e_n)$に対して、

$$ta(s)=ta_k(s_k)+t_k-t'(s)$$

ただし、$ta_k(s_k)+t_k=\underset{e\in E,a\in A}{min}\{ta_e(s_e)+t_e,ta_a(s_a)+e_a\}, t'(s)=\max\{t_1,...,t_m,e_1,...,e_n\}$で

ある。ここで $s_a=q|_{f_{AK}(a)\cup f_{KA}(a)}$つまり、$s_a$は結合変数 q の中で、a という内部 DEVS に関係する（入力と出力）結合変数だけを取りだしたものである。

内部状態遷移関数 $\delta:S\to S$

満期になる（現在の状態を終了すべき）ジェネレータや活動を$due(s)$ とする。それらの活動を終了するイベントと他の活動の開始イベントを実行して新たな擬似状態$\delta(s)$ となるように、全体 DEVS システムの内部状態遷移関数 $\delta:S\to S$ を下記のように定める。

$s=((s_1,t_1),...,(s_m,t_m),q,e_1,e_2,...,e_n)$とするとき

$$due(s)=\{k\in E\cup A\mid ta_k(s_k)+t_k=\min_{e\in E,\,a\in A}\{ta_e(s_e)+t_e,ta_a(q_a)+e_a\}$$

$\delta((s_1,t_1),...,(s_m,t_m),q,e_1,e_2,...,e_n)=((s'_1,t'_1),...,(s'_m,t'_m),q',e'_1,e'_2,...,e'_n)$ ということは次のとおり：

$$(s'_e,t'_e)=\begin{cases}(\delta_e(s_e,t_e),t_e+ta(s)), \text{if } e\in due(s)\\(s_e,t_e), \quad \text{otherwise.}\end{cases},$$

$$q' = f_Q(f_{FinQVal}(q, due(s))),$$

$$e'_j = \begin{cases} e_j + ta(s), \text{if } j \in due(s) \cup \bigcup_{k=0}^{\infty} f_{QA}(f_{Dispatch}{}^k(f_{FinQVal}(q, due(s)))) \\ e_j, \text{otherwise.} \end{cases}$$

上の定義に使われている関数は以下のように定める。

$f_{FinQVal} : Q \cup P(A \cup E) \to Q$

$f_{StQVal} : Q \cup P(A \cup E) \to Q$

関数$f_{FinQVal}$は活動完了時に、その活動から影響を受ける結合待ち行列ベクトルを更新する関数であり、また、f_{StQVal}は活動開始時にその活動への入力となる結合待ち行列ベクトルを更新する。いずれも結合ベクトル更新関数と呼ぶ。

$f_{QA} : Q \to P(A)$;

現在の結合変数qの値から見て、どの内部 DEVS を開始できるかを決める関数で、複数のものが実行開始可能の場合もある。フローチャートの判定部分である。内部活動選択関数と呼ぶ。

$f_{Dispatch} : Q \to Q; f_{Dispatch}(q) = f_{StQVal}(q, f_{QA}(q))$

この関数は、状態遷移フローチャートの「内部 DEVS の実行開始」を表現している。qの値からf_{QA}によって選択された内部活動の開始したことがqに記録される。内部活動開始関数と呼ぶ。内部活動選択関数と結合ベクトル更新関数によって定まる。

$f_Q : Q \to Q$;

現在の結合変数の値から開始可能な内部 DEVS をすべて開始する仕組みを表現する関数。状態遷移フローチャートでの、条件判定による繰り返しループを表現する。内部活動実行関数と呼ぶ。内部活動選択関数と結合ベクトル更新関数が定まれば、次のように定まる。

$$f_Q(q) = \begin{cases} q, \text{if } f_{QA}(q) = empty \\ f_Q(f_{StQVal}(q, f_{QA}(q))), \text{if } f_{QA}(q) \neq empty \end{cases}$$

ただし、つぎの３条件を満たす。つまり、結合変数の値の更新や時間進め関数は、結合している変数にしか影響しないようにローカライズ（局所化）されている：

(1) ひとつの活動aだけが終了した場合の全体ファイルの更新は、aによる更新と同じ変更結果となること。つまり：

$$f_{FinQVal}(q, \{a\})|_{f_{AK}(a) \cup f_{KA}(a)} = \delta_a(a)$$

(2) 活動aが開始されない状況では、全体ファイルの更新は活動aの満期時間に無関係であること。つまり：

任意の$u \in P(A)$と$a \in A$ に対して、$a \notin f_{QA}(q_a)$ならば、

$ta_a(q|_{f_{AK}(a) \cup f_{KA}(a)}) = ta_a(f_Q[f_{StQVal}(q,u)]|_{f_{AK}(a) \cup f_{KA}(a)})$であること。

（結合 DEVS システムの定義終わり）

上で、条件(1) は、完了した活動が a だけのとき、それによる待ち行列システムを変更した値である$f_Q[f_{FinQVal}(q,\{a\})]|_{f_{AK}(a) \cup f_{KA}(a)}$は、構成要素$a$の状態遷移関数である$\delta_a$によって変更した値$\delta_a(a)$であることを要求する。条件(2) は、活動$a$が開始されないときには、その$a$の満期時間$ta_a(q_a)$は、他の活動によって起こる待ち行列システムの変更の影響を受けないことを示す。つまり、変更後の値である

$$ta_a(f_Q(f_{StQVal}(q,u))|_{f_{AK}(a) \cup f_{KA}(a)})$$

は、変更前の満期時間である$ta_a(q|_{f_{AK}(a) \cup f_{KA}(a)})$と同じ値であることを要求する。内部活動実行関数$f_Q$は、定義式の中に$f_Q$が使われて再帰的に定義されているが、後述の定理 1 が成立するので、定義は well-defined である。

図 4-2 の買掛金決済プロセスの AID に対する上記の各関数の具体形を、内部 DEVS として a =「請求書照合」について示す。静的構造の例で見たように、$f_{AK}(a) \cup f_{KA}(a) =$ {請求書, 仕入れ記録, 照合担当, 照合済請求書} である。任意の待ち行列の状況$q \in Q$に対して、内部活動選択関数の定義は次のとおりである。

$a \in f_{QA}(q) \Leftrightarrow$照合を待つ請求書があり、照合担当の手が空いていて、かつ、請求と照合すべき仕入れ記録があること

$\Leftrightarrow q($請求書$) \geq 1,\ q($照合担当$) \geq 1,$かつ$q($仕入れ記録$) \geq 1$

$\Leftrightarrow (\forall q_i \in f_{AK}(a) \cup f_{KA}(a))(q(q_i) \geq 1)$、ただし、$q(q_i)$は結合待ち行列変数$q_i$（たとえば$q_i$=「照合担当」）の現有数である。

定義 4　DEVS 結合システムの動的構造

任意の DEVS 結合システムの静的表現として、待ち行列システム、外生トランザクション、内生トランザクションからなる$<Q,E,A,f_{EK},f_{AK},f_{KA}>$という組を考える。この DEVS 結合システムの静的表現に対して、上記のような条件を満たす$<S,\delta,ta>$をその動的構造と呼ぶ。

定義５　DEVS 結合システム

待ち行列システム、外生トランザクション、内生割り当てトランザクション、内生終了トランザクションの全体が、ひとつの DEVS 結合システムであるとは、静的構造とその動的構造を持つことである：

次の定理によって、結合 DEVS システムの動的構造は確かに状態表現となっている。つまり結合 DEVS システム自体が、全体として動的構造で表現される状態遷移をする DEVS であることを示せる。

定理１

任意の DEVS 結合システムにおいてすべての外部 DEVS と内部 DEVS が有限時間消費を満たすとする。このとき、どんな状態からも有限個の活動だけが開始可能であるならば、全体の結合 DEVS システムも有限時間消費を満たし、したがって$<S,\delta,ta>$は状態表現となる。すなわち：
もし、ある正整数pがあって、

$$\bigcup_{k=0}^{\infty} f_{QA}(f_{Dispatch}{}^{k}(f_{QueVal}(q, due(s)))) = \bigcup_{k=0}^{p} f_{QA}(f_{Dispatch}{}^{k}(f_{QueVal}(q, due(s))))$$

が成立するなら、全体としての結合 DEVS $<S,\delta,ta>$も有限時間消費を満たす。

(定理の証明)
外部 DEVS(ジェネレータ)の個数をmとし、内部 DEVS の個数をnとする。結合 DEVS $<S,\delta,ta>$が有限時間消費であることを、次の形で示す。
「任意の正の時間$\Delta > 0$と、$k \geq 0$と$s \in S$ に対して、適当な整数$h \geq 0$があって、$\sum_{p=k+1}^{k+m+n+h} ta(\delta^{p}(s)) \geq \Delta$が成立する。」これがいえれば、全体 DEVS システムの状態遷移関数は well-defined になる。

証明の基本的考え方は、(1) $m+n+1$の状態遷移をしたとすると、$m+n$個しかない DEVS のどれかひとつが必ず２回以上状態遷移していること、(2) 全体の適当回数の状態遷移によってただ一つの活動が状態遷移すると、その状態遷移による全体状態の満期時間の和の増加分は、その遷移した活動の満期時間の和の増加分と一致すること、(3) したがってひとつの状態から多数回の遷移を繰り返すとどれかの DEVS の有限時間消費時間条件によって全体の有限時間消費時間

条件が満たされることになる。以下に(1)ー(3)が成立することを示す。

(1) 任意の $k \geq 0$ と $s \in S$ をとる。$s = ((s_1,t_1),...,(s_m,t_m), q, e_1,e_2,...,e_n)$ に対して δ を k 回適用した状態 $\delta^k(s)$ を $s^k = ((s^k{}_1,t^k{}_1),...,(s^k{}_m,t^k{}_m), q^k, e^k{}_1,e^k{}_2,...,e^k{}_n)$ とかくことにする。結合 DEVS の δ の定義より、s^k から $s^{k+1} = \delta(s^k)$ へと遷移するときには、ある $e \in E$ か $a \in A$ があって、

$e \in due(s^k)$ か、$a \in due(s^k) \cup \bigcup_{h=0}^{\infty} f_{QA}(f_{Dispatch}{}^h(f_{FinQVal}(q^k, due(s^k))))$ か、あるいは両方が成立する。つまり、δ による 1 回の遷移では少なくとも E か A の要素が 1 つは状態遷移をする。

(2) したがって、s^k から $s^{k+m+n+1}$ までの δ による遷移では、s^k から $\delta(s^k)$ へと遷移したときの E か A の要素の DEVS が 2 回以上遷移する。その DEVS を $a \in A$ をする。（a のかわりに ある $e \in E$ である場合にも以下の議論は同様に成立するので、こう仮定しても一般性がある。）

$a \in due(s^k) \cup \bigcup_{h=0}^{\infty} f_{QA}(f_{Dispatch}{}^h(f_{FinQVal}(q^k, due(s^k))))$ である a が、これ以降、はじめて遷移するのが $k+i$ 回目の遷移であるとする。つまり、各 j（$1 \leq j < i-1$）について

$$a \notin due(s^{k+j}) \cup \bigcup_{h=0}^{\infty} f_{QA}(f_{Dispatch}{}^h(f_{FinQVal}(q^{k+j}, due(s^{k+j}))))$$

かつ、

$$a \in due(s^{k+i}) \cup \bigcup_{h=0}^{\infty} f_{QA}(f_{Dispatch}{}^h(f_{FinQVal}(q^{k+i}, due(s^{k+i}))))$$

とする。すると□の定義から、各 j（$2 \leq j \leq i$）について

$e_a{}^{k+j} = e_a{}^{k+j-1} = e_a{}^{k+1}$ および、$e_a{}^{k+i+1} = e_a{}^{k+i} + ta(s^{k+i})$

が成立する。ここで、ta_a と f_{StQVal} の性質の(2)から、各 j（$1 \leq j \leq i$）について $ta_a(q_a{}^{k+j}) = ta_a(q_a{}^{k+1})$ である。また、各 j（$1 \leq j \leq i$）について $ta_a(q_a{}^{k+j}) = ta(s^{k+j})$ であるから、$ta_a(q_a{}^{k+1}) = ta(s^{k+j})$。

よって、

$$e_a{}^{k+i+1} = e_a{}^{k+i} + ta(s^{k+i}) = e_a{}^{k+1} + ta_a(q_a{}^{k+1}) \qquad (*)$$

また、に δ よる k+1 回目から k+i+1 回目までの遷移について次が成立する。

$$e_a{}^{k+i+1} = e_a{}^{k+1} + ta(s^{k+1}) + ta(s^{k+2}) + \cdots + ta(s^{k+i})$$

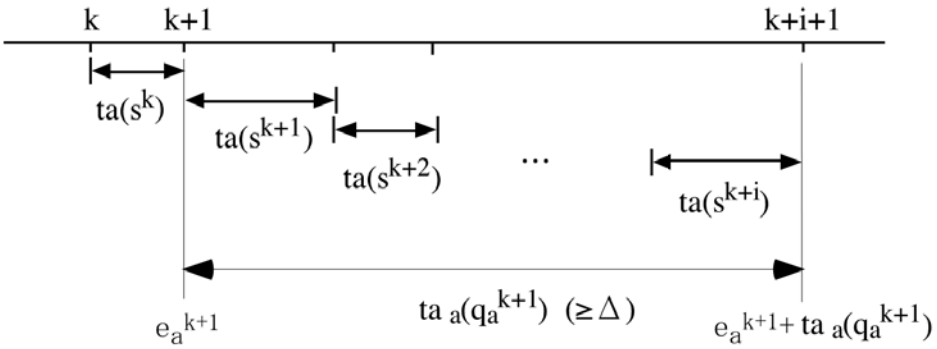
$$= e_a{}^{k+1} + \sum_{p=k+1}^{k+i} ta(s^p) = e_a{}^{k+1} + \sum_{p=k+1}^{k+i} ta(\delta^p(s)) \qquad (**)$$

よって(*), (**)から

$$e_a^{k+1} + \sum_{p=k+1}^{k+i} ta(\delta^p(s)) \;=\; e_a^{k+1} + \;\; ta_a(q_a^{k+1})$$

を得る。これより、$k+i$回の全体状態遷移の中で活動aだけが２回遷移するとき、全体の満期時間はaだけが２回遷移するときと同じ値となる。

(3) したがって、全体が多数回遷移するとき、全体の満期時間の和はこのような構成要素の活動の満期時間の和と一致する。各活動は有限時間消費条件を満たすので、全体も満たす。



（証明終）

次の定理は、同一な静的なモデルを持っていても動的な振る舞いが異なる原因を述べている。これまでの構造の分析により、証明自体は簡単であるが、概念的に重要である。同じ静的モデルによってビジネスプロセスがモデル化されたとしても、情報システムや運用方針が異なると、異なるパフォーマンスになることの原因を述べている。

定理　2

任意のDEVS結合システムの静的構造$<Q,E,A,f_{EK},f_{AK},f_{KA}>$という組を考える。このDEVS結合システムの静的表現に対して、２つの動的構造$<S,\delta,ta>$, $<S,\delta,ta>$が同一となるのは、

内部DEVS$\{M_a \mid a \in A\}$、ジェネレータDEVS$\{M_e \mid e \in E\}$、

結合ベクトル更新関数$f_{FinQVal} : Q \cup P(A \cup E) \to Q$, $f_{StQVal} : Q \cup P(A \cup E) \to Q$

内部活動選択関数$f_{QA} : Q \to P(A)$;

内部活動実行関数 $f_Q : Q \to Q$ ；
がすべて等しい時である。

（証明）
DEVS 結合システムの静的構造が所与の時、動的構造を決定するためには、全ての要素 DEVS、結合ベクトル更新関数、内部活動選択関数、内部活動実行関数が定まればよい。したがってそれらが等しければ、全体システムも同一となる。

QED

4.4 業務取引システム

DEVS 結合システムの待ち行列は先入れ先出し（FIFO）とか優先順位による作業順番指定など、待ち行列を作っているオブジェクトについての比較的簡単な状況しか記述できない。ビジネスプロセスでは、たとえば、発注品目の分割納入とか、生産計画や物流計画の作成や計画による運転など、雑多で多様な情報を待ち行列のオブジェクトとして取り扱う必要がある。そのため、待ち行列のオブジェクトが持つ情報の記述能力を向上させて、ファイルシステムに置き換えたものが業務取引システムである。ファイルシステムを導入することによって、DEVS 結合システムにおいて活動間の同期をとっているのが結合変数であることがよく理解される。言い換えれば、活動がファイルシステムを変更したり、また、ファイルシステムが活動の開始を決定することを通じて、活動間の影響関係を決定する相互作用を媒介している仕組みがファイルシステムであることがわかる。

図 4-2 の買掛金決済プロセスの場合を例にとって、結合変数をファイルシステムに置き換えて得られる買掛金決済の業務取引システムを例として示す。その後で、形式的な定義によって一般に業務取引システムを示す。

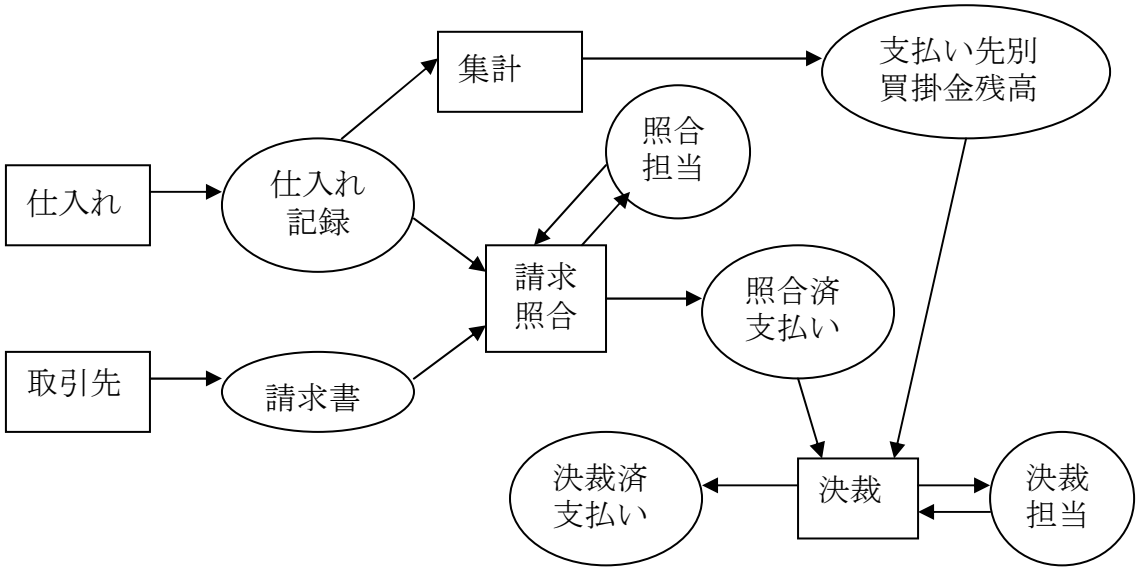


図 4-5.　買掛金決済プロセス（集計活動を追加）

買掛金決済プロセスを構成する各活動は次のような作業からなっている（トーマツ・コンサルティンググループ編、購買・固定資産管理システムの設計、1997）。

1) 請求照合活動（支払い依頼業務）

(1) 仕入れ等の発生データの取り込み

(2) 支払い条件マスターに記載した条件を適用して支払依頼データを作成する。支払対象は多様であって、購買の仕入れ計上データのほかに、経常的支払として家賃、水道光熱費、購読料、リース支払いがある。また、一回限りの購入による支払もある。

支払日の設定

毎月発生するもの：例　支払日１５日、開始年月、終了年月

支払いを年に 4 回行うもの

指定した年月日にのみ発生するもの

いくつかの当社部門が共通の支払先をもつときには期間ごとに取りまとめる。

支払い方法

振込み支払い：決済銀行から振込み

期日現金払い：一定期間経過後の振込み支払い

自動引落支払い：決済銀行から口座からの口座振替による支払い

手形払い：支払い手形による

相殺支払い：支払依頼データの修正や売掛金との相殺を行う
仕入れ先の本社が同じでも、支払先が異なる場合があるので、適切な決済銀行口座を指定する。
支払予定日の変更や、振込支払と手形支払を変更したりすることもある。手形の場合は、支払期日やサイトを指定する必要がある。
ターンアラウンド伝票（こちらが請求時に出した伝票のカーボンコピー部分を、支払い者が支払い時使用する伝票）の伝票番号を使う場合も多い。

2) 決裁（支払決裁業務）
決済銀行口座の管理を資金管理との関連で決定したり、変更したりする。基本的に支払依頼処理によって作成された支払金額、支払方法、実際支払日を確定する。決済銀行を指定していない支払依頼について、当社が有している決済銀行口座の中からひとつを指定する。支払決裁後は変更することはないような運用にする。
(1) 決裁処理のタイミング　経常支払依頼データとして毎月自動計上されるもののうち、確定されいないものやゼロ金額のものは、決裁（確定）しないようにする。後行程が無駄だから。
(2) 支払決裁銀行の決定 各支払先（債務管理単位）ごとに、支払いの決裁銀行をマスタファイルで指定する。
(3) 支払い処理後のデータ処理
決裁銀行の変更、振込先銀行変更、郵送料変更、実際支払日変更

請求照合と決済の活動の結果用意された「決裁済み支払いデータ」を使い、以下の一連の活動が行われる。
３）支払い処理
(1) 払込み支払いデータの作成
(2) 支払手形データの作成
(3) 支払い実績データ、
４）仕分けデータ自動作成
５）マスタ管理　取引銀行マスタ、支払い条件マスタ、カレンダ・マスタ、振込手数料マスタの各ファイルの更新・管理。
６）債務管理情報作成出力処理

7）月次更新処理　未払いの計上、前払いの計上、締処理

これらの活動が論理的に正しい順序で行われていくために、次のようなデータが用いられる。ファイルシステムを図 4-6 のようなデータモデルによって構成し、関係データベースを使って図 4-7 の中の表（テーブル）の形式で実現することができる。これを疑似状態とする状態遷移は図 4-7 のようになる。ただし、図においてイベントが発生した時に変化する部分をアミカケで表している。

図 4-5 の結合変数は、それぞれ表 4-2 のようなテーブルによって表現されている。

結合変数名	対応するテーブル名
請求書	支払い見出し
仕入記録	仕入れ見出し
	仕入れ明細
照合済み支払い	支払い照合見出し
	支払い照合明細
決裁済み支払い	決裁済み支払い
支払先別買掛金残高	支払先別月次買掛サマリ

表 4-2 結合変数のデータモデル表現

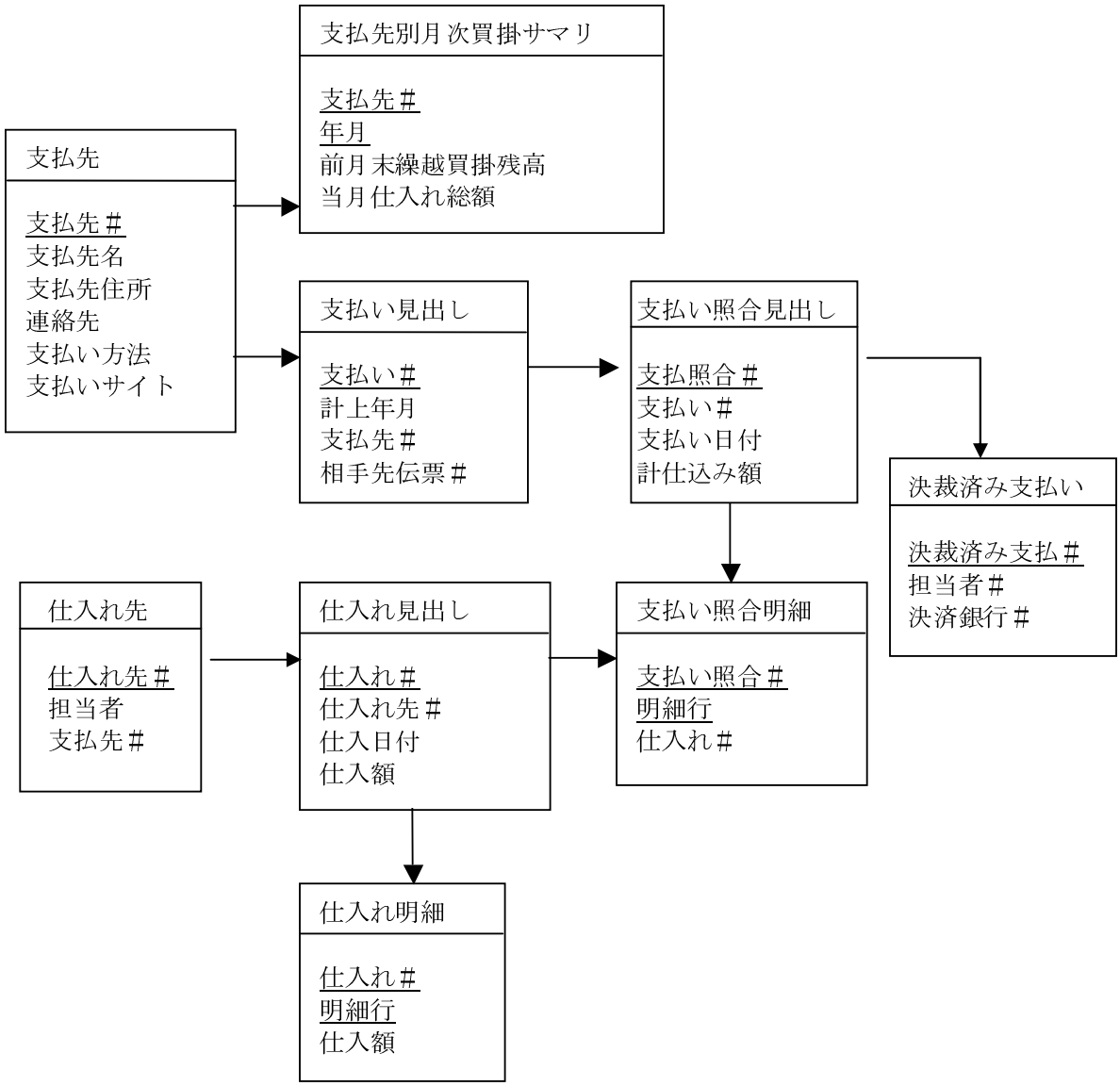


図 4-6 買掛金決済プロセスの結合変数のデータモデル

仕入れ先		
仕入れ先＃	担当者	支払先＃
72	堺商人	2031
89	加藤他人	2031
546	加藤他人	2031

支払先					
支払先＃	支払先名	支払先住所	連絡先	支払方法	支払サイト
2031	(有)佐藤商	東東京市1-2-	03-9095-	振込	ー
1095	(株) 山商	北東京市4-5-	03-9345-	手形	90
290	佐藤個人	南つくば市1	029-123-	振込	ー

図 4-7(a) ２つのマスターテーブル：仕入れ先と支払先

	取引先	仕入係	照合担当	決裁担当
t=220	(#1282981, 210)	(#9328178, 217)	-	(#100186, 203)

（ある取引先が1282981番の伝票を発行するところである。仕入れ係は9328178番の仕入れの確認記録をt=217に開始した。
決済担当はt=203 に作業を開始した）

支払い見出し

支払 #	計上年月	支払先 #	相手先伝票 #
83258	20XX/02	2031	342032
83243	20XX/02	14	2,250,000

仕入れ見出し

仕入れ #	仕入れ先 #	仕入日付け	仕入れ額
9328145	546	20XX/02/09	231,000

仕入れ明細

仕入れ #	明細行 #	仕入額
9328145	010	200,000
9328145	020	31,000

支払い照合見出し

支払照合 #	支払い #	支払い日付	消し込み額
100185	83243	20XX/02/25	2,250,000
100186	83258	20XX/02/25	320,000

支払い照合明細

支払照合 #	明細行 #	仕入れ #
100185	010	9328139

支払い決済済み

支払決裁済み #	担当者 #	決済銀行 #
100183	1022	1458176

支払先別月次買掛サマリ

支払先 #	年月	前月末繰越買掛残高	当月仕入れ総額
2031	20XX/01	33,500	220,525
1095	20XX/01	23,500	908,000

	取引先	仕入係	照合担当	決裁担当
t=224	(#1282981, 210)	(#9328179, 224)	-	(#100186, 203)

（仕入へ9328178番のデータを追加した。仕入れ係は9328179番の仕入れの確認記録をt=224に開始した。）

仕入れ見出し

仕入れ #	仕入れ先 #	仕入日付け	仕入れ額
9328145	546	20XX/02/09	231,000
9328178	72	20XX/01/17	145,700

仕入れ明細

仕入れ #	明細行 #	仕入額
9328145	010	200,000
9328145	020	31,000
9328178	010	30,150

	取引先	仕入係	照合担当	決裁担当
t=225	(#1282981, 210)	(#9328179, 224)	-	(#100186, 203)

（取引先から請求され支払いへ83259として記録した）

支払見出し

支払 #	計上年月	支払先 #	相手先伝票 #
83258	20XX/02	2031	342032
83259	20XX/02	2031	1282981

	取引先	仕入係	照合担当	決裁担当
t=225	(#3522, 225)	(#9328179, 224)	(#100187, 225)	(#100186, 203)

（照合担当が支払いと仕入れを照合して、支払い依頼を作成する作業を開始した。
仕入れの9328145, 9328178をあわせて100187の照合済み支払依頼とする）

支払い照合見出し

支払照合 #	支払い #	支払い日付	消し込み額
100185	83243	20XX/02/25	2,250,000
100186	83258	20XX/02/25	320,000
100187	83259	20XX/02/25	581,000

支払い照合明細

支払照合 #	明細行 #	仕入れ #
100185	010	9328139
100187	010	9328145
100187	020	9328178

t＝231	取引先	仕入係	照合担当	決裁担当
	(#3522, 225)	(#9328180, 231)	(#100187, 225)	(#100186, 203)

（仕入れ係が仕入れ見出し 9328179番と、それの２つの仕入れ明細を記入）

仕入れ見出し

仕入れ #	仕入れ先 #	仕入日付け	仕入れ額
9328145	546	20XX/02/09	231,000
9328178	72	20XX/02/17	350,000
9328179	72	20XX/02/19	50,000

仕入れ明細

仕入れ #	明細行 #	仕入額
9328145	010	200,000
9328145	020	31,000
9328178	010	350,000
9328179	010	9,000
9328179	020	41,000

t=232	取引先	仕入係	照合担当	決裁担当
	(#3522, 225)	(#9328180, 231)	(#100187, 225)	(#100185, 232)

（決済担当者1022が100186の処理を終えて記録した。すぐに100185の決済処理を開始した）

支払い決済済み

支払決裁済み #	担当者 #	決済銀行 #
100183	1022	1458176
100186	1022	9088190

図 4-7(b) 買掛金決裁プロセスの状態遷移（表 4-1 に対応する）

業務取引システムでは、以上のように、結合変数の待ち行列がファイルシステムに取り替わっている。したがって、形式的な定義を明示すると次のようになる。

第 3 章のデータモデルと同じ記号を使い、KS は管理実体型の集合、FTS はファイル型の集合、$FT : KS \to FTS$ をファイル型関数、dom はドメイン関数、LES はリスト実体を要素とするリスト実体集合、などとする。$x \in KS$ に対して、$FT(x)$ は x のファイル型であった。

このとき、$File = \{f : FTS \to LES\}$,ただし f はファイル内容関数であって、管理実体型 $x \in KS$ について、その属性が $A(FT(x)) = \{a_{i_1},...,a_{i_n}\}$ であるときに $f(FT(x))$ が $S(dom(a_{i_1})) \times ... \times S(dom(a_{i_n}))$ の部分集合である。$f(FT(x))$ は $FT(x)$ のある瞬間のテーブルのデータを表わす。

定義１　外生トランザクション

E：外生トランザクションを要素とする有限集合。

$f_{EK} : E \to P(KS)^+$　外生トランザクションが起こると、あらかじめ決められた管理実体型のファイルにデータが記録される。外生トランザクションが発生したときに影響するファイルを表わす関数。

したがってすべての外生トランザクションは f_{EK} によって影響しうる管理実体型を必ずいくつか持つ。

定義２　内生トランザクション

A: 内生終了トランザクションを要素とする有限集合。

$f_{AK} : A \to P(KS)^+$　内生トランザクションが終了したり開始したりしたとき、あらかじめ決められたファイルにデータが記録される。内生トランザクションが開始や終了したときに影響するファイルを表わす関数。

$f_{KA} : A \to P(KS)^+$　内生トランザクションが開始するための条件が書かれているファイルを指定する。

f_{AK} は活動から出て行く矢印を表現し、逆に、f_{KA} は活動に入ってくる矢印を表わす。

E, A は互いに素とする。

定義３　業務取引システムの静的表現 (static structure of business transaction system)

ファイル、外生トランザクション、内生トランザクションからなる組み
$< File, E, A, f_{EK}, f_{AK}, f_{KA} >$を業務取引システムの静的表現という；

定義４　業務取引システム（business transaction system）

ファイルシステム $File$、外生トランザクション、内生割り当てトランザクション、内生終了トランザクションの全体が、ひとつの業務取引システムであるとは、つぎの静的構造と動的構造を持つことである：

（１）$< File, E, A, f_{EK}, f_{AK}, f_{KA} >$という静的表現を持つこと。

（２）動的構造は、DEVS 結合システムの動的構造で待ち行列ベクトルをファイルシステムに取り換え、ファイル更新関数の条件にファイルシステムの一貫性を保持することを追加したもの。

つまり、以下のようになる。

m 個のジェネレータそれぞれを $e \in E$ を添え字として $G_e =< S_e, \delta_e, ta_e >$ とかき、n 個の内部活動 DEVS を $a \in A$ を添え字として $M_a =< S_a, \delta_a, ta_a >$ とかく。

疑似状態　$S = (\underset{e \in E}{\times}(S_e \times T^{\infty})) \times FIle \times (T^{\infty})^n$

を使って、結合 DEVS システムと同様の動的構造が定義される。ただし、結合ベクトル更新関数

$f_{FinQVal} : Q \cup P(A \cup E) \to Q$

$f_{StQVal} : Q \cup P(A \cup E) \to Q$

に対応する関数をファイル更新関数と呼び、表記を変えて次のようにする。

$f_{FinFileVal} : File \cup P(A \cup E) \to File$

$f_{StFileVal} : File \cup P(A \cup E) \to File$

これらのファイル更新関数が、任意のファイル内容関数の更新に際して一貫性条件 C_{File} を満たすこと。つまり：

$(\forall f \in File)[(f, f_{FinFileVal}(f)) \in C_{File}, (f, f_{StFileVal}(f)) \in C_{File}]$

結合 DEVS の場合と比較すると、ファイル更新関数の一貫性保持条件が追加されている。これは、どのようなファイルの変更も、ファイルシステムの一貫性条件 C_{File} を満たすこと、言い換えれば、一貫性条件を満たすファイル書き換え

しか許さないということである。これは、データベース管理システムを用いて参照制約等を設定しておくことで保持できる。

結合 DEVS システムと比較して、業務取引システムでは、ファイルシステムに現実の出来事を記録できる能力が比較にならないほど大きい。

たとえば、営業業務を表わす場合を考えると、たくさんの顧客からのそれぞれの注文を区別して表現する（記録する）場合に、ファイルシステムでは可能だが素朴な待ち行列では不可能である。一方で、それにもかかわらず、動的構造は全く同型である。したがって、業務取引システムに対しても定理 2 が成立する。この事実は、ビジネスプロセスの構造やモデル化の基礎的なことがらを明らかにしている。

ファイルシステムの実現はコンピュータシステム（データベース管理システム）を使う場合が多いが、まったくコンピュータを使わないことも可能である。伝票や諸帳票を使って整然と記帳して内容を一貫性のあるものに保つことができればよい。したがって、ビジネスにおける帳票ベースのファイルシステムは、コンピュータが一般に利用可能となる 1950 年代よりはるか昔から実現されていた。逆に、すべてのデータが電子化された電子商取引の場合においても、電子帳票が企業間でやりとりされるので、組織をまたがった大きなビジネスプロセスとして一貫して動く仕組みになっている必要がある。特に、全体としてデータが一貫して整合している必要がある。この場合も全体として業務取引システムになっている。

さて、ビジネスプロセスの構築において次に問題となることは、このように表現されたビジネスプロセスの動的な特性の分析である。そのために記述方法と、それで表現できる特性を後の章で考える。

問 4-4

業務取引システムのモデル化練習として、情報システム方法論の本で扱われている例題の記述を使うことができる。第 6 章の章末問題を参照のこと。

第 5 章 ビジネスプロセスのコントロール

ビジネスを捉える方法はいくつかある。最も歴史があるのは会計的な方法である。ビジネスを会計的に測定するわけである。また、4 章で説明したような、アクティビティ・インタラクション・ダイアグラム（AID）とデータモデル（DM）を使った業務取引システムによる方法もある。

本章では、ビジネスプロセスをコントロールするために発明され使われてきた伝統的な方法を説明する。まず、すべての業種のビジネス活動を金銭データによって総括的にとらえることができる会計的な方法を説明する。次に、生産のビジネスプロセスに対して生産計画によって時間軸を一定期間ごとに区切って動特性を管理する方法を述べ、その後で、業務取引システムモデルを使った時間的特性を調べる方法を説明する。

5.1 損益計算とバランスシート

アクティビティ・インタラクション・ダイアグラムを用いると、いろいろなビジネスプロセスの構造の違いが表現される。少し抽象化すると、どのようなビジネスでも、製造業の場合には図 1 のような部署を設定して分業していると見ることができる。

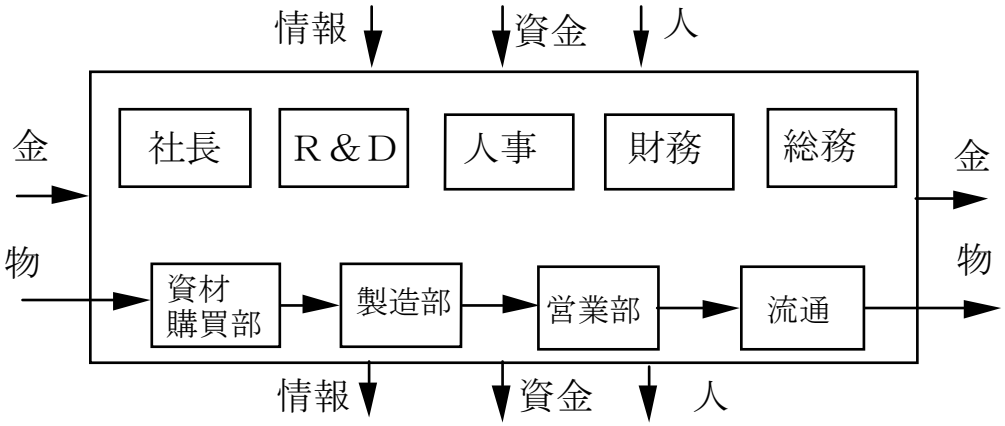


図 5-1　組織を取り巻く人、物、金、情報

企業は法人といわれるように、社会的な存在者である。縦の矢印の流れのように、製造に必要な機械や保守用具、工場や事務所建物、自動車などの固定的な設備は設立時などに集めた資本金が姿を変えたもの（投資した）である。人を雇うので給与を支払うし、企業の外部環境や内部の操業状態についての情報に基づいて日々の意思決定や、新製品開発・多角化のための新規事業などの意思決定が行なわれる。

また、上の図の横の矢印のように日々物を受け入れ加工して販売しているモノの流れがあると共に、売上金や支払いや借入金返済といったカネの流れもある。購買から

製造して営業で売って物流を通じて顧客に届けるまでの業務プロセスを広義のロジスティクス（モノが流れるプロセス）といい、また、それらの活動で発生するお金の出入りを扱う財務関連業務と合わせて基幹業務という。つまり、基幹業務とは購買、製造、営業、物流、財務会計、人事、研究開発、総務である。基幹業務で発生・参照するデータを統合的に扱う情報システムが ERP-enterprise resource planning package- 基幹業務システムである。ひとことで製造業といっても、多少でも細かくみると扱う製品や産業ごとにさまざまな基幹業務の実行方法の違いがある。パソコンや電化製品のような消費者に届く製品を製造する企業もあれば、石油化学やプラスチック、製鉄や半導体のように他の製造業の原料や資材となる製品を製造している場合もある。また、銀行や病院やレストランや生地の染色など、サービスを生産するサービス業に属する企業や組織もたくさんある。さらに、知識社会が深化しており、製造企業も個別の顧客に個別のサービスによって製品を届け、関係を継続するという方向に進んでいる。

基幹業務の各活動を実行するうえで運転資金を必要とする。原料や製品を買ったらお金を払う義務（債務という）が生ずるし、配送や給料なども必要である。モノを売ればお金を受け取る権利（債権という）が生ずる。時に生産のための工場を建設したり、製品を開発する活動にもお金がかかる。

いろいろな業種や業態があるが、どんな企業でもお金の出し入れがあるので、ジネスのあり方をお金という抽象化した記述方法を使ってとらえることができる。すべてのビジネス共通に表現する方法であって、損益計算書・貸借対照表・キャッシュフロー計算書などの財務諸表を用いる。微分方程式によってお金の流れを連続的に近似して表現することは、本来的に離散事象システムあるビジネスプロセスの記述にはそぐわない。会計的記述では、便宜的に、月次とか 4 半期、上・下半期や年度ごとに区切って時間軸を離散化して考えている。

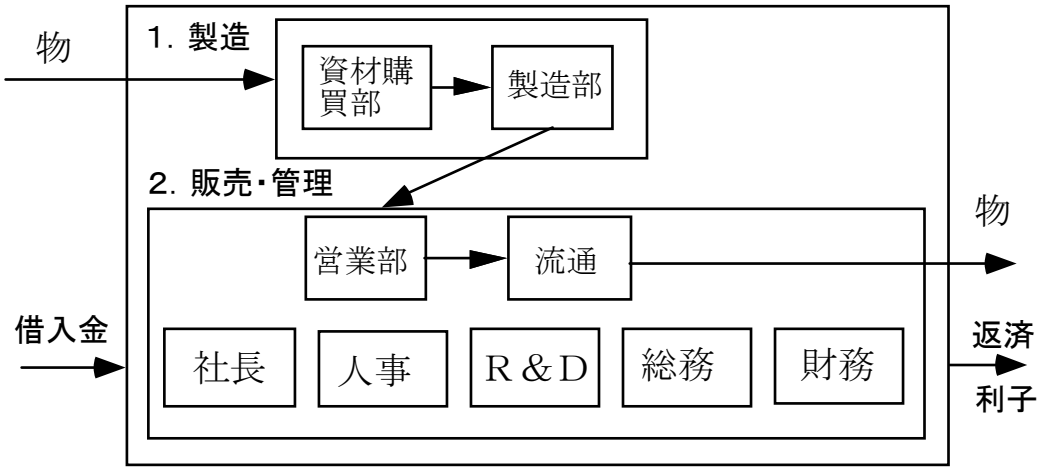


図 5-2　損益計算から見た組織イメージ

企業は、銀行や証券会社を通じて多くの人間から集めたお金を使って運営されているし、社会の特に経済的側面における重要な行為者であるので、法律によって財務諸表の作成が義務付けられている。紙だけではなくインターネットでも公表している。このように、商法や企業会計原則などの法律によって作成方法が定まっており、また、一般市民もアクセスしやすいので、損益計算書や貸借対照表は、ビジネス活動の結果を見るための便利で簡単な方法となっているのである。

企業の状態を、ある時点での企業の所有物（資産）の現状を表現するものと、一定期間内の収入・コスト・利益のようすの２種類で表現する。前者はいわば資産のようすのスナップショット（瞬時値）であり、後者はいくら使っていくら収益を上げたかという儲け方の効率をしめす。

企業の所有物（資産）の現状を表現：貸借対照表(balance sheet, B/S)
一定期間内の収入・コスト・利益を表現：損益計算書(profit and loss statement, P/L)

ビジネスを行なう組織について、B/S は静的な特徴の表現であり P/L は動的な特徴を表現する。以下はある製造業の場合の例である。例の後に各項目の説明をする。

貸借対照表

右側の項目と左側の項目とに分けてある。左側の「資産」は会社の財産である。つまり理論的には他人に売れるものである。大きくわけて流動資産と固定資産がある。流動資産は金や移動可能なすぐに売れるものである。商品や仕掛品や原材料在庫もこれである。個人にたとえれば、預貯金や株や債券、お米や牛乳や今晩のホワイトシチューのための鶏肉などである。固定資産は建物や機械や土地などである。個人でいえば家や自動車、パソコンやテレビ、本やボールペンである。

右側はそれらの資産を得るためにどこから資金を持ってきたかを記述する。「負債と資本金」のうち、負債とは借金（他人が所有する資本）のことである。１年以内にカタをつけて支払うものを流動負債、それ以上経ってから払えばよいものを固定負債という。資金の借入時に流動か固定に決まっている。ビジネスとは運転資金を借金し、それを使って製造販売して収入を得て、その借金を返し、また次の借金をして運転する。その中で利益を上げていくというイメージである。資本金はビジネスを始めるときにいろいろなものを揃えるために集めた金であり、したがってすでに機械設備や原

料や工場などに投資されていて現金としては存在しない。あとで株を発行して資本金を増やすこともできる。

（単位：10 億円）

資産の部			負債及び資本の部		
流動資産			負債の部		
当座資産		417	流動資産		494
預金・現金	87		支払手形・買掛金	271	
受取手形・売掛金	201		短期借入金	19	
有価証券	129		納税引当金	52	
棚卸資産		160	未払費用	78	
商品・製品	128		その他流動負債	74	
仕掛品	16		固定負債		147
原材料・貯蔵品	16		社債	100	
その他流動資産		90	長期借入金	41	
流動資産計		667	その他固定負債	6	
固定資産			負債計		641
有形固定資産		481	資本の部		
建物・設備	278		資本金		86
土地	199		資本準備金		163
建物仮勘定	4		利益準備金		22
無形固定資産		3	諸任意積立金		543
投資その他資産		390	当期未処分利益		86
固定資産計		874	資本計		900
資産合計		1,541	負債及び資本合計		1,541

図 5-3 製造業企業の貸借対照表

損益計算書

「収入-コスト＝利益」ということを示しているのであるが、企業を取り巻く４つの市場（製品市場、原材料市場、資本市場、労働市場）があるのを反映して、種々の収入とコストがあり、それをきちんと表現するのでいくつかの項目数がある。

製造業の損益計算書

（単位：10 億円）

売上高		1,507
売上原価	1,195	
売上総利益		312
販売費・一般管理費	212	
営業利益		100
営業外収益	29	
営業外費用	13	
経常利益		116
特別利益	0	
特別損失	2	
税引前当期純利益		114
法人税及び住民税	44	
当期純利益		70
前期繰越利益金	16	
中間配当等	0	
当期未処分利益		86

図 5-4　損益計算書

貸借対照表と損益計算書の主な項目は以下のような意味を持つ。

貸借対照表の項目

資産

所有している財産や権利の総称。流動資産と固定資産からなる。

流動資産

現金と，１年以内に現金化される資産。

棚卸し資産

商品，製品，仕掛品，原材料

ある時点での実際保有量の調査（棚卸し）によって計算される。

その他流動資産

従業員への貸付，関連企業への貸付，商品手付金，

前払い貸借料，前払い広告料，前払い利息

建物仮勘定

建設にあたっての前払い金

無形固定資産

漁業権，鉱石採掘権，水利権，採油権

特許権，実用新案権，意匠権

投資その他資産

長期出資，長期前払い費用

負債

マイナスの財産。つまり他人所有の財産で後日返済する必要があるもの。見方を換えると、資産を構成する資金の源泉のことである。

資本金

企業が発行した株式を個人や法人が買って代金を支払うと、企業には対応する資本金が集まる。

資本準備金

株式発行により株主によって支払われた金額のうち額面を超える部分，その他。

利益準備金

利益を社内に留保し，後日の巨額欠損の場合の補填に使われることもある。

（当期未処分利益の一部である）

損益計算書の項目

売上原価

商社型　　期首商品棚卸し高＋当期商品仕入れ高 - 期末商品棚卸し高

製造業型　期首商品棚卸し高＋当期製造原価 - 期末商品棚卸し高

当期製造原価＝当期材料費＋当期労務費＋当期経費＋期首仕掛在庫 - 期末仕掛在庫

販売費・一般管理費

販売費（給料，旅費），広告費，発送費，事務用品，通信費，家賃，雑費など。

営業外収益

金融的な利益。配当，受取利息，有価証券売却益

雑収入。家賃収入，受取特許使用料など

営業外費用

金融的な費用。支払い利息，有価証券売却損，売上割引，社債利息，手形割引料。

経常利益

経常成績の表現。

特別利益，特別損失

前期損益の修正，固定資産や長期保有証券の売却に伴うもの。風水害とその保険収入。

問 5-1

（1）次の貸借対照表と損益計算のブランクに適切な数値を入れなさい。

単位　10 億円

資産の部		負債および資本の部	
流動資産		流動負債	690
当座資産	560	固定負債	
棚卸資産	120	負債計	813
流動資産計		資本の部	
固定資産		資本金	
有形固定資産	421	（その他資本の部）	285
無形固定資産		当期未処分利益	12
固定資産計	520	資本計	
資産合計		負債及び資本合計	

図 5-5(a) 貸借対照表

単位　10 億円

売上高	3,540
売上原価	
売上総利益	740
販売一般管理費	
営業利益	340
営業外損益	□ 90
経常利益	

図 5-5(b) 損益計算書

問 5-2

経営指標というものが考えられている。

(a) 売上高営業利益率　＝　当期営業利益÷当期売上高

(b) 総資産利益率　＝　当期経常利益÷総資産

これらの指標が持つ意味を考えよ。たとえば、２つの小売販売の企業Ｘ，Ｙがあったとして、企業Ｘは全国展開していて総資産も総売上も大きい。企業Ｙは関東に限定してビジネスをしているので、Ｘほどの規模はないが、売上高営業利益率はＸの 2 倍以上ある。あなたが銀行の貸し出し係だとするなら、どちらのほうが優良な顧客企業だろうか。

分子と分母に B/S, P/L のさまざまな項目をとることでいろいろな経営指標（会計的な経営指標）が定義されている。次の図 5-6(a), (b) ２つの会社の損益計算書を完成し、いくつかの経営指標を計算することによって、２つの企業を比較してみなさい。

（単位：10億円）

売上高	3,700
売上原価	2,926
売上総利益	(
販売費・一般管理費	727
営業利益	(
営業外収益	80
営業外費用	88
経常利益	(
特別利益	5
特別損失	5
税引前当期純利益	(
法人税及び住民税	6
当期純利益	(
前期繰越利益金	39
中間配当等	16
当期未処分利益	54

図 5-6(a)（株）ABC の損益計算書

（単位：10億円）

売上高	7,769
売上原価	6,267
売上総利益	(
販売費・一般管理費	942
営業利益	(
営業外収益	190
営業外費用	124
経常利益	(
特別利益	－
特別損失	－
税引前当期純利益	(
法人税及び住民税	261
当期純利益	(
前期繰越利益金	77
中間配当等	38
当期未処分利益	404

図 5-6(b)　DEFG（株）の損益計算書

問 5-3 図 5-7 の売上総利益と期首在庫を、3 期に 1 度だけ 30 個を納品するサプライヤ A について計算しなさい。A も B も仕入れ価格は 1 個あたり 200 万円販売価格は 250 万円とする。

サプライヤ B

期 n	期首在庫 z(n)	n 期の入荷数 x(n)	n 期の出荷数 y(n)	売上総利益 （万円）
1	0	10	10	500
2	0	10	10	500
3	0	10	10	500
4	0	10	10	500
5	0	10	10	500
6	0	10	10	500

サプライヤ A

期 n	期首在庫 z(n)	n 期の入荷数 x(n)	n 期の出荷数 y(n)	売上総利益 （万円）
1	0	30	10	500
2		0	10	
3		0	10	
4		30	10	
5		0	10	
6		0	10	

図 5-7 サプライヤ A と B

状態遷移は各期（または、半期や４半期）ごとに環境からの入力によって離散時間的に動くものと考えればよい（図 5-8)。貸借対照表と損益計算書の変化を見る。つまりは、組織の資産（財産）の変化と、利益生産力の変化を見る。

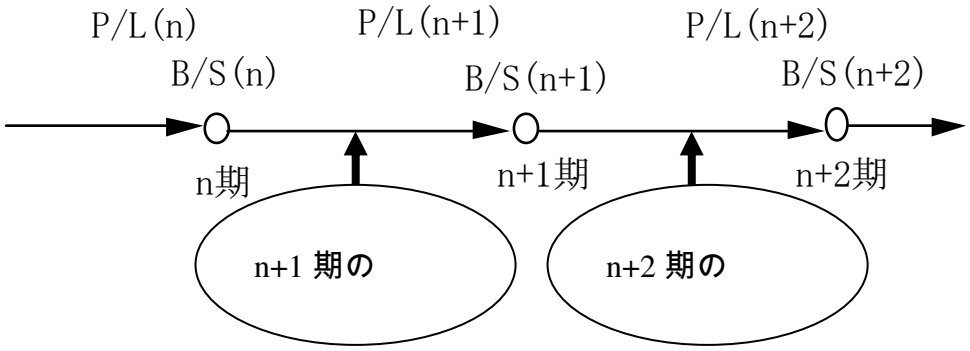


図 5-8 B/S と P/L の変化

B/S(n) = B/S(n-1)＋（n 期のビジネスによる資産と負債の増減）

P/L(n) = （n 期のビジネスによる損益）＋P/L(n-1)の次期繰越利益

なお、上の式で＋と書いてあるのはイメージであって、「含めて考えられる」ことを意味している。

多くの企業はいろいろな異なるビジネス領域で活動するために、企業グループを構成している。たとえば、発電所向けの製品と一般の家電製品では製造方法にかなり違いがあるので別会社にして最適的な経営を指向しているし、製造と販売を異なる企業にしたり、地域別に販売子会社にして競争的状況にしたり、情報システム部門を独立子会社にしたりしている。ビジネスプロセスとして関係が深い場合には、しかし、運営や経営上の意思決定をグループで一体的に行っている場合がある。企業が社会的存在者であることから、一体的運営を行っている場合には、企業グループ全体としての B/S や P/L を社会・投資家に対して開示することが義務付けられている。期末に公表する決算では、単独の決算データ以外にグループ全体をあらわす連結決算が開示される。連結決算が必要な理由の一端を以下の例で示す。

親会社である P 社と子会社 C 社がある。いずれも商品を仕入れて販売している。P 社が外部の仕入れ先から購入する業務を行い、C 社は P 社から仕入れて販売している。P 社と C 社についてそれぞれ単独で損益計算した場合と、連結決算した場合を示す。

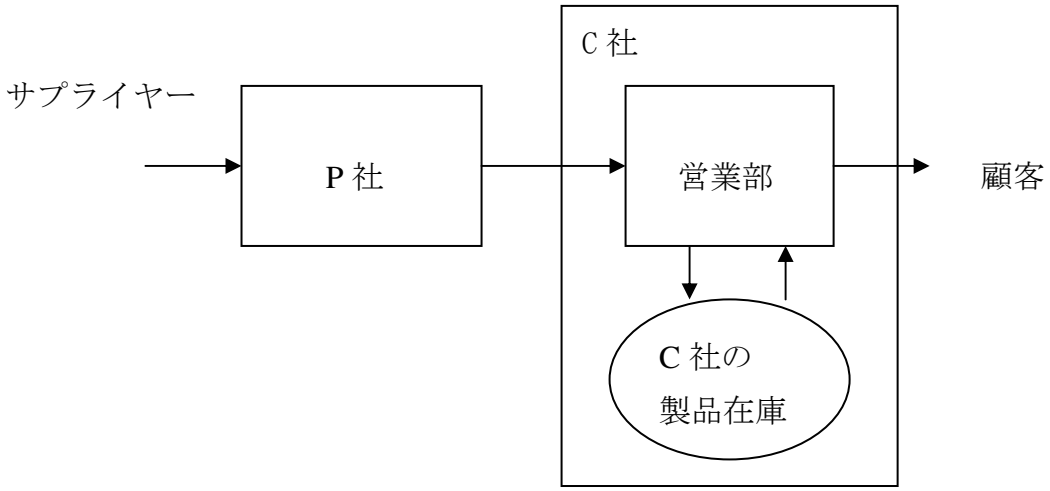


図 5-9　親会社と子会社の関係

親会社 P 社： 200 個を仕入れ。子会社 C 社に売り切った。仕入れ単価 80 万円、販売単価 100 万円であった。

子会社への売上	100*200 = 20,000
売上原価	80*200 = 16,000
販売費	3,000
営業利益	1,000

子会社 C 社では 120 個を販売した。80 個が売れ残った。販売単価は 110 万円であった。売上に寄与した仕入分だけが売上原価として計上されるので次の通り。(□はマイナスを表す)

売上高	110*120 = 13,200
売上原価	100*120 = 12,000
販売費	1,500
営業利益	□300

グループとしてのビジネスの状況を表す直感的方法として単純合算の P/L を作ると次の通りになり、700 万円の利益があったように見える。

売上高	33,200
売上原価	28,000
販売費	4,500
営業利益	700

しかし、P 社と C 社が同一グループとしてビジネスをしているので、ビジネスの図を全体的に見ると P 社が仕入で C 社が販売をおこない、グループ全体での在庫が売れ残りとして 80 個分発生している。したがって、これを P 社 C 社グループとしてひとつの会社が、200 個を単価 80 万円で仕入れて 120 個を販売単価 110 万円で販売したのだから、次の計算が正しい。販売のための経費は実際に合計額が発生している。

連結P/L

売上高	110*120 = 13,200
売上原価	80*120 = 9,600
販売費	4,500
営業利益	□900

実際には 900 万円の損失が発生したことが分かる。このように、グループ全体で連結して考える時には、グループ内取引を相殺して、全体としての経営活動からみて表す必要がある。

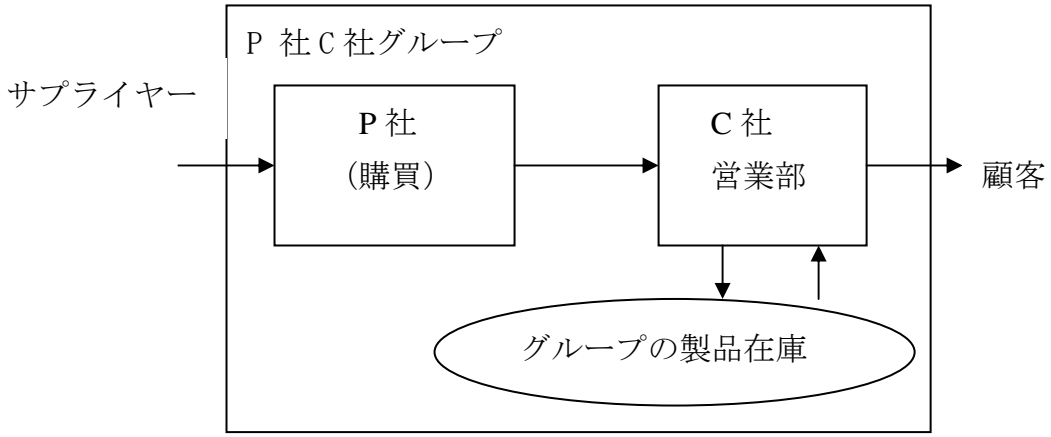


図 5-10　連結決算の考え方

5.2 原価管理（製造原価管理の場合）

形式的な原価の定義とか種類の議論を省略して、原価計算の基本的な考え方の特徴を説明する。受注生産の製造企業でよく用いられる個別製造原価計算というものだけを説明する。

原価計算は、すべての業種の企業に適用できるようにするために、図 5-11 のような粗さで企業の活動を注目する。原価計算の目的は、製品を作るのにかかったコスト（＝原価）を知ること（計算すること）であり、そのコストから改善点を探ることである。つまり、管理会計の道具である。

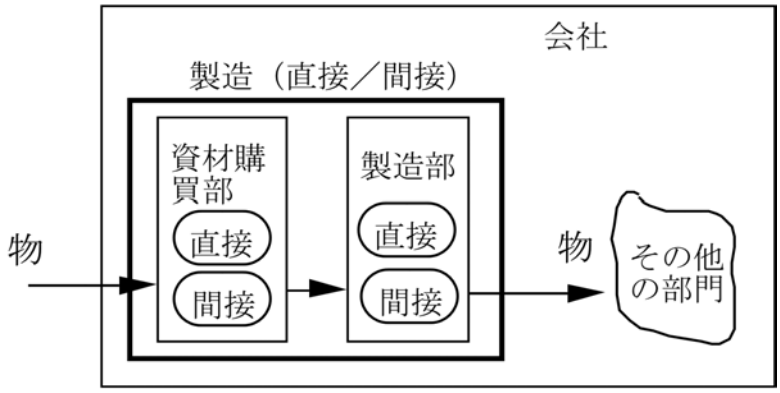


図 5-11　（伝統的）原価管理計算が想定する製造企業のイメージ

なぜ、コスト計算が簡単ではないのか？　理由は、ものを作って製品を売るまでには、いくつかの組織内のプロセスがあって、計算を実行する時に判断を要する場面があるからである。業務プロセスごとに支出がある。原材料を買うと、原材料の価格は決まる。それに付随して運搬の手数料だとか、製品の設計とか、品質管理活動とか、あるいは、役員会の費用などがかかる。生産制御用にネットワークを敷設したり、また、従業員教育の一環として新技術などの説明会を行ったりもする。工場建物を建てたり、加工機械を取り替えたりもする。支出が多岐に渡る。

一方、企業が得ることができる収入は、基本的に製品の売上だけである。ある製品を作るためのかかったお金は、実際に材料を買い機械をオペレータ動かした以外の製造に関わるすべての活動の全コストを、製品 1 個あたりに割り振ったときに、製品 1 個の売り値より大きくないと儲けはない：

$$\text{全コスト} + \text{利益} = \text{製品価格} \times \text{販売個数}$$

$$\frac{\text{全コスト}}{\text{当期製造個数}} < \text{製品価格}$$

マークアップ率といって、製品原価にこの係数を乗じて製品価格とする方式もあるが、価格とコストの間には上の不等式のような関係が成立しなければならない。たくさん

の種類の製品を作っている場合には、たとえば、製品 A についての原価は次のようになる：

$$\frac{\text{製品Aの全コスト}}{\text{製品Aの当期製造個数}} < \text{製品Aの価格}$$

収入には利息や子会社からの株式配当などもあるが、それらが収入に占める比率はわずかだ。こうしたことをすべて考慮に入れたうえで、経営管理に使えるように合理的にコストを計算したいのである。ある製品の製造にかかった全コストを計算するためには、何が分かればよいだろう。

ある鉄工所では顧客からの個別注文に応じて鉄板を切ったり熔接したり仕上の研磨や塗装をしたりしている。必要に応じて部品や製品の設計も行うし、材料の手配や運送をすることもある。時には、新技術を使った新たな工作機械を買うこともある。休憩時間や昼食時のお茶も必要だ。

原価計算では、管理の目的に使うために、伝統的に製造コストを３つの原価要素である材料費、労務費、経費に分類し、図４のように考える。

材料費：

材料購入費、材料運賃、資材購買部の費用（その製品の分だけ。検収も行う）、取り扱い手数料

労務費：

製造のための労働は、直接に機械を運転するとかの直接作業と、修理・運搬・清掃・製造のための会議打ち合わせ・段取り換えなどの製造に付随する間接作業がある。現場監督者給料、工場従業員の福利費・退職金引き当て・賞与手当てなど。

経費：

特定製品のための直接的経費は、外注加工費、設計費、特許使用料。ほとんどは間接的で、電力料・ガス代・水道料、旅費交通費・運賃・厚生費、原価償却費・賃借料・火災保険料・租税公課などがある。いわば、材料でも労務費でもなく、製造に関係する支払いはここに分類される。

製造指図書

指図書NO. 32829

生産品目名	シリンダ5A	顧　客	(株) ○　×
品目番号	KV1008	発令日	月　日
数　量	60個	着手日	月　日
納　期	○月○日	完成日	月　日
担当者		引渡	

図 5-12　製造指図書の例

３つの原価要素は各々直接費と間接費に分ける。直接材料費は、購入金額のみ。他は間接材料費。直接労働費は、直接作業時間のための給料だけ。直接経費とは、直接的経費のこと。個別注文を数多く扱っている鉄工所では、材料費・労務費・経費の間接費を、個々の製品（注文）に振り分ける必要がある。これを間接費の配賦(cost allocation)という。

受注生産をしているときは、注文ごとに図 5-12 のような製造指図書（という伝票）を発行し、生産をスタートする。ひとつの受注についての原価計算は、図 5-13 のようになる。経費はすべて間接費であった。

製造指図書 No.32829　原価計算書　　（円）

直接材料費	直接労務費	直接経費	製造間接費
700,000	650,000	50,000	610,300

合計	
適用	金額
直接材料費	700,000
直接労務費	650,000
直接経費	50,000
製造間接費	610,300
製造原価	2,010,300
製品出来高	60　個
製造原単価	33,505 (円)

図 5-13　原価計算書

間接費の配賦については、状況に応じてできるだけ合理的になるように考える。たとえば、ひと月の間の製造で３つの受注生産を行ったとする。そこで製造間接費が発生する。これを３つの受注に「合理的に」分ける必要がある。もし、どの受注生産でも材料費が高い場合には、それぞれの直接材料費の割合にしたがって製造間接費を配賦するのは一理ある。もし直接労務費が原価のかなりの部分を占めるなら、労務費の比率を使うのはひとつの「合理性」である。さらに、また、直接費の合計を比較して配賦基準にするのもいいだろう。要は、計算の簡便性（データをそろえることの簡便

性も含む）と合理性をバランスよく採用することである。配賦基準を何にするかによって製造原価は変化するが、配賦基準は自然に決まるものではなく、たとえば原価検討委員会のような場で組織において合意され、判断されるものである。

このような原価情報をもとにして、たとえば「コストダウンに取り組め」という号令より、過去の同種製品における原価構成を比較したり、あるいは原価要素別に検討を加えることで、より具体的な「10%の残業をカット」というような具体的な目標を見つけることができる。

5.3　会計的な方法の限界

損益計算などの財務諸表はビジネスの結果を知るための基本である。また、原価計算は、どんな企業でもかならず行う簿記によるビジネス記録を用いて、管理のための分析と提言が可能な道具である。さらに、製造ばかりでなく同じ考え方をサービスの生産にも適用できる便利な管理ツールであるといえる。

一方で、いくつかの重要な問題点も認識されている。

根本的問題は、歴史的には、大量生産をいかに効率的に行うために原価から問題を発見したいという問題に応えるために整備されてきた方法であるという傾向を持つことに起因する。つまり、以下のような計算方式をとっている：

(1) 現代の製造ラインは、計画主導であったり、品質管理を従業員が行い改善する活動も定常的に生産の中で行ったり、高価な工作機械を使ったり自動化したりして多品種少量生産を行っているので、少品種大量生産の時代に較べて直接費の割合が間接費よりかなり小さくなっている。原価の大きな部分を占める間接費を小さな直接費の情報をもとにして配賦するのは、原理的に困難になってきている。無理にやると、配賦のやり方で原価は大きな影響を受けるから、不正確極まりないものになる。

(2) 売れ残りの製品在庫を気にしない仕組みになっている。呑気に、次期に売れると思っている計算をする。テレビが発明されたころとか、電気洗濯器が普及し初めた頃の経済の発展期のように、工業製品が不足していて作れば作るほど売れるという前提に立っている。ところが現実は違っている。自動車やパソコンやビデオ、デジカメやケータイなど、技術発展のスピードは速く、競合製品の数は多い。作り過ぎた製品は陳腐化し二度と売れなくなる。

(3) 同様に、製造プロセス内にある仕掛り品の在庫量や原材料在庫の多少を気にしない。単に製品在庫に換算する。

(4) 多品種生産を行うための段取りの素早さを無視する。ある原価計算方式（直接原価計算）では、段取りを行うほど利益が減少する計算方法となっている。つまり段取

りをしないで機械をほとんど動かしっぱなしにするのが、計算上は利益が高い。たとえば、A、Bの２つの会社があり、どちらも月間（20 日間）５０台の製品を製造する。A 社は段取りが素早いので、５台のロットで 2 日間に５台作ることを繰り返す。B社は品質安定のために大きな５０台のロットでしか作れず、ひと月かかって一度に５０台完成するとする。５０台売れている市場では、段取り時間が０でないかぎりは、B社の方の直接原価が安いことになる。多品種少量生産の市場では、A 社が圧勝するにも関わらず。つまり製造プロセスの高速段取り能力の有効性が、素朴な原価計算では大量生産向きになっていて計れない。現代は顧客といわずに個客と呼んで、実質的に個別の顧客にあった製品を大量生産の低コストで提供するというマス・カスタマイゼーションの時代であるため、ビジネスプロセスの性能尺度としては、ずれているものがある。

(5) 顧客がもっと短い顧客リードタイムでの納品を望んでいて、自社のリードタイムはそれよりも大きいとする。どんな原価情報も、どうやればいいかを示さない。そこには金銭の情報しかない。

原価管理の側面からビジネスプロセスの結果の評価を行うことは必要だ。しかし、原価管理の結果を使ってビジネスプロセスの改善をやろうとすると、上のような理由からマスカスタマイゼーションには適さないどころか、場合によっては、リードタイムを短縮するための段取りの高速化のための改善を阻害したりするので、止めたほうがよいとジョンソン（米国製造業の復活, 1994）は指摘した。

5.4 時間生産性

ビジネスプロセスを改善しようとするとき、原価ではなく、直接に時間的な特徴に注目すべきである。マスカスタマイゼーションの経済では時間生産性の概念が必要なのだ。ビジネスを制御対象としてのシステムとして認識する場合、処理プロセスとしての制能分析が必要になる。そこで、会計的な側面ではなく時間的側面に注目する。田中一成（時間生産性をどう高めるか, 1993）によれば、２つの時間生産性を分類できる。

（１）時間量生産性

単位時間あたりの量＝処理スピードを表わす時間的指標。

作業生産性指標　単位時間あたりの生産個数とか生産金額

設備生産性指標　設備稼働単位時間あたりの生産個数とか生産金額

設計生産性指標　設計作業時間あたりの設計図面枚数

（2）期間生産性

あることを成すのにどのぐらいの期間が必要かを表わす。工数削減活動（工数＝標準作業時間）では改善しない側面をとらえ、分業のしくみによってしか改善できない生産性を表わす時間的指標。

期間生産性指標

納期生産性　一定期間内の受注オーダへの納期の（オーダ）平均値（の逆数）

在庫期間生産性　１日あたりの倉庫通過量

（当該品が倉庫に滞留する平均数量／平均滞在日数）

変更期間生産性　一日あたりの計画変更平均進捗度

（計画変更総件数／計画変更事務日数合計）

（3）時間生産性をアップさせる基礎技術

時間量アプローチ

速度、稼働率、平準化

期間アプローチ

同期化、短行列化、バイパス化、パラレル化、雁行化、フロー化

・短行列化　各仕事の入り口に発生する待ち行列を短くする仕組みを導入する。

・バイパス化　一連の必要な作業の一部をコンピュータ化するなどして、処理時間を短くする。担当者にとっては、その仕事をやらずに済むのでバイパスしたことになる。

・雁行化　パイプライン化。一連の作業を、順にやるのではなく、後工程の可能な部分はどんどん開始する。

・フロー化　中間にある在庫を廃止する。それでも動く仕組みにする。

（4）田中氏が指摘している問題

・ 処理プロセスに内蔵されているコントロールの仕組みがあり、その性能評価を行う必要がある。指標が必要だ。期間生産性→その意味ずけ、特徴、計測可能性、計測方法を用意する。

・時間生産性の向上の着眼点

コミュニケーションの高度化

伝統的合理化技術の再開発

経営システムの再構築

について、経営システム（ビジネスプロセス＋コントロールシステム）の再構築が問

題だ。これまでの高度情報化システムの導入が役にたたなかったとすれば、何が悪かったのか。「情報の共有」ができたとしても、それがマクロ経営システムのコントロールに役立ち全体最適を達成するのでなければならない。管理システムの何をつかまえてどのように使うかの仕組み作りが問題だ。

BPR（ビジネスプロセス・リエンジニアリング：ハマー&チャンピー）という言葉があった。「ビジネスプロセスを劇的に改善する」というものだが、それを批判して、定義のなかに程度問題を語る形容詞が入っているのは、学問的定義としては物足りないことが指摘されている。

情報の有効利用の基本は、フィードバック、適応、自己組織化である。伝統的管理会計の手法である予算管理では、製造原価予算や販売予算などの形式を使って活動計画を立て、期中の自部署の活動が全社的に整合しているか、また、目標額からずれた細目を早期発見して何らかの問題を早期発見し対処するというようにフィードバックしてきた（図 5-14）。

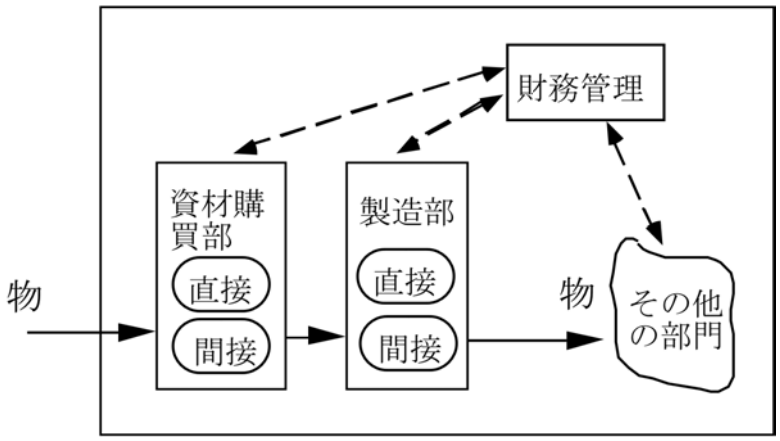


図 5-14　原価情報のフィードバックや削減指示

図 5-14 よりも詳しくビジネスプロセスの内容を記述するアクティビティ・インタラクション・ダイアグラム(AID)では、動作を決定づける情報が、ファイル全体と活動の現状のデータ全体である（図 5-18）。

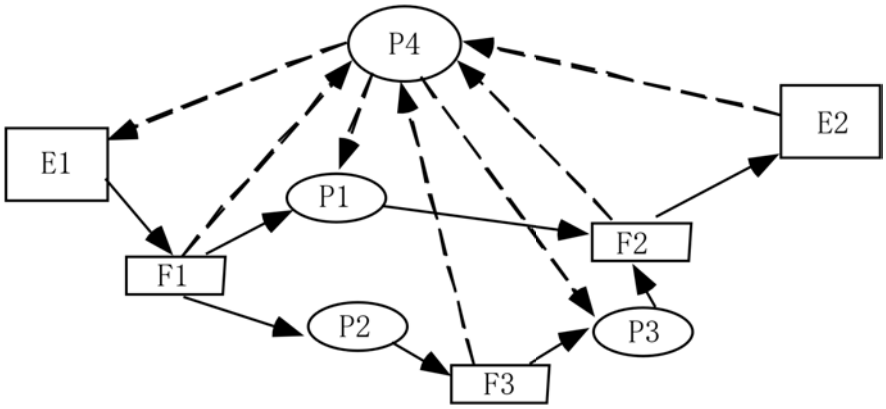


図 5-18　　AID のフィードバック

だから、ビジネスプロセスの動作を変更するには、それらの情報を使って、各活動にフィードバックする。どこかでビジネスの状況が変化したことはファイルに書き込まれたデータから分かるから、そのデータを利用して、ビジネスのやり方を適応的に変えればよい。これが情報を「共有する」ことだ。その場合の問題設定は、どういうデータをもとにして、どのように行動を変化すれば、全社的な性能がよくなるかを設計することである。

5.5　MRP

素早く反応するビジネスプロセスをつくるには、システム内に存在するいろいろなモノや要求の在庫を減らす必要がある。そのためには、あらかじめ稼働の計画を立てて実行していくことが考えられる。計画は、何を、いつ、誰が行うのかを指定する。

生産ビジネスプロセスの生産計画方法として、MRP （materials requirement planning：資材所要量計画）が発展した。MRP の考え方の基本部分は合理的であって、生産に限らず広くビジネスプロセスに適用可能であると期待される。まず、MRP について説明し、その後 MRP の欠点や別の計画方法を示す。

製品の生産は、図 5-16 のような構造を持っているプロセスである。

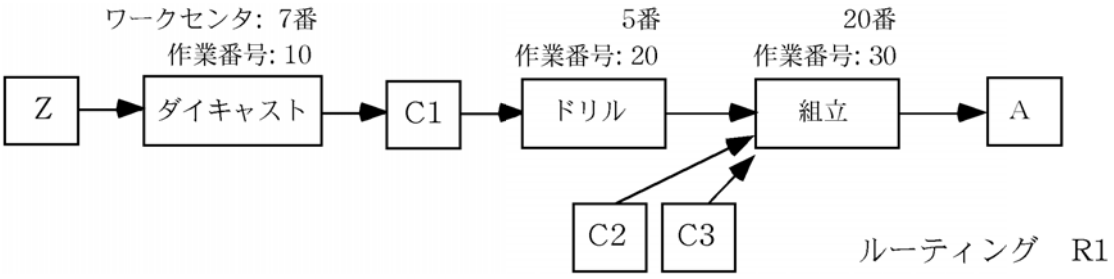


図 5-16　部品を加工し組立する作業手順

製品 A は､部品 C1, C2, C3 からドリルによる穴開けや組立によって作られる｡一方、C1 はアルミ材料 Z を溶かして鋳型で精密鋳造して作られる。このように、製品は部品や原材料を加工して物理的に変化させたり組み立てたりする作業を通じて生産される。イスや机、パソコンや自動車は同じような生産の構造を持ち離散組立型生産と呼ばれている。他方、ラーメンの麺や医薬品原料やプラスチック原料の生産のような化学プロセス型生産もあって、組立のかわりに連続的プロセスで作られるが、各原料の配合比率を離散生産の部品のように考えることで、同様の構造で考えることができる。メーカは基本となる製品を基にして色や材質を変えたり装備や高機能部品を使用するなどして、多くのバリエーションを持っている。さらに、ひとつの製品だけを扱っているメーカというものはなく、非常に多くの種類の製品を扱っている。

生産のビジネスプロセスがこのように複雑なので、生産計画方法の発展には歴史的にいくつかの段階があった。

まず、伝統的な在庫管理の方法がある。需要・販売量や在庫量は時間とともにランダムと思われるふるまいを示す。このために、統計的な側面に注目して平均的な特性を管理しようとしたものである。発注点方式、定期発注方式、補充点方式、ダブルビン方式などの在庫管理・発注方式を、ABC 分析などによってわかる品目の重要度に応じて使い分けるものである。このようなランダムな在庫変動のモデルは、実務的に簡単に使ってみることができる上に、数学や統計学の応用領域としても興味深いものがあったので種々発展した。安全在庫や経済的最適発注量の概念等は、たとえ直接に使えない場面であっても、考えるための基本モデルを提供している。

伝統的在庫管理論の考え方の背景として、主に２つを指摘できる[鳥羽]。

（１）材料・部材や製品の何万という在庫品目はすべてが互いに独立の需要を持ち，独立に発注できると考えている。

（２）年間を通じた平均使用量や平均使用速度という量を使えるというような，比較的単純なビジネス環境を想定している。

工業製品は多くの部品から成る．たとえば，自動車１台は 20,000ー30,000 点の部品からなると言われている。すると，最終製品の需要が決まると，それを構成する部品の数はその需要に従属して計算によって定まるから，独立に部品を予測して発注したら部品不足や過剰在庫になるのは目に見えている。つまり、すべての材料や中間製品が独立に使用されるという仮定はおかしい。また、今どき年間の需要が一定している製品は珍しいのではないだろうか。ケータイやパソコンは一度売れ残ったらもはや誰も引取らないぐらい陳腐化のスピードも速い。すると，平均した年間使用量を仮定し

て考えるよりは，もっと直接に販売の状況を反映した生産の計画を立て実施し，さらに実際の需要の一部が確定したところで再計画しては生産することをくり返すのが合理的だろう．つまり、あまり主観の入らない生産計画と販売実績による徹底した再計画の仕組みを作る必要がある。ただし，多品種少量生産を行うためには，手計算では間に合わない。

「伝統的」方法という名称は古くて使えないことではまったくない。原材料や中間部品や製品は、重点的に管理すべき品目もあれば、手抜き管理でラフにやればいいものもある。手抜き管理を実践するのには、伝統的在庫管理法の中の適切で簡便な方法がある。たとえば、ダブルビン方式では在庫品に対して２つの容器を用意し片方がカラになったら容器分の量だけを発注する。また、ねじや、オフィスで使うクリップなどの間接材と呼ばれるものの多くも簡便な管理方法が望まれる。オフィスにおけるパソコンのような間接材（MRO 材）でも購買情報の集中による管理をすればコスト削減の余地があるが管理の手間とのバランスが必要となる。

製品がある程度以上の物理的複雑性をもつとき、図 5-16 のように、各作業が必要とする部品を、必要なときに、必要な数量だけ用意することで生産が滞りなく行われる。しかもこの例よりも複数段階の構造を持つ製品がたくさんある。部品がひとつでも足りなければ作業は順番通りに進まないので、うまくやらないと生産プロセスの至る所で不急部品の過剰在庫と必要部品の在庫不足が頻発することになる。したがって、計画が必要になる。

MRP の目的は、購買や製造に関わる原材料と部品の手配計画を計算して作ることである。この意味で MRP 計算とも呼ばれる。

5.6　MRP 計算

5.6.1　PDM：製品データ管理

MRP 計算に使用される情報は 4 種類ある。部品構成表（BOM - bill of materials)、作業手順（routing)、基準生産計画、部品在庫データである。

（１）部品表

部品構成表は製品や部品の物理的な構成を示す。部品表ともボムとも呼ばれる。化学プロセスによる製品の場合は配合表(recipe)という。パソコンはマザーボードとハードディスクとボディなどの部品で構成され、さらにマザーボードは 2 つの CPU と 4 つのメモリと基盤 1 枚からなる、というような組立に必要な部品構成が表現される。

図 5-16 のような製造工程の製品 A の部品表は図 5-17 で表わされる。ローレベルコードは、顧客に近いほうを 0 としてつけた階層の番号である。図 5-17 では A が 0 であ

り、C1, C2, C3 が 1 であり、Z のローレベルコードが 2 である。企業が扱う製品や部品は多種類あり、また、ひとつの部品が異なるレベルで同一製品の BOM に現れることもある。ある部品や材料のローレベルコード値として、最も大きな値のローレベルコードがつけられる。製品データが大量になると正しいローレベルコード付けが人間では間違いなくできないので BOM プロセッサーというプログラムで行う。ローレベルコードは、MRP 計算の部品展開・在庫割当て計算法であるレベル・バイ・レベルアルゴリズムで用いるために付けられている。

部品表の末端の C2, C3, Z は、必ず購買品目である。

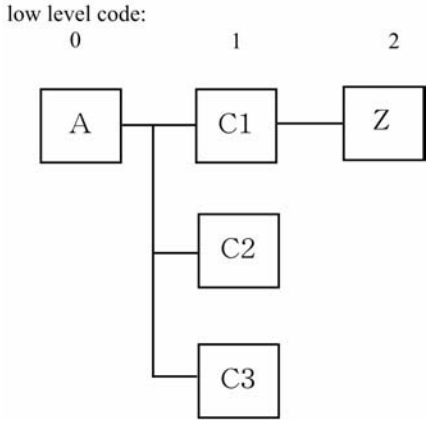


図 5-17　製品 A の BOM

図のままだとコンピュータでの計算に使えないので、情報システムではこの図の情報を 2 つの表で図 5-18 のように表わす。品目#, 親品目などの値は、品目に付けられたデータベースの主キー値である。識別のために、部品名や商品名は重複する可能性があるので必ず人為的に番号をつけてその品目の主キーとして用いる。構成数とは、親部品をひとつ作るのに子部品が何個使われているかを示す。一枚の鋼板を 1/3 にして使用するときは 0.333（個）、親品目 1 個当たり 1.7 リットルの塗料を使うときは 1.7（リットル）という構成数になる。

品目表

品目＃	品目名	ローレベル
1001	A	0
1002	C1	1
1003	C2	1
1004	C3	1
1005	Z	2

部品構成表

構成＃	親品目	子品目	構成数
501	1001	1002	4
502	1001	1003	1
503	1001	1004	1
504	1002	1005	0.5

図 5-18　BOM のテーブルによる表現

（2）作業手順

作業手順によって、子部品から親部品を製造する作業群を指定する。イメージとしては部品表の各レベルの間に存在する。図 5-16 は製品 A が、材料 Z や部品 C1 などからどのようにして作られるかのイメージである。作業手順としては２つある。品目 A の作業手順がドリルと組立の２つの作業からなる。また、品目 C1 の作業手順はダイキャストという作業だけである。ドリルはボール盤などの機械で行うことができるが、異なる年式のボール盤が複数あるときや組立ラインなども機械と同様に扱って計画計算に便利なように、もう少し一般に、製造作業から見て類似した機械設備や人の集まりをワークセンタと呼んでとらえる。ワークセンタの一覧表（マスタ）によって、工場内のすべての機械を記述し、どの製品のどの作業には製品 1 個当たりどれぐらい時間がかかるかを記録して保持する。当然、作業改善が進むと、作業の標準値は変化する。また、製造単位数（ロット）が異なると、原料の梱包サイズが異なったものを使ったり異なる機械を使用したりするので、ひとつの品目に対して複数の BOM や作業手順がある。オフィスでの会議資料作成でも、3 枚コピーするときと、200 枚コピーするときでは使うコピー機が異なることがある。100 枚の宛名書きは一人で手書きも可能だが、1 日で５万枚となると一人で手書きは不可能であって別の体制を取る必要がある。

以上の BOM と作業手順が PDM のデータの一部である。製品の設計結果は製造用の PDM データとして保持される。

品目には需要の側面からみて 2 種類ある。他の品目と無関係に需要があるのが独立需要品目である。最終製品や修理用サービスパーツなどがこれにあたる。需要が予測や注文で決まる、あるいは決める。 従属品目は，他の品目の需要からその数量に基づいて需要が決まる品目である。部品や原材料である．独立需要品目の生産計画を基準生産計画という。

在庫データは各品目ごとの在庫量を保持したものである。これには、すでにリリース済みであって、たとえば 2 週間後に 30 個と 5 週間後に 6 個入庫する、というような発注残も含まれる。なお、計画をリリースするとは、サプライヤへの購買注文を実際に出すことや、製造現場に、各ワークセンターごと（と各作業者ごと）に製造指図書を実際に出すことである。後者は黒板のような板の上の各作業者ごとの所定の場所に、作業内容を書いた指図書を差し立てたりしたので「差立て（dispatch)」と呼ばれることもある。いずれも、将来の時点での入庫があるので、在庫データとして扱う必要がある。

5.6.2　MRP 計算（固定リードタイム計画の場合）

MRP による計画はタイムバケットという一定期間ごとの製造数量を計画していく。バケットとはバケツのことで、連続時間ではなくタイムバケットごとにまとめて離散化して製品や部品や原材料購買を計画する。タイムバケットの単位として週がよく用いられる。ひと月や日や旬（10 日間）をタイムバケットの単位として使うこともある。

MRP の計算法には大きく分けて 2 種類ある。作業手順の情報を用いないものと用いるものである。作業手順の情報を持ちいないで、各部品について予め設定した所要日数を考慮して計画を立てる方法を概説する。

図 5-17 の製品 A の BOM が与えられているとする。作業手順ほどには詳細に見ないで、品目 A, C1 の作業時間は通常の生産量の場合に、それぞれ 1 週間と 2 週間だとする。同様に、C2, C3, Z には購買を発注してから納品までの時間がかかるので、それぞれ図 19 のかっこの数字のように時間が必要だとする。いずれも、開始してから終了するまでの時間であり、リードタイムと呼ばれる。製造のばあいは製造リートタイム、購買の場合は購買リードタイムである。また、図には C1, Z のそれぞれの構成数も示した。構成数 1 は省略してある。

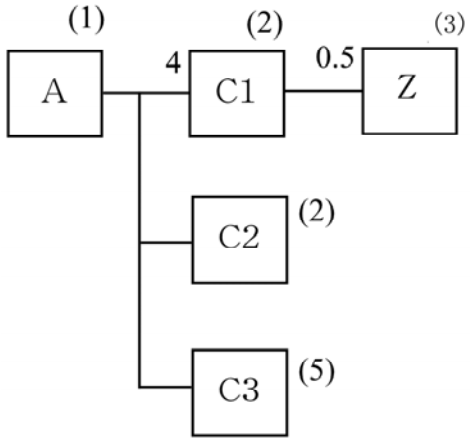


図 5-19　リードタイムが設定された BOM

独立重要品目の生産計画を基準生産計画（MPS - master production schedule)と呼ぶ。いま、製品 A の MPS が次のとおりであるとする。

タイムバケット	1	2	3	4	5	6	7
総　所　要　量			10		8		5

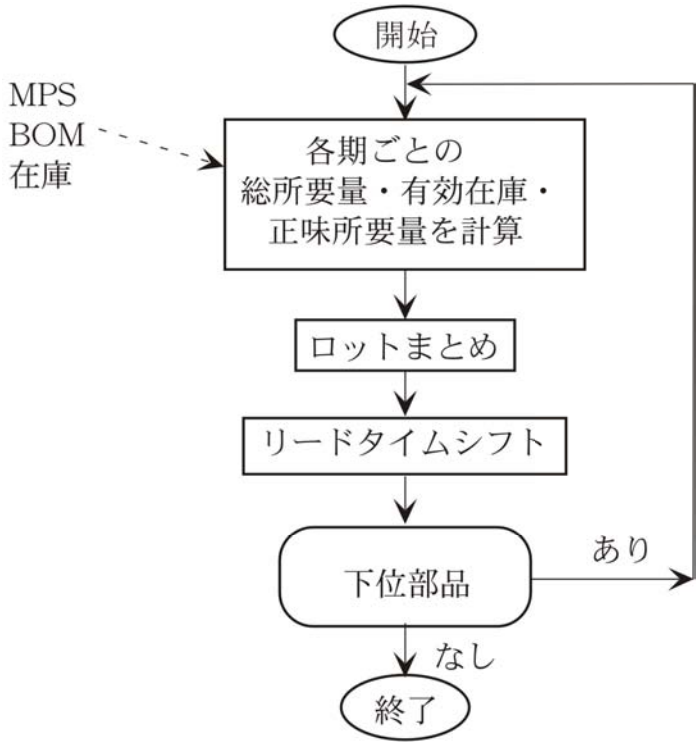


図 5-20　MRP 計算の手順

MRP 計算の手順（固定リードタイム計画）

Part 1: タイムバケットの各期ごとに総所要量と有効在庫を計算。

(1) 総所要量計算。各タイムバケットについて生産計画から独立需要品目や従属需要品目の総所要量計算を行う。独立需要品目の総所要量は基準生産計画ＭＰＳそのものである。共通部品であるような従属需要品目の場合の総所要量は，各 BOM の構成数から計算される量を，複数の独立需要品目に対応して合計した量になる。たとえば，タイムバケットの時点 n における C1 の所要量は，構成数が 4 だから n における A の所要量の 4 倍になる。つまり、子の総所要量(n)＝親の正味所要量(n)×構成数。

(2) 有効在庫の計算

各タイムバケット n ごとに計算する。

有効在庫(n)＝有効在庫(n-1)＋オーダ残(n)-総所要量(n)

なお、上でオーダ残とは既に製造現場に生産命令を発注済みであって、ある時点で生産が終わって利用可能になるものである。サプライヤーへの購買オーダについても発生しうる。

Part 2:

すべてのタイムバケットについて有効在庫がもとまった状況で以下の計算を進めていく。

(3) 正味所要量の計算

余分には生産しないことが前提なので，有効在庫(n)が負の値となっているときにだけ，その絶対値の分を正味所要量(n)として生産することにする。

(4) ロット編成（ロットまとめ）

ロット(lot)とは，原材料のような購買品については発注する単位（個数）であり，自社工場で製造する内作品については製造単位個数である。

正味所要量に対して生産の単位数であるロットサイズを決める。正味所要量をそのままロットサイズとする方法(都度ロット(lot for lot, L4L)という), 20 個づつ製造する等の固定数の倍数にする固定ロット，定期発注（2 週分とか 3 週分とか)、経済的最適ロット数など，いくつかの方法が提案されている。

ロット編成を終えると，各期に対する完了日順の計画オーダ（生産命令計画）ができあがる。

(5) 先行化（リードタイム・シフト）

完了日順の計画オーダとは，そのタイムバケットの期首に（タイムバケットを週にとっている場合では週のはじめに）生産または納入が完了していなければならない量を表わしている。よってリードタイム分だけ先行させて製造や発注を開始しなければならない。リードタイム以上に余分に先行させると過剰在庫となるのでちょうどリードタイム分だけにする。製品 A の場合は，ちょうど 1 週分ずらせばよい。

以上の Part 1, 2 からなる MRP 計算が終わると、生産開始を担当部署に指示するための生産計画である着手日順計画オーダの計算ができあがる。

製品 A の MRP 計算（空欄は 0 を表わす）

製品 A の条件：リードタイム 1，都度ロット，初期在庫 10，オーダ残が 3 期に 5 個。

タイムバケット	1	2	3	4	5	6	7
総所要量			10		8		5
オーダ残			5				
（有効在庫）	10	10	5	5	-3		-5
正味所要量					3		5
完了日順計画オーダ					3		5
着手日順計画オーダ				3		5	

部品 C1 の MRP 計算

リードタイム 2、固定ロット 10、初期在庫 0、オーダ残が 2 期に 3 個。

タイムバケット	1	2	3	4	5	6	7
総所要量				12		20	
オーダ残		3					
（有効在庫）		3	3	-9		-20	
正味所要量				9		20	
完了日順計画オーダ				10		20	
着手日順計画オーダ		10		20			

部品 C1 が他の品目の部品としても計画されている場合は、上の総所要量に追加されて計画される。つまり、MRP は共通部品をすべてまとめて生産計画を立てる。

固定リードタイム計画の MRP 計算の別法がいくつかある。固定ロット生産計画の場合、固定ロットで製造したときの余分の部品を明示的に扱いたい時、次のようにすることもできる。

各タイムバケットついて次の(1), (2), (3) を計算する。

(1)総所要量計算(gross requirements)。

(2)オーダ残(n)

(3)予想在庫バランスの計算

在庫(n-1)＋オーダ残(n)-総所要量(n)

を計算し、値がプラスの時はそのまま書き込む。

マイナスになるときには、生産することにする。そのためには、着手日順計画オーダを、リードタイム分だけずらして書き込む。同時に、そのオーダが計画通り終了するものと見なして計算する。（例では、第 4 期、第 6 期。）

リードタイム 2、固定ロット 10、初期在庫 0、オーダ残が 2 期に 3 個。

タイムバケット		1	2	3	4	5	6	7
総所要量					12		20	
オーダ残			3					
予想在庫バランス	0		3	3	1	1	1	1
着手日順計画オーダ			10		20			

さて、日程シフト計画では生産量の変動に関わらずに製造時間は一定として計画する。したがって余裕時間が計画に組み込まれる。さらに、余裕は、前工程や調達の遅

れとか自工程の故障や不良による遅れを吸収するためのバッファの働きを持つ。これらの機能の他にも、ERP パッケージには人間との共同作業を円滑にするために、製品グループの設定や品目ごとの計画余裕設定などの機能や販売計画からの独立所要量の自動計算などの機能が利用できる。製品グループとは、香りの違うシャンプーのバリエーション製品 A1, A2, A3 があるとき製品グループ AG としてまとめて考え、総和が１になるような比率を（たとえば A1 が 0.5、A2 が 0.3、A3 が 0.2）定義し、そのグループに対する総生産計画を実行するとき、自動的に比率にしたがった MRP 計算（BOM 展開計算）が出来上がるものである。たとえば、ある期に製品グループ AG を 4000 個生産するということは、A1 は 2000 個、A2 を 1200 個、A3 を 800 個生産する計画を意味する。

問 5-4

BOM 図 5-9 の材料 Z は購買リードタイムが 3 期である。Z についての MRP 計算を、予想在庫バランスの考え方を使って計算しなさい。ただし、初期在庫 5、オーダ残が 1 期に 5、ロットフォーロットとする。

5.6.3　レベル・バイ・レベル

MRP では共通部品の生産計画を作成する時にある注意が必要である。さらに、その問題を解決して計画時にうまく在庫引き当て（在庫使用の予約）を行う計算法として、ローレベルコードを使ったレベル・バイ・レベル計算がある。鳥羽著（1995）を参考にして説明する。図 5-21 の BOM を持つ２つの製品 A, B があるとする。C は共通品で構成数は 1 とする。

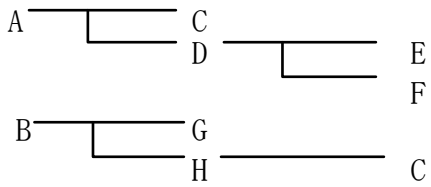


図 5-21　共通部品 C を使う２つの製品 BOM

A, B からの所要量があるとき、C の MRP 計算を示す。C の在庫は 60 個で、1 期にオーダー残が 20 個あるとする。

（１）品目主キー番号順に A, H の順に部品表展開計算をして在庫を割り当てたときの

共通部品 C の MRP 計算

A からの正味所要量に対する計算

タイムバケット	1	2	3	4	5	6	7
総所要量				40	30	20	
オーダ残	10						
（有効在庫）	70	70	70	30		-20	
正味所要量						20	

H からの正味所要量に対する計算

タイムバケット	1	2	3	4	5	6	7
総所要量		50	50		30		
オーダ残							
（有効在庫）		-50	-50		-30		
正味所要量		50	50		30		

この計算では在庫 60 個を長い間もっていることがありうるので好ましくない。つまり、早い納期を持つ H の生産に在庫を早く引き当てるべきである。したがって上のような A, H のための MRP 計算はまずい。

（2）ローレベル・コードごとに在庫を引き当てたときの C の MRP 計算

A の正味所要量と有効在庫計算

タイムバケット	1	2	3	4	5	6	7
総所要量				40	30	20	
オーダ残							
（有効在庫）				-40	-30	-20	
正味所要量				40	30	20	

H からの正味所要量に対する計算

タイムバケット	1	2	3	4	5	6	7
総所要量		50	50		30		
オーダ残	10						
（有効在庫）	70	20	-30		-30		
正味所要量			30		30		

このように製造オーダと在庫を納期順に引き当てる部品表展開法がレベル・バイ・レベル計算法である（具体的計算法は鳥羽著参照）。

問 5-5

図 5-21 の２つの製品の BOM を図 5-18 のような２つの表のデータとして表現しなさい。

5.6.4　MRP 計画の更新

MRP によって計画を立てた後は、実績を反映しながら常に計画を更新して「最適」的な管理を行う。時間が経って、たとえば 1 週間ごとに再計画をする場合、１週間のうちに計画と異なる販売実績が発生するので、計画時とは異なる状況になる。もちろん不良品が出たりラインがストップしたときも計画と異なる状況になる。

計画を更新するための方法には、再計画法と正味変更計画法がある。さらに、正味変更計画には、計算の手間と計算結果の正確さのトレードオフによって、いくつかのやり方がある。鳥羽著（1995）を参考に、下記に説明する。

（１）BOM

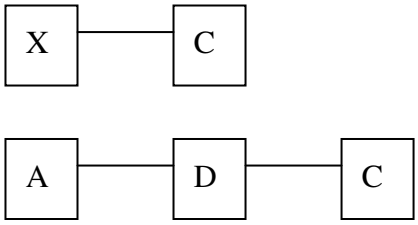


図 5-22　共通部品 C を含む BOM

（２）部品 C の当初計画

リードタイム１，都度ロット，初期在庫 0，オーダ残が 1 期に 10 個。

タイムバケット	1	2	3	4	5	6
X からの総所要量			10	10	10	
D からの総所要量	10	10	10			
オーダ残	10					
（有効在庫）	0	-10	-20	-10	-10	
正味所要量		10	20	10	10	
完了日順計画オーダ		10	20	10	10	
着手日順計画オーダ	10	20	10	10		

図 7-23　部品Ｃの当初計画

（3）正味変更計画

変動する可能性のあるものは

総所要量（X, D それぞれから)、在庫、オーダー残

である。

1 期の計画を実行中に以下の（ア）ー（ウ）の事態が生じた場合を考える。

（ア）X の追加注文があり、総所要量がタイムバケット 5 に 5 個増えた。

（イ）D のタイムバケット 1 の 10 個の製造が 4 個の遅れを出しして当期に間に合わない。その 4 個は次の期に完成する。

（ウ）1 期には入るはずの 10 個のオーダ残による納品の一部が、上の（イ）の遅れによって 4 個は 1 期末の在庫となることになった。

このような状況で、1 期末に MRP 再計画を計算する。

品目 C の変動部分だけをあらわすと次のようになっている。

(ウ)

（2 期の）初期在庫 4　　（イ）　　（ア）

タイムバケット	1	2	3	4	5	6
X からの総所要量					5	
D からの総所要量		4				
オーダ残						
（有効在庫）						
正味所要量						
完了日順計画オーダ						
着手日順計画オーダ						

図 5-24　部品 C の変動

正味変更計算前の総所要量

タイムバケット	1	2	3	4	5	6
X からの総所要量			10	10	15	
D からの総所要量		14	10			
オーダ残						
（有効在庫）	(4)	-10	-20	-10	-15	
正味所要量		10	20	10	15	

図 5-25　部品 C の総所要量

1 期末における正味変更計算の結果

タイムバケット	1	2	3	4	5	6
X からの総所要量			10	10	15	
D からの総所要量		14	10			
オーダ残						
（有効在庫）		-10	-20	-10	-15	
正味所要量		10	20	10	15	
完了日順計画オーダ		10	20	10	15	
着手日順計画オーダ		30	10	15		

図 5-26　部品 C の正味変更計画結果

計算を行っている時点が第 1 期末なので、第 1 期に製造する計画を作ることはできない。そのため、第 2 期の生産量が 30 個になっている。

問 5-6

図 6-26 の MRP 計算結果では、カレンダー第 2 期に着手する計画の 30 個は第 2 期の総所要量には応えることができない。この着手日順計画と整合するような品目 D の総所要量はどのようになるか。D の総所要量を変更して、着手日順生産計画と整合するような 2 期と 3 期の値を示せ。（結局 D の納期を変える必要があるということなので、顧客と交渉しなければならない。または、第 2 期の前半に残業を行って特急増産する方策が可能な場合もある。）

5.6.5　MRP 計算（作業時間シフト計画）

作業手順の情報を用いる作業時間シフト計画の計算では、あらかじめ設定した 1 個当たりの標準作業時間をもとにして、生産する数量に応じて作業ごとに総作業時間を計算し、完了日順計画オーダからその時間分を日程に直した分をシフトして着手日順計画オーダを計算する。

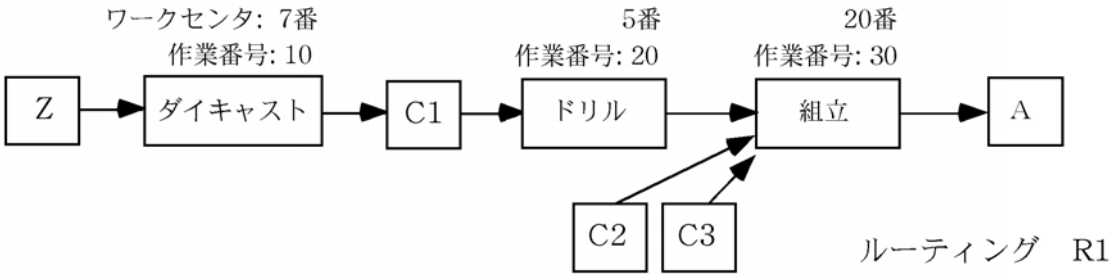


図 5-16（再掲）　部品を加工し組立する作業手順

BOM において階層があることは、いくつかの部品から作業手順で指定されたいくつかの作業によってその部品から別の部品が作られることを意味する。たとえば、図 5-16 では、C1, C2, C3 からドリルによる穴あけ作業と組立作業によって部品（製品）A が作られることになる。したがって、たとえば A を 200 個必要な場合には、1 個あたりに必要な加工時間の 200 倍の時間が費やされるので、工場やオフィスの稼働日カレンダーに照らしてその分の時間をシフトして着手日と開始時刻を決める。

作業時間としては、製品や部品ごとに用いる作業手順を構成する各作業ごとに、段取り時間、加工時間、移動時間、片づけ時間の標準時間を設定する。必要な作業時間のことを、能力、能力所要量、工数とも呼ぶ。

5.6.6 MRP の特徴

MRP で作られる部品の計画オーダーはあくまでコンピュータ内部の名目的計画として保持されるが，実際の生産オーダや購買発注オーダは将来の定められた期までの分しかリリースされない。たとえば 2 週先までの分しか出さないとすると，現時点から第 3 週以降での計画変更や需要変動の影響を計算処理だけですませられる。したがって、需要の変動に対してかなりの範囲で適応でき、製品在庫の管理が精密になる。

もしずっと先のオーダまで製造現場に知らせると，各工程の責任者は遅れや不良による欠品を防ぐために早めに開始しがちで，結果として工場全体としては過剰在庫をかかえ，その過剰分も製造するための時間がまた遅れをよび，この状況がくり返され増幅するということが起こりがちになる。

MRP は直感的な意味でのジャストインタイム を実現するために，また計画と実行結果の差に基づいて再計画を行うためにも，必要な時点で必要なオーダしか出さないことで，生産のビジネスプロセス全体をできるだけ在庫なしで同期化させようとするものである。独立，従属需要品目，タイムバケットごとの管理ということと，情報を

集中してから各活動にフィードバックすることが制御方法の特色となっている。

製造リードタイムは次のような成分からなる。

製造リードタイム＝待ち行列時間＋段取り時間＋加工時間＋移動時間＋片づけ時間

ここで待ち行列時間とは、加工すべきオーダが混んできたためにすべての材料がそろっていても、加工開始を待つ必要があるときの待ち時間である。待ち時間は製造リードタイムの大きな部分をしめるといわれる。待ち時間はダイナミックに変化するものだが MRP ではこれを一定時間として扱うので正確ではない。そのため、計画作成時に作業前後に余裕時間を設定するとか安全在庫数量を所要量に追加するなどの工夫によって、待ち行列や不測の事態に備えようとする。

一定リードタイム計画では製造数量に関わりなく一定の時間で製造が終わるかのごとくに計算した。加工時間に較べて加工精度調整のための段取り時間が長いとか、待ち行列時間が長いというときに、それを計画作成上の余裕時間として見込んだり、リードタイムに含めたりして計画の実行可能性を高める。ただし待ち行列時間を論理的に突き詰めるのは、離散事象システムとして製造プロセスを見て需要（受注）と突きあわせる必要がある。また、製造の現場が計画値通りに稼働するには、人間がからむことであるから現場管理や作業改善・管理が浸透していることが重要となる。

5.7 ERP と APS とプロジェクト日程計画

MRP の基本機能は資材所要量計画であって、製品や中間部品の手配計画と在庫管理手法としての意味を持っている。この計算に続いて、各機械設備の時間あたり標準原価があれば、MRP の結果から得られるワークセンタごとの稼働計画時間に掛け合わせることで、ワークセンタごとの直接原価を計算できる。また、同様に、MRP の結果の購買計画について材料費の原価データを掛け合わせたりすることで、直接材料費が計算できる。また、原価情報とは別に、MRP の結果から計画実行に伴う各タイムバケットごと・各機械設備ごとの所要時間が計算できる。たとえば、機械が 1 台で 8 時間×5 日＝40 時間稼働可能という状況で、MRP 計算上 60 時間必要ならば、この機械に対する負荷率は 150%であり、残業か人材派遣等の外注の手当てや生産計画見直しなどがなされる必要があることが分かる。このように負荷状況を検討し生産計画を変更・修正し実施可能な計画にすることを能力計画と呼ぶ。

元来の部品手配計画の MRP に引き続いてこのような計画管理をおこなえる MRP は MRP II (management resouce planning)とか、ERP (enterprise resouce planning package, 統合基幹情報システム)と呼ばれている。製造原価などの管理会計や生産諸要素全般の計画

が可能で、経営管理システムとしての MRP であり、企業統合情報システム-組織の個々の業務を一貫して協調管理する統合システムという意味が込められている。

MRP II 計算における能力とは利用可能時間である。能力計画の特徴を次の製品Xで説明する。

製品 X の MRP 計算								
タイムバケット	1	2	3	4	5	6	7	8
生産計画オーダ	100	100	200	200	200	0	100	0

製品Xを 1 個作るために段取り時間などは不要で加工時間だけが 3 分必要だとする。すると各期には下の通りの所要能力となる。

製品 X の MRP 計算								
タイムバケット	1	2	3	4	5	6	7	8
生産計画オーダ	300	300	600	600	600	0	300	0

もし、Xを製造するのに使われる機械が 1 期あたり 450 分であるとすると、第 3,4,5 期の残業の可能性を考えるとか、製造量を再考するとか、外注を考えるとかの必要がある。さらに、製品X以外にY、Z、Tなどが同じ機械や組立ラインを使う場合にも、それらを合算して考えた上で調整が必要だ。

このように通常の MRP 計算では能力使用状況は、生産量を決めた後で事後的にしかわからない。生産量を決めるときに同時に能力が 100%以内に収まるように調整することが考えられる。これを行うということは、タイムバケットごとの計画を作るというよりは、機械や設備の使用時刻まで作成するというスケジューリングを行うことになる。ここでスケジューリングとは、各ワークセンタ（生産資源）ごとに、ガントチャートと呼ばれる図的技法で表わされるような、詳細な生産活動の開始終了時刻を決めたものである。当然、作業手順の順序の制約や作業負荷が 100%に収まっていることは考慮されて決められる。ただし、スケジューリング計算に当たっては、多くの納期を持つ多種類の製品製造のための作業の組み合わせの数が多すぎるので、すべての場合を考えてできるだけ最適なスケジュールを得るためには計算時間が天文学的数字となるために不可能である。いろいろな経験則を用いて、実践的な意味で満足の行くスケジュールを作成する。市販の ERP パッケージでは、能力状況の表示以外にも、能力グループ階層を設定することによって工場全体の概算能力計画が可能にするとか、能力の負荷率が 50%を超えたら自動的に時期に製造量をずらすなどの種々の便利な機能を備えている場合が多い。

MRP ではまずタイムバケットごとの製造計画を決め、それを実施するスケジュール

はその後でいろいろな工夫をしながら考えだされる。APS（advanced planning and scheduling)と呼ばれる生産計画作成ソフトウエアでは、この計画からスケジューリングという順序をやめて、いきなりスケジューリングして実行可能なスケジュールを作りそれを単にタイムバケットごとに集計して生産計画とするという方法で、スケジューリングと計画を融合させている。MRP と同様のボム展開計算や作業手順ファイルへのアクセスと組み合わせ計算を同時にメモリ上で行わないと計算時間が非常にかかるため、巨大なメモリを必要とすることが多く、近年の計算機構の発展によって、はじめて可能になってきたソフトウエアである。

スケジューリングとは与えられた製造オーダ群に対して、各オーダの作業手順を構成する作業の全体の実行時期を定めることであって、ガント・チャートがスケジュールの表現のために便利に用いられる。これは、各ワークセンタごとに作業の実行順序と時刻を示した図である。製造オーダをジョブ、ワークセンタをショップと呼んでいることもある。こなすべき製造オーダ群と作業手順群が与えられたとき、ワークセンターごとに作業手順を構成する作業の実行順序（優先順序）を決める必要がある。ある観点からすると、空のガント・チャートに、何らかの方針や評価を用いて、適切な計算法によって作業を配置していくことにほかならない。たとえば、

FCFS (first-come-first-served): 到着の早いオーダから配置する。FIFO ともいう。
SPT (shortest processing time): 作業時間の小さい品目の作業から割り付ける。
EDD (earliest due date): 納期の早い品目から割り付ける。

などがある。簡単で基本的ケースとして、1 品目を 1 つの機械で加工する製造オーダがたくさんあり段取り時間を省略して考えられるときは、SPT によるスケジューリングが、各製造オーダ全体が実行開始可能になってから終了するまでに必要な時間の総和（総フロータイム）とか、オーダごとの納期遅れの総和などの指標において、最適なスケジュールを与えることが簡単に証明できる。

スケジューリングは組み合わせ的な問題であり、数学的に興味深い構造を持っていたので多くの研究がある。組み合わせの数が多いこと、部品表と作業手順と段取りや片づけ時間などを考慮すると組み合わせの構造が非常に複雑になることから、簡単な場合以外は、多品種少量計算に APS で適用できるほどの精密さを持ったスケジューリング計算方法はまだ利用可能でない。APS はその計算ロジックの細部が公開されていないが、近似的な計算を行っている。たとえば、実務的な「最適解」（満足解）を探索するときに、自然界で生物が環境に適応しながら適者である遺伝子が生き残っていく方法を利用する研究例が数多くある。環境への種の適応は、言い換えれば適者生存と

いうランダムな組み合わせを効率良く使う探索方法であり、遺伝的アルゴリズム（GA-genetic algorithm)とよばれる。最適性を考えるには評価軸を定める必要がある。コスト、サービス率（納期遵守率)、総フロータイム、プロセス内の平均在庫量、各種の生産性（投入量と生産量の比率)、平均稼働率などがある。どういった評価の組み合わせにすべきかについての万能な方法はないので、ケースバイケースで考える。

PDM のデータで表されるような BOM と作業手順は生産の基本条件を正確に表しているので、MRP 計算ばかりではなく APS や先進的スケジューリングにおいても用いられる。

5.8 ロジスティクス

物流は製品によっていくつもの中間製品がある場合があり、物流工程が相当異なるが、少し一般化することで、物流製品というモノとサービスを「組み立てて製造する」サービス活動から成る離散事象システムとみなすことによって，MRP のような配送計画管理システムを構築できることを示す。田中一成著「生産物流統合管理システム D/SNS」（1988、日刊工業新聞社）のアイデアに基づく、製造の前後の購買と配送のサプライチェーンを含むロジスティクスプロセス全体の計画・管理を考えるときの出発点となるモデルである。

5.8.1　物流プロセスの概要

図 5-27 と図 5-28 は、物流プロセスを離散事象プロセスとしてモデル化したビジネスプロセスである。製品は工場の製品倉庫から流通センタへ配送される。地理的条件によって決められた、あらかじめ流通センタと対応するいくつかのデポ（一時貯蔵所）がある。デポには受け持ちの営業拠点が決まっており、営業拠点へデポから配送される。

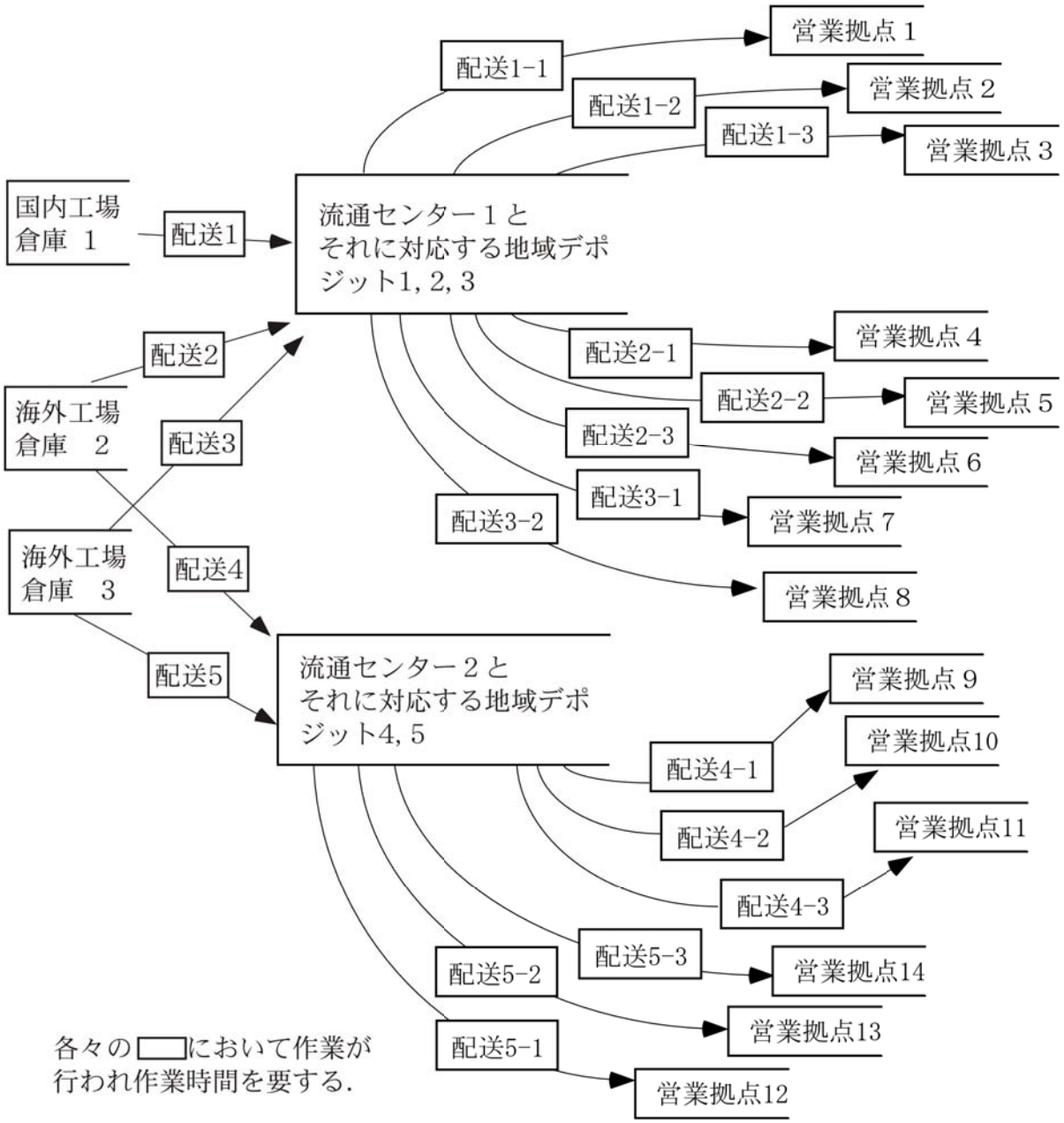


図 5-27　物流のビジネスプロセス

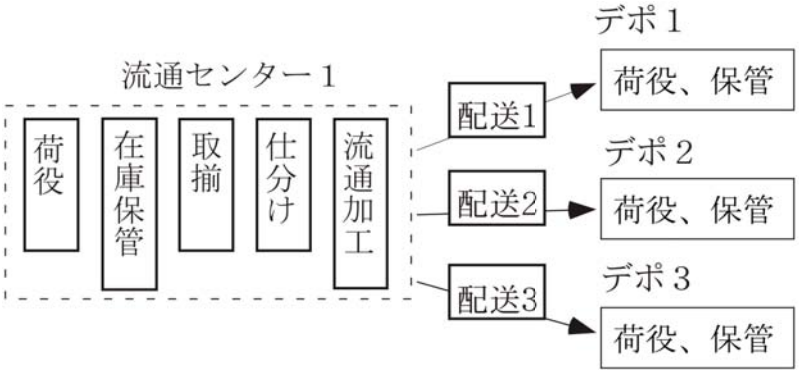


図 5-28　流通センターとデポ

流通センターもデポも、それ自体の作業のプロセスを持つ業務取引システムである。図 5-28 のような内部プロセスであるが、図 5-27 では内部のプロセスを省略してひとつの在庫と見なしている。センターとデポと営業拠点は流れ方が決まっているため、センターに投入したらあとはデポからの配送によって営業拠点に到達すると見るのである。したがって、図 5-27 のプロセスについての稼働の計画を立てる場合には、各配送という活動がリードタイムを持つとする。

物流ビジネスプロセスを考える場合は、物流サービス製品または単に物流製品という考え方をとる。図 5-27 のような物流作業手順（物流 routing）を持つ物流部品構成表（物流 BOM）を考える。

実際、顧客からの購買注文が満たされるということを「品物が顧客の手元に届けられたこと」と定義する。一般的には「品物＋存在場所」のペアによって，物流サービス製品が定まるものと考える。すると、図 5-29 のオーダ S のように、品目 A を５個注文した顧客のオーダは，物流センタを通って始めて完成されるので次のように物流部品表の階層を考える。

「工場倉庫に在る製品 A」という物流部品　　（ローレベル２）
　↓　流通工程
物流過程に在る５個まとめの製品 A という物流部品（ローレベル１）
　↓　流通工程
顧客の手元に在る５個まとめの製品 A という完成品（ローレベル０）

このように，製造部品表とのアナロジーを考えることができる。物流プロセスも製造工程も組織の仕事の仕組みであり、業務取引システムだから、当然同じような表現が可能になっている訳である。

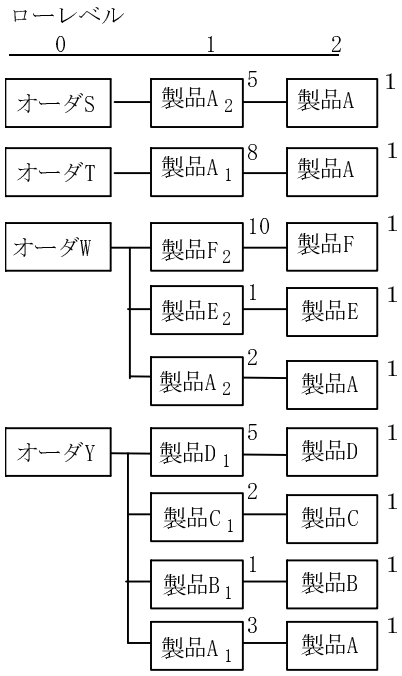


図 5-29　物流部品表

5.8.2 物流部品表の特徴

多品目を製造している生産工程の MRP 管理システムのモジュール部品表（ユニット部品表）というものがある。パソコンなどの BTO(build to order) 生産のように、メモリやマザーボードなどの、モジュール部品という共通性の高い品目の部品表だけを保持しておき，注文が定まった時点で完成品のための 1 階層程度の部品表を生成して生産計画に用いるものであった。

図 5-29 の部品表は３レベルしかない。レベル 0 が物流製品の完成品、つまり顧客の手元にある（あるいは、状況により、営業拠点にある）製品である。レベル１では流通過程にある。流通センターに対応するデポはあらかじめ決まっていて他の決定なしで「自動的に」オペレーションしているため，流通センタにある製品をレベル１と表現した物流部品表で充分である。レベル２は工場倉庫にある製品である。生産の場合の部品表と作業手順の関係と同じように、物流部品表のレベルの異なるところで、荷姿の変換といった物流加工と図 5-28 のような作業手順があるわけである。

物流計画システムの場合にはより多段階の物流があるとしても、図 5-29 のようにローレベル数は２とか３程度のものである．製造の場合は部品ごとに加工工程が異なるから多品種の品製造のための組み合わせパターンは膨大なものになる．一方，物流では配送形態によって製品をグループ分けしてものを物流部品表のレベルに対応させればよく、製品は異なっても物流パターンは変化しない。したがってパターンの数は非

常に少ないものとなり，MRP の場合のような共通部品（モジュール部品）を設定せずに，個別生産のように受注ごとに部品表を作るというやり方で計画の数は爆発しない。たとえば，（配送形態から見た）製品グループが 3 つあり，流通センターが２つ，デポがそれぞれの流通センターに対してそれぞれ７つと８つあれば，物流パターンは 3＊2＊7＋3＊2＊8＝90 通りしかない。

5.8.3　物流計画

オーダを物流製品とみなすことによって，オーダごとの個別の物流プロセスを物流部品表に登録して，それを使って物流段階ごとの日程計画を立てることができる．

物流計画の狙いは，物流在庫ゼロである。

製造の場合のワークステーションと同様に，物流過程のワークセンター（物流 WS と呼びたい）においては，工程管理の責任を果たさなければならない。つまり，その物流 WS の納期の達成，実在庫の正確な把握，作業の無欠点化，正確な進度把握である。ただし，物流プロセスは自社で閉じていることはほとんどないので，物流 WS は自社以外の組織ということになる。それを管理するのは簡単ではないし，精度をあげることも単純ではない。

物流計画の策定では、各物流製品について，納期日ごとの物流計画を作成する．さらに，品目が同じで納期が等しいものはオーダをまとめて集約することができる．物流の場合にも，荷姿が同じものを作る作業を一度にやるという，いわゆるロット効果をねらえる。

5.8.4　計画の手配と管理

・手配計画

生産の場合のオーダリリースに相当する。物流手配とは，各物流ワークセンタに対して物流作業の指示をおこなうことである。

保管業務ワークセンタ：おもな業務は製品の保管であり，指示内容は入出庫の指示である。

転送業務ワークセンタ：物流センターからデポへの移動が主な業務である．転送手配とは業者に対して輸送指示を行うことである。遠距離輸送では，輸送自体が転送ワークセンタとなる。

取り揃えワークセンタ：ピッキング作業によって単体製品を物流製品に変換する．この指図のためには，取り揃えリストを発行するが，そのときにも物流 BOM が使われる。

マテハンワークセンタ：上記の３つの業務以外の作業である．仕分け，荷役，流通加

工などである．これも作業伝票が必要となる。

・進度管理

生産の場合の SFC に相当する。物流センター，デポ，工場倉庫ごと．生産管理の場合と同様に，計画と実績の差異のグラフなどを示し，作業改善に働きかける。

5.9 製番管理 - 伝統的日本式生産管理

5.9.1 製番管理と MRP

製番管理はフレキシブルな生産プロセスとして生まれた、個別受注生産のための日本の製造業の伝統的生産形態である。プロジェクト型の生産方式と呼ばれることもある。製番とは製造番号のことだが、個別受注を納期まで管理するために、諸材料の手配や生産の計画や製造指図や進捗管理などのすべてに、他の個別受注と区別しやすくする方法として製造番号を用いて、そのため、個別受注生産の計画管理方式を製番管理と呼んでいる。

発電所だとか、ビルやオフィスごとに異なるエレベータや空調設備、情報ネットワークシステムなどの必然的に一品料理的な「システム製品」ばかりでなく、たとえばワイシャツや他の衣服のような日用品の標準品が、個別受注で低コスト高品質で提供されるビジネスプロセスになると、消費者が得る価値は相当高次元になる。同時に、高齢者や障害者も衣服についてユニバーサルなサービスを受けとることが可能になる。このような意味で、完成された個別受注生産は社会経済システムのひとつの理想型でもあるし、マス・カスタマイゼーションの実際的な形ともいえるだろう。

製番管理・製番方式の基本的イメージは、特定顧客からの個別受注の機能を実現するための製品の具体的な設計と納期達成のための生産計画と生産管理を行っていき、具体化することで顧客や製造の都合によって時々発生する設計変更とそれにともなう生産計画変更を行いつつ納期内に完成するというものである。

MRP が「部品中心型」の生産方式であるのに較べ、製番方式は「個別製品中心」というべき考え方を持つ。すべての部品の生産計画を完全的に作ってから次に製造と管理を進めるのではなく、個別製品の計画と管理をかなり同時的に進めようというものである。MRP のタイムバケットごとの計画のような、一定期間ごとに区切って、生産数量をまとめて計画・管理するということもない。

5.9.2 製番管理の計画体系と進捗管理

製番管理の計画と管理は、たとえば、大日程計画・中日程計画・小日程計画に分けて次のように行うことができる（田中著：生産の実務　製番管理、2004）。

（1）大日程計画

新製品開発プロジェクトと似ており、設計、生産技術、購買、加工・組立の各部門についての日程をそれぞれ計画する。

各部門ごとの納期を見積もる。各部門の内部にはいくつかの作業が部品表のような構造を持っているはずだが、最長のパスについてのみ納期を計算することによって、その部門の納期見積もりを得る。そのため、各部門の日程は 1 本の線として考える。

特に設計部門からの出図時期が重要である。また、調達に長期間がかかるものはあらかじめ発注する必要がある。

新規の製品といってもまったくの新製品であることは少なく、過去の経験から部品表の概要を知っていたり、作業手順のどこがネック工程になるかの予想ができたりする。ネック工程とは負荷がかかり過ぎる工程のことだから、ネック工程の負荷調整をこの段階で行う。

（2）中日程計画

大日程計画での部門ごとの納期を、部門での仕事の工程に分解する。製造部分は生産技術部門が作業手順を作成する。

MRP は標準製品なので、設計図は不変。部品表も不変である。

製番管理では、設計から開始し、組立図も相当数必要なる。が、全くの新しい製品であることは珍しいので自社の類似品からの流用が可能である。部品表も流用する。作業手順も作業実行のワークステーションの配置も類推する。作業区ごとの日別作業を負荷計画として作ることで、ネック工程の予想と対策を行う。

治工具（特殊部品を汎用工作機械で加工できるようにするために部品を固定する等の目的の大小のアタッチメントなど）の製造も同時におこなう。

（3）小日程計画

生産計画を具体的に日程と作業内容を指定した手配計画（製造指図）として作る。MRP の場合の製造指図と同じ内容であるが指図情報の決め方が MRP の時のような自動的進行とは異なる。設計者がまず、各部品の手配区分を仮決めし、その後生産技術担当が最終決定する。手配区分とは、在庫品・非在庫品、反復品・非反復品、購入品・製作品の区別の指定のことである。製番手配マスターという製造指図を総まとめした情報を作って使う。MRP では計算の結果、製造指図が作られたが、製番管理では、設計と計画と管理を同時的に進めていく割り合いが　MRP よりはるかに強いので、いわば人力で種々の製造指図を作る。このため、逆に、設計変更があっても製造指図の調整も人間が行うことができる。

5.9.3 最終ラインがネック工程の時の製番管理

まず、ネック工程である最終組立ラインにおけるオーダの完了日・着手日が決められ、それから、通常、製造ネットワークにそって他の工程の着完日が決められる。製造指図をリリースした後に後追いで調整が入るのを避けられないと見て、在庫引き当てを真剣に行わなかったり、全く行わない。1 個のオーダの部品表と作業手順から計算する。

製番方式の基本手順は次の通りである。

(a) 受注を生産の単位にまとめ、製番（主キー）を付ける。まとめ方は、顧客、製品仕様、数量、納期を 1 セットとして生産単位とする。

(b) 個々の製番ごとに大日程計画を作成する。必要な資材を手配し、生産日程を決める。

(c) 各部門の中日程計画を作成する。

(d) 小日程計画を作成し、生産指図のリリース後の管理も、個々の製番ごとに行う。

個別受注生産工場の基本的な管理方法として、製番管理は自然な方法である。納期、コスト、損益、生産高も製番ごとに管理する。製番管理では、製作の指示、在庫品払い出し、外注品、購入品の調達、社内作業など、すべてに製番とその作業を表す項目番号が付けられているため、製番別の納期やコストなどの管理が、うまくいっていればやりやすい。

反面、製番管理では、すべての活動のキーが個々の製番だから、たとえ作業の内容や時期が同じだったり共通部品があっても別々に管理されることになるので、計画・管理しなければならない作業の種類や内容が増える。特に計画の何らかの変更があると作業間相互の調整も増えるので、間接仕事が膨大になる。個別管理を貫くので、期間別、品目グループ別の計画と管理がやりにくい。生産規模が拡大するにつれて受注物件が飛躍的に増えてくると、個別手配業務や、個別製番ごとの納期管理が追いつかなくなり、管理要員はその対応に追われ超繁忙で、その結果、所定業務の不消化や積み残しという悪循環に陥る。これを未然に防ぐために、部品やユニットなどの材料在庫を持つようになり、結果、間接要員の増大、在庫増、仕掛り増という問題になる。ただ、価格下落がなく売れ残りが死蔵品にならないような高度成長期には、これでも運転できる。

つまり、個別注文ごとの計画・管理のために製番管理の基本コンセプトは自然で必要であるが、基本型のままではまずいということである。

マス・カスタマイゼーションを目指すには、方向性としては、製品中心主義を部品中心主義と整合させることが必要である。たとえば、MRP とスケジューリングによって計画され、計画値がリリース後にそのまま実績値となっていくような優秀な生産プロセスを持っている場合には、個別注文の引合いのたびにそれを独立所要量として MRP を動かして所要能力の状況を見ながら再計画し、納期を確約できる。変更に対しても顧客との間のビジネスルールにしたがって可能なものは対応できることになる。このようなことはたとえば湯沢氏（中根著：総合化 MRP システム,1984）にも Make-to-order MRP として紹介されている。また ERP パッケージにおいてもすでに実現されている。問題点は、MRP を使う以上、その計算結果である計画の実行可能性ということである。

5.10 リードタイムのダイナミックな変化

これまで、加工や組立の工程のリードタイムは、品目ごとにちがうのは当然としても、同一品目については同じ数値になるものとしていた。MRP 計算のよりどころであった。実際には、リードタイムは期間生産性に属するものであってビジネスプロセスのネットワークの組み方によってもオーダの込み具合（需要の込み具合）によっても変動する。

作業手順は作業からなる。図 5-30 のように、各作業のリードタイムには、段取時間（セットアプタイム）、処理時間、片づけ時間と、待ち行列時間（すべての材料が揃ってから実際の段取作業が始まるまでの時間や機械が空くのを待つまでの時間）、移動時間（ワークセンター間の運搬時間）、移動待ち時間（処理終了後に次のワークセンターへの移動に取りかかるまでの平均時間）があった。

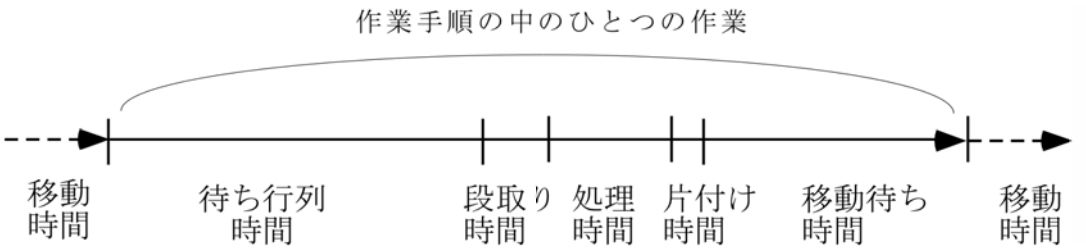


図 5-30　作業のリードタイム

待ち時間、段取時間、移動待ち時間、片付け時間は、場合によってはない場合もあるし、また、作業間でオーバーラップしていることもありうる。

例を見る。

(1) 組立プロセス：ＤＦＤモデルの例

大小２つの部品を組み立てて製品を作り、それを検査した後で製品倉庫に入れるビジネスプロセスである。材料は予め材料倉庫に入れられている（図 2-1)。

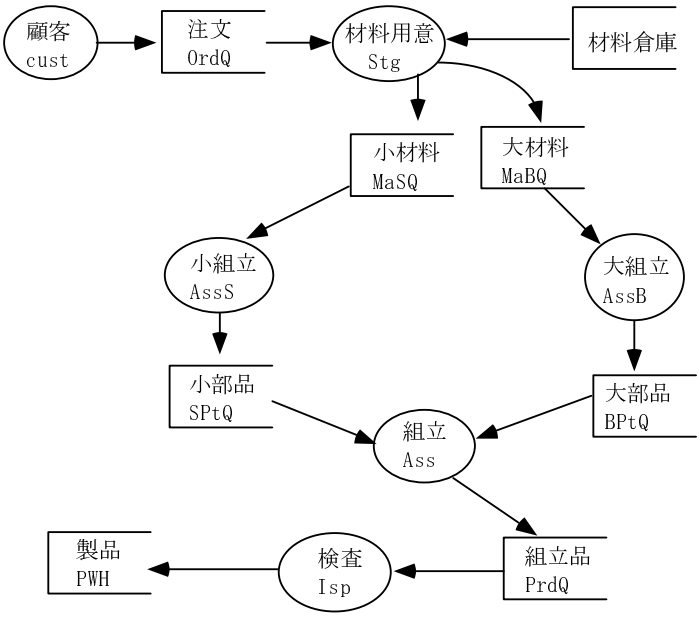


図 2-1（再出）　何の管理もない組立プロセス

各活動の作業者・開始条件・所要時間

各活動を実行するのは人か機械である。

１つの製品は、大小ひとつづつの部品から組み立てられる。各部品ひとつは各材料ひとつから作られる。注文は、1 つづつ注文するものとする。各活動は条件が満たされれば開始される。

材料用意(Stg)　　大小のペア 1 個当たり 5 分。

小材料組立(AssS)　小材料から小部品を組み立てるのに 1 個につき 5 分。

大材料組立(AssB)　1 個につき 6 分。

製品組立(Ass)　　1 個につき 6 分。

検査(Isp)　　1 個につき 2 分。

顧客(Cust)　　1 つづつ注文する。4 分おきにかならず誰かが 1 個注文する。

(2)　状態遷移表

始業時には、どの待ち行列もからっぽだとする。つまり仕掛りゼロとする。すると以

下の表 2-1 の状態遷移表を得るのであった。

この状態遷移表において、一番左の列が経過時間である。Stg とは図 5 の材料用意(staging)のことであり、-stg- とは、その時刻において活動 Stg が何もしていないことを表わす。1(5) というのは、1 個の仕事を扱っており、あと 5 分で活動による処理が終わることを表わす。Cust の列も同様。ordNo4(2)とは、2 番目の注文があと 4 分で発生すること。

time	Cust	OrdQ	Stg	MaSQ	AssS	MaBQ	AssB	SPtQ	BPtQ	Ass	PrdQ	Isp	PWH
0	odNo1(4)	0	-stg-	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
4	odNo2(4)	1	-stg-	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
4	odNo2(4)	0	1(5)	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
8	odNo3(4)	1	1(1)	0	-SAm-	0	-BAm-	0	0	-Asm-	0	-Ins-	0
9	odNo3(3)	1	-stg-	1	-SAm-	1	-BAm-	0	0	-Asm-	0	-Ins-	0
9	odNo3(3)	0	1(5)	1	-SAm-	1	-BAm-	0	0	-Asm-	0	-Ins-	0
9	odNo3(3)	0	1(5)	0	1(5)	1	-BAm-	0	0	-Asm-	0	-Ins-	0
9	odNo3(3)	0	1(5)	0	1(5)	0	1(6)	0	0	-Asm-	0	-Ins-	0
12	odNo4(4)	1	1(2)	0	1(2)	0	1(3)	0	0	-Asm-	0	-Ins-	0
14	odNo4(2)	1	-stg-	1	1(0)	1	1(1)	0	0	-Asm-	0	-Ins-	0
14	odNo4(2)	1	-stg-	1	-SAm-	1	1(1)	1	0	-Asm-	0	-Ins-	0
14	odNo4(2)	0	1(5)	1	-SAm-	1	1(1)	1	0	-Asm-	0	-Ins-	0
14	odNo4(2)	0	1(5)	0	1(5)	1	1(1)	1	0	-Asm-	0	-Ins-	0
15	odNo4(1)	0	1(4)	0	1(4)	1	-BAm-	1	1	-Asm-	0	-Ins-	0
15	odNo4(1)	0	1(4)	0	1(4)	0	1(6)	1	1	-Asm-	0	-Ins-	0
15	odNo4(1)	0	1(4)	0	1(4)	0	1(6)	0	0	1(7)	0	-Ins-	0
16	odNo5(4)	1	1(3)	0	1(3)	0	1(5)	0	0	1(6)	0	-Ins-	0
19	odNo5(1)	1	-stg-	1	1(0)	1	1(2)	0	0	1(3)	0	-Ins-	0
19	odNo5(1)	1	-stg-	1	-SAm-	1	1(2)	1	0	1(3)	0	-Ins-	0
19	odNo5(1)	0	1(5)	1	-SAm-	1	1(2)	1	0	1(3)	0	-Ins-	0
19	odNo5(1)	0	1(5)	0	1(5)	1	1(2)	1	0	1(3)	0	-Ins-	0
20	odNo6(4)	1	1(4)	0	1(4)	1	1(1)	1	0	1(2)	0	-Ins-	0
21	odNo6(3)	1	1(3)	0	1(3)	1	-BAm-	1	1	1(1)	0	-Ins-	0
21	odNo6(3)	1	1(3)	0	1(3)	0	1(6)	1	1	1(1)	0	-Ins-	0
22	odNo6(2)	1	1(2)	0	1(2)	0	1(5)	1	1	-Asm-	1	-Ins-	0
22	odNo6(2)	1	1(2)	0	1(2)	0	1(5)	0	0	1(7)	1	-Ins-	0
22	odNo6(2)	1	1(2)	0	1(2)	0	1(5)	0	0	1(7)	0	1(2)	0
24	odNo7(4)	2	1(0)	0	1(0)	0	1(3)	0	0	1(5)	0	1(0)	0
24	odNo7(4)	2	-stg-	1	1(0)	1	1(3)	0	0	1(5)	0	1(0)	0
24	odNo7(4)	2	-stg-	1	-SAm-	1	1(3)	1	0	1(5)	0	1(0)	0
24	odNo7(4)	2	-stg-	1	-SAm-	1	1(3)	1	0	1(5)	0	-Ins-	1
24	odNo7(4)	1	1(5)	1	-SAm-	1	1(3)	1	0	1(5)	0	-Ins-	1
24	odNo7(4)	1	1(5)	0	1(5)	1	1(3)	1	0	1(5)	0	-Ins-	1
27	odNo7(1)	1	1(2)	0	1(2)	1	-BAm-	1	1	1(2)	0	-Ins-	1
27	odNo7(1)	1	1(2)	0	1(2)	0	1(6)	1	1	1(2)	0	-Ins-	1
28	odNo8(4)	2	1(1)	0	1(1)	0	1(5)	1	1	1(1)	0	-Ins-	1

表 2-1　組立プロセスのふるまい（状態遷移表）

開始後28分までの間で考えると「材料用意」活動のリードタイムはいくらになるか。

28 までの間に製造注文は No7 までが到着し、28 分では２つ（オーダ 6 とオーダ 7）が処理を待っている。入力待ち行列の最終オーダであるオーダ 7 の処理を終えるまでの時間を含めて、各オーダのリードタイムを計算する。

	待ち時間	加工時間	オーダごとの合計
オーダ１	0	5	5
オーダ２	1	5	6
オーダ３	2	5	7
オーダ４	3	5	8
オーダ５	4	5	9
オーダ６	5	5	10
オーダ７	6	5	11

「材料用意」活動のリードタイムは（合計）／７＝５６／７＝８分と計算される。

この例で、もし、顧客からのオーダが到着する時刻を変化させると、待ち時間が変化するのでリードタイムは変化する。タイムバケット内に製造する量が与えられたとき、製造に必要な作業ネットワークの中の作業する順番を変更すると、待ち時間が少なくできる可能性がある。個別作業の着手計画がスケジューリングであった。

5.11 間接業務の改善

一般のビジネスプロセスは、生産プロセスだけではなく多くのいわゆる間接業務のネットワークから構成されている。間接業務について管理会計の立場からより正確な記述を目指すものに、活動基準原価 (activity based costing) がある。考え方の基本は、生産プロセスや物流プロセスの場合と同型である。

たとえば、買掛伝票処理に必要なビジネスプロセスが図 5-31 のようであるとする。1 案件について、作業は左から右に行われるので、「買掛伝票処理」という「サービス製品」を表す作業手順とも部品表と見なせることから、生産や物流の場合と同じように、計画やスケジューリングや原価計算ができることが想像される。各活動についてこのようなサービス製品ととらえその部品表と作業手順を考えることで、伝票 1 枚あたりの作業のコストであるとか、催促ファクス 1 通あたりの作業のコストを標準原価計算法として定めておき、枚数などに応じて計算できる(Sheer, 1989)。

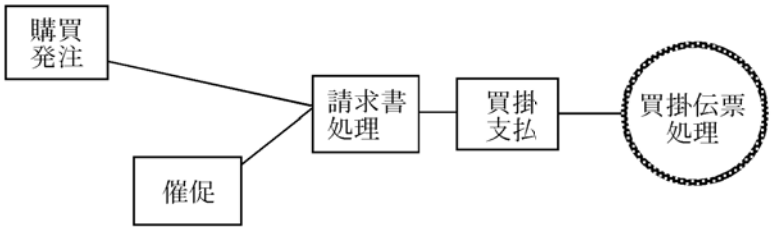


図 5-31　買掛伝票処理

会計的なビジネスプロセスのとらえ方は業種を問わずに普遍的に適用できるし、実践的で有効であるが、ジョンソン（1994）が指摘するような限界を持つ。ジョンソンは組織の構成員によるボトムアップなエンパワメント、つまり構成員各自の不断の改革意思と実行が有効であることを主張したが、具体的な方法論については明確ではない。以下では、田中著「時間生産性をどう高めるか」にしたがって、間接業務製造業においては、直接に製品を設計したり加工したりして物理的に製品にする活動ではない仕事を、伝統的に間接業務と呼んでいる。間接部門の仕事の内容を大分類すると、デスクワーク、コミュニケーション、その他の３つに分けられ、それぞれが一人の仕事時間に一定の割合いを占める（図 5-32)。

	標準作業	例外作業	個別作業
デスクワーク	10	5	35
コミュニケーション	6	3	21
その他	4	2	14

図 5-32　担当者 A の仕事内容の構成比率（田中著 : 時間生産性をどう高めるか)

デスクワーク：書類作成・資料収集・資料検討・コピー・ファイリング・データの入力やチェック。
コミュニケーション：電話・会議・社内や来客の打ち合わせ・問い合わせ・送受信。
その他：思索・情報閲覧・喫茶や喫煙・離席。

表において、標準作業とは形式化が進んだ定型業務である。たとえば、標準品目の注文やそれに関する問い合わせ、定型的な生産計画作成などがある。したがって基本的にはコンピュータ化されている仕事である。つきつめれば、担当者の仕事はほとんど端末機の操作であり、意思決定としての判断はほとんど必要ない。
例外作業とは、代表的にはンピュータ処理における例外作業のことである。たとえば、スイッチを入れる・切る、予測できない動きをコンピュータがしたときの対処など。だから、コンピュータ部門が作業内容を決めるので、意思決定するような性質の作業

は含まれない。

個別作業が間接業務の大部分をしめる。組織の仕事を専門化し細分化した結果、同じグループ内でも自分のことしか分からない状況になっている。担当者は自ら決定し自ら実行する。ある顧客へのアプローチの検討、そのための実績の収集、そして決定後の書類作成と会議が一つの例である。金額がある一定額以上の材料の納入業者の選定に関わる決定や、また、あるいは、なんとなく部署の仕事の流れがうまく行っていないという認識からの問題設定と解決や検討への一連の筋道をつけること、など、多岐に渡る。個別作業を要求する仕事とは、細分化された専門性への要求、不定形さという特徴を持っている。情報システム部の仕事も、製品を作る直接作業というわけではないので間接作業であり、専門性と不定形性を持っている。

第１の仕事以外には第２の仕事がある（図 5-33)。これは第１の仕事のやり方や構造を考えて変更や改革をするものである。ある時点での仕事の結果を踏まえ、改善をする。さらにその結果を踏まえ、さらに改善する。これを続けていく。自分で自分や組織の仕事を考える。社会学では自己参照フィードバックともいわれる。

間接部門業務には、このようにフレキシビリティがある。また、個別作業がほとんどをしめるとなると、作業の標準化が困難である。能力・習熟・気合いが、おこなうべき仕事によって個人間でのばらつきが大きい。さらに、間接業務でたとえば仕事のやり方を改善する場合には、何をどの範囲までを仕事としてとらえるかという段階においてすでに個人間のおおきな違いがある。

こういった間接業務の時間生産性を高める方法は何らかの枠をはめることである。まず、標準時間や標準作業を設定して、作業のフレキシビリティ(個人依存性）を減らすことである。たとえば、面談時間は相手と面談内容で大きく異なる。しかし、面談時間を設定すれば、面談はそれに収まる。また、仕事のシステムを変える方法もある。組織階層を減らし、かつ間接業務の担当者数を減らすことはすぐにでも実行可能だが、その結果、個々の担当者の仕事の内容は拡がって、結果として全体の意思決定が遅くなったり、質が低下することがありうる。その対処策のひとつが、コンピュータ化であったが、標準業務と例外業務には効き目があっても、個別業務には適用しにくい。

さらに、人間は、枠をはめられると士気(モラール）が下がる可能性もある。

個別業務に有効に枠を与える方法は「評価」であると考えられる。図 5-33 にその基本構造が示してある。組織機能のこのような階層性は、人工物の科学として基本的構造として認められるし（サイモン, 1999)、プリンシパル・エージェント関係や、さらに組織の統合や自己組織化を論ずる際の基礎となる考え方でもある。

担当者は、まず、組織の命令と責任の仕組みの中に置かれている。組織構成員とし

て、指示されれた仕事を行う。しかし、仕事はこまごまと述べられるのではない。大枠が示されるに過ぎない。おおよその見当で担当者にディスパッチされるのである。具体的な進め方は自分で考えて実行し、時に応じて自ら変更していく。このような担当者の決定や行動に影響を及ぼすのが、「評価」なのである。

評価には 3 種類ある。組織評価、自己評価、チーム評価である。

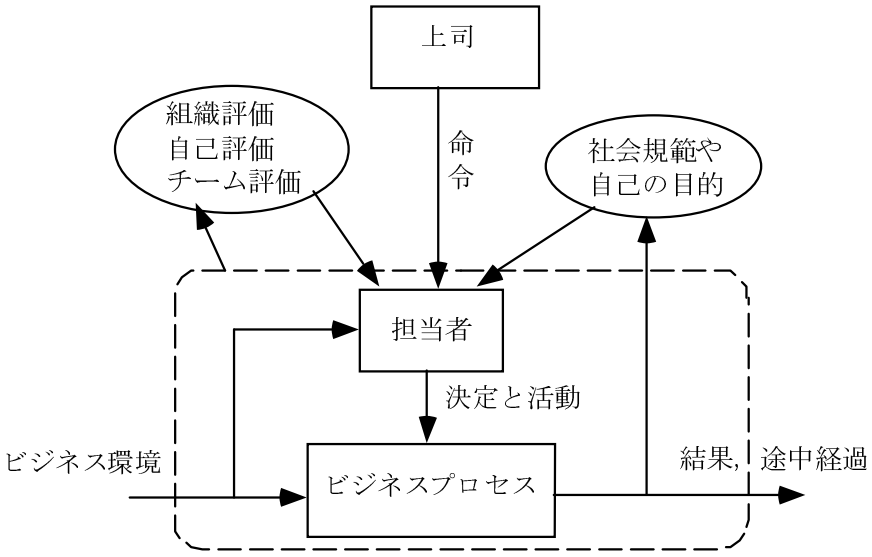


図 5-33　命令と活動と評価

組織評価：

フォーマルな評価。期初に上司と部下の間で評価項目や目標値についての契約が交わされ、期末にそれらを環境状況や他者状況を踏まえて評価する。多角性、公平性、いろいろな項目間のバランスが重要になる。数値化できる項目にだけ限定するという方向もある。

自己評価：

プロとしての自己の仕事に対する自分が行う評価である。プロフェッショナルであることを高めていくには、高い自律心が必要とされる。また厳しい自己評価は、自分が自分を伸ばすためには不可欠だろう。

チーム評価：

チームメンバーや、あるいは同業他者によって相互に行われる。インフォーマルに行われるところが、組織評価と異なる。しかし、このインフォーマルな相互評価という他者の視線が、自己の評価に与える影響が大きいので、組織のビジネスプロセスの仕組みに取り入れることができれば、間接業務をコントロールする仕組みにできそうである。

個別作業が持つ不定形性とは、いくつか用意してある専門的な対処のパターンからの逸脱である。多くの不定形な個別的作業が発生するので、専門担当者におおよその検討で割り当てられはするがそれは専門外の知識も要求する。それにもかかわらず、結果を求めて仕事に対処する過程において、自己の専門の幅を拡げたり、上司・先輩・部門外や組織外の人間との力の貸し借りが行われる。成長や変化がチームの他者から評価されるのである。

日本の会社人の行動を外から見れば、戦う狼集団と言われたことがある。内に向かってはお互いに助け合うが、外に向かっては、相手を倒すまで一丸となって原価割れを起こしてまで戦う（ように見える）というのである。

また、終戦後の高度経済成長期には、十分な知識を持った人も足りず、資源もなく、必要な技術もなかった。たとえば自動車産業である。そこで、各企業に個別の状況で互いに知識を持ち寄り発展させながら個別の方法を発展させ、人間の能力を向上させる方向に、2 番手戦略などを用いて発展してきた。チーム評価が有効に働いていたといえよう。ここに来て、年俸制、契約社員、人材派遣と、急に個人の能力や知力や先見性が取りざたされるようになってきた。プロであり続けるとはいろいろな姿があるだろう。自己がプロである領域を定めたり、あるいは変化していく先端領域を造り出したりするというスーパープロフェショナルもいれば、税理士や中小企業診断士のような公的資格を持つという意味もある。プロの主婦や主夫、プロの学生、プロの生活者という主体が可能であり、それぞれの生活の作業からなる「ビジネスプロセス」を持っている。

第 6 章　情報システム方法論の意味論

6.1 情報システム方法論の図的ツール

これまで展開した業務システムの離散事象モデル理論によって、アクティビティ・インタラクション・ダイアグラム (AID)とデータモデルは、ビジネスプロセスの仕組みを記述できることを学んだ。本章では業務取引システムの静的構造（ファイルシステ、外生トランザクション、内生割り当てトランザクション、内生終了トランザクションの全体）を使って、いくつかの情報システム方法論の重要な図的ツールがビジネスプロセスの何を記述しているのかを考察する。

情報システムはそれ自体のために存在するのではなく、ビジネスプロセスを有効に機能させるしくみの一つである。したがってどんな情報システム方法論でも、それが分析の対象とするのはネットワークとコンピュータ化された情報システムであるよりは、まず第１にビジネスプロセスであるべきである。そのようなビジネスシステムを分析し設計するために開発されてきたのが、さまざまな情報システム方法論が持つ概念とツールである。それらのツールによって業務システムが定義される。種々のツールは便利に用いられてきたが、それらがビジネスシステムのどの側面をどれぐらいの程度で記述しているかは必ずしも明確ではない。

ビジネスプロセスと情報システムの設計図が必要であるが、ビジネスプロセスの設計図として、たとえば機械部品の図面のような精度を持つものはない。ビジネスのしくみは不定形で、物的に規定されていないことが原因であって、VLSI や IC チップの電子回路の論理設計レベルの設計図のようなものは望めなかった。いきおい、ハードウエア構成やネットワーク構成図によってビジネスプロセスの概略が描かれることになりがちである。一方、財務情報はビジネスプロセスの活動をすべて金銭情報に置き換える優れた方法であるが、金銭データは情報システムの機能や生産管理方式の設計や性能評価とは独立である。

本章で、ビジネスプロセスと情報システムのモデリングのために提案された図的ツールの意味を確定する方法を示す。これによって、図的ツールによるモデル化へのガイドラインも得られる。なお、ビジネスプロセスの性能評価は次章のトピックである。

情報システム方法論の歴史的発展と、ビジネスと情報技術との連係を文献を示しつつ次のように眺めておく。

（１）構造化分析と設計：Structured analysis and design methodologies

構造化プログラミングと同じ時期の 1970 年代に出てきて以来、IDEF という名称で標準化されて現代でも使われることが多い。流行すると何でも「構造化○○」というように使われる傾向があった。主要な文献の一例として次をあげることができる。

・DeMarco, T 著"Structured Analysis and System Specification" Prentice-Hall, 1979.（高梨智弘他訳、構造化分析とシステム仕様、日経 BP 社、1986）

・Page-Jones, M.著 The Practical Guide to Structured Systems Design, Yourdon Press, 1980.

（２）オブジェクト指向方法論：object-oriented methodologies

コンピュータのハードウエアが進歩を続け、グラフィカル・ユーザインターフェイス（GUI）やソフトウエアを「部品化」した上で少ない変更で再利用する方法の確立をめざして、オブジェクト指向プログラミングが 1990 年代初頭から提唱された。やはり、流行すると何でも「オブジェクト指向○○」と呼んでその精神を取り入れようとする傾向がある。オブジェクト指向方法論は、オブジェクト指向プログラミングとほぼ同時期に提唱されて現在も発展している。構造化分析・設計方法論の欠点を検討し改善した結果出てきたというのではなく、プログラミング・パラダイムの流行と発展が情報システム方法論に大きく影響するのである。オブジェクト指向方法論の主要な提唱者が集まって統一的な定義を決めて、UML（unified modeling language）として図的ツール群が世の中に広がっている。図的な「言語」であり解説書が数多く出されている。日本では、比較的早い時期に出版されながら、UML の表記法と意味の説明を超えて方法論も提唱した優れた著書として次があげられる。

・吉田裕之、山本里枝子、上原忠弘、田中達雄著「UML によるオブジェクト指向開発実践ガイド」技術評論社、1998.

（３）CIM と ERP

（CIM - computer inegrated manufacturing、ERP - enterprise resource planning

package）

コンピュータの計算能力のビジネスへの本格応用として、データベースが世界で初めて使われたのが MRP であったことからも分かるように、コンピュータや情報技術の発展はビジネスの構造を変えていくインパクトを持つ。化学工学でも DDC（ダイレクト・デジタル制御）という制御計測装置の計算機化による自動化が起こったが、製造プロセスとそれを取り巻く形で受発注や会計処理を含んで動いているビジネスプロセスへのコンピュータの適用は、CIM を経て ERP となっている。なお、ERP の訳語としてひろく使われている統合基幹情報システムはよく意味と意図をとらえている。ERP の場合の基幹業務プロセスとは、営業、購買、生産、物流、財務経理、人事、R&D といった基本業務のことである。ERP は実務を運転し管理するためのデータ統合の仕組みを持っているばかりでなく、データ倉庫や分散化統合技術等のあらたな情報技術が現れるたびにそれを取り込んだり統合プラットフォームを用意することなどにより、まさに情報化社会や知識社会における情報的な社会基盤となりつつある。

次の本は CIM や ERP の構造をデータモデルで表した画期的なものである。

・Scheer, A.-W, Business Process Engineering, 2nd edition, Springer, 1994.

また、ERP 使用を前提とした情報システム方法論は、当然ながらゼロからビジネスプロセスと情報システムを分析設計する方法とは異なる。ERP の中に既に実現されている資源を使える優位性と、ERP の中に既に実現されている幅広い機能の組み合わせによって自社の都合を実現するという制限があるからである。そのための方法論は数少ない。実際に使えるレベルの詳細さで説明したものとして次がある。

・Keller, G and Teufel, T.著「SAP R/3 Process Oriented Implementation: Iterative Process Prototyping」 Addison-Wesley, 1998（田熊博志訳、SAP R/3:プロセス指向型の ERP 導入、2000）

（4）分散システムとビジネス連携とシステム連携

分散化と連携は、ビジネスと情報システムについての古くて新しい話題である。しばらく前から、データベースといえば、単体のソフトウエアではなく DBMS + Web アクセスのことを意味している。つまり、自社内で使うときも社外から使うときも、データベースを集中サーバーに置き、クライアントのコン

ピュータからネットワークを介してアクセスする形式が一般的になった。分散したハードとソフトとネットワーク全体がひとつの機能を果たすので、分散システムと呼ばれている。したがって、他社のシステムとの接続や、部門をこえたシステム結合や、あるいは、蓄積データを分析ソフトで使うためのシステム接続を行う場合、「接続」の意味は複数の分散システムの接続ということになる。ERP でさえもひとつの分散コンポーネントしてとらえられるわけである。

分散システムの結合は、異なるデータフォーマットや通信プロトコルの接合を行う必要があるし、また、セキュリティー技術が重要である。社内であっても異なる情報システムが分散して存在しているので、それらを連係させたり、さらに外部組織とのロジスティクスプロセスとサプライチェーンを構成して連係することで、ビジネスプロセスの統合と拡大をはかろうとしている。

HTTP と XML と SOAP をつかった Web サービスを要素技術として使った SOA (service oriented architecture)や、EA (enterprise architecture)を実現する方向が徐々に実現されつつある。SOA では既に使われている多くの通信プロトコルの変換と（プロトコル・アダプター)、特定情報システムや会社の枠を超えた分散システム上の通信メッセージと（データモデル＋データ)、そのデータの受け手・送り手プログラムの指定を行うことで（メッセージ・コレオグラファー＝振り付け師)、多くのビジネスプロセスの連携を行えるように基盤技術を整えつつある。また、EDI（電子データ取引）はインターネット以前からの長い歴史を持ち、主に受発注伝票の電子化によるファイル転送によってシステム連携を行っていた（三菱総研, 2005)。EDI ではデータ転送は自動的だが、送られたデータを使うプロセス連携はクローズドで実質的にオフラインの定期処理であった。これを XML 等の利用によってオープン化して中小企業にも門戸を広げ、かつ汎用的技術で利用障壁を下げて、社会的なビジネスプロセスのオン・デマンド的連携とプロセス連携による企業を超えたプロセスの自動化を広げたいというのがこれからのシステム連携の意図である。実質的に互いのビジネスプロセスを他から利用可能なサービスとして提供し連係しあって、業界全体としての分散プロセスによって高度なビジネスを達成しようとするものである。

6.2 データフローダイグラム

6.2.1 光速通販 (mail-order-business) の受注配送のビジネスシステム

ひとつの通信販売の業務システムを通して、データフローダイアグラム(DFD)やエンティティ・ライフ・ヒストリー（ELH）について業務取引システムの理論が明らかにすることを説明する。国際標準の IDEF0 は実質的に DFD をもとにしたビジネスプロセスの図的標準ツールである。以下では、読みやすいように図を多く用いる。

光速通販物流管理部

図 6-1 は本章で用いる想像上の会社、光速通販社の物流管理部門である。このケースは島田清一[システムエンジニアのための業務分析の手法, 日刊工業新聞社, 1988] からの解題である。物流管理部門の主な機能は受注と配送管理と在庫管理である。光速通販社の顧客は、商品カタログをみて注文する。注文は顧客の近所のアクセスポイントへの電話やファクスによるか、コンピュータネットワークを介して届けられる。商品の種類によって、全国に展開している自社の倉庫に在庫として持つ商品と、直接にメーカから配送される商品がある。図 6-1 は物流管理部門のビジネス環境、つまり文脈を示している。

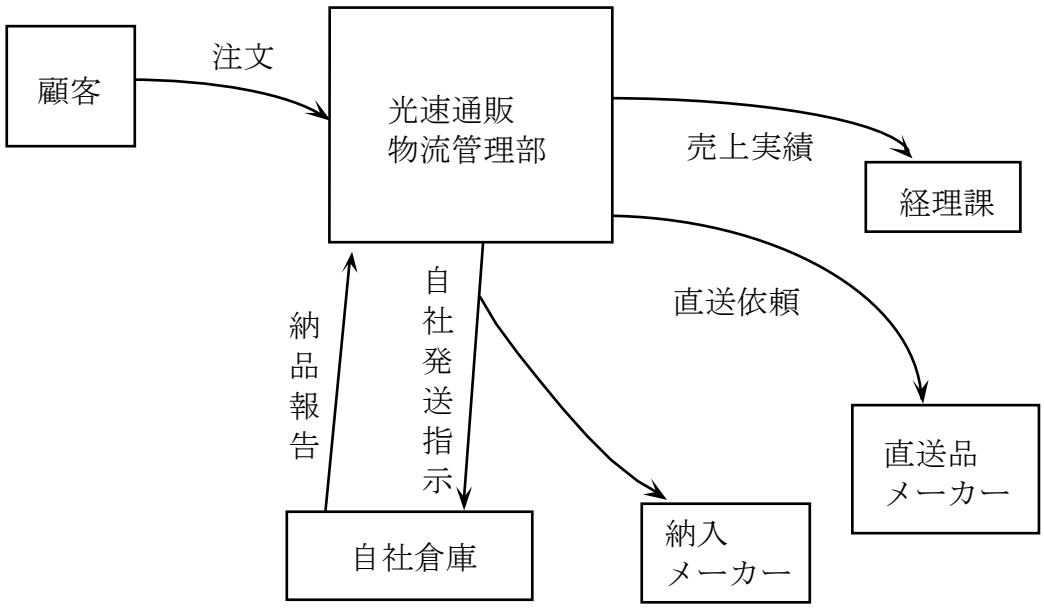


図 6-1　光速通販社物流管理部の文脈ダイアグラム

図 6-2 は受注配送業務を行なっている物流管理部の定型業務のようすを記述したアクティビティ・インタラクション・ダイアグラム（AID）である。図 6-2 の入庫伝票と出庫伝票は、図 6-1 の納品報告である。図 6-1 の自社発送指示は図 6-2 の出荷依頼書、納品書、未記入受領書の組である。図 6-1 の直送依頼も同様である。

物流管理部の図 6-2 の中の各活動は以下のようなビジネスルールで運営されている。

受注する：

注文はメモやファクスやコンピュータネットワークを通じて届く。あらかじめ登録してある客からの注文（一般注文）を受注表に書き込む。一般注文でない注文は、与信調査の後に受注する。図 6-1, 6-2 では、例示のために一般注文の場合に集中し、与信調査は省略されている。また、EDI や Web サービスが利用されている可能性があり、データが届く方法は何でもよい。

在庫管理と製造指示をする：

自社倉庫の在庫とする品目は、納入メーカーに対して発注するとともに、自社倉庫へもその発注内容を送る。

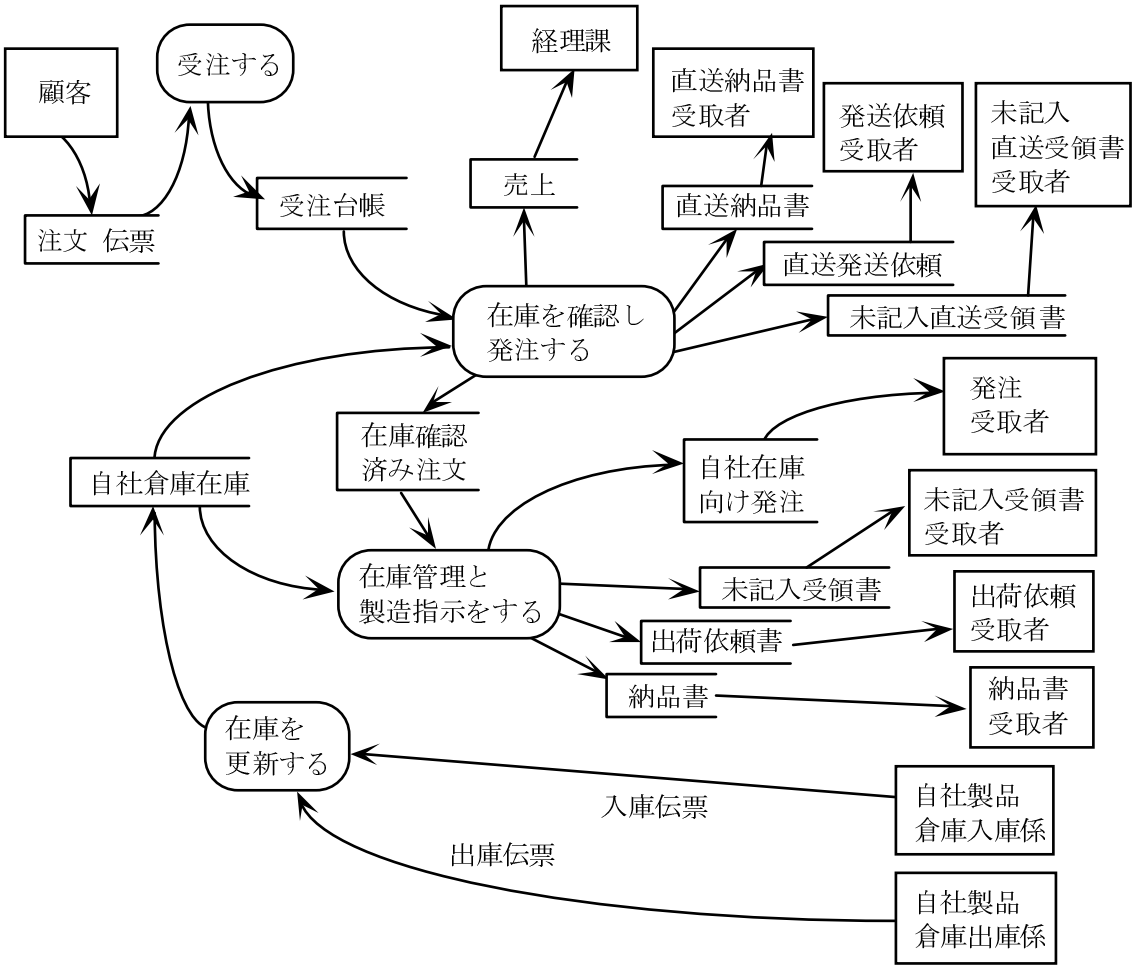


図 6-2　光速通販の物流管理部の AID モデル

在庫を確認し発注する：

自社在庫品目についての受注は、引き当て可能在庫があればそれを引き当てて、在庫からの出荷可能注文とする。

自社在庫が不足ならば、在庫不足注文とする。したがって、在庫確認済み注文は、出荷可能注文か在庫不足注文かのどちらかである。

直送依頼品目については、直送依頼書を自社納品書、未記入顧客受領書とともに直送メーカーにファクスかモデムで送る。

直送依頼品目と、自社在庫品目のいずれも売上を計上する。

在庫を更新する：

入庫と出庫の実績を、自社倉庫在庫表に反映するために更新する。

各活動をつなぐのは、伝票やメモ書きやデータベースである。コンピュータベースの情報システムが使われているときには、パソコンやモバイル、ケータイの画面が使われている可能性もある。物の移動である物流も、仕事の開始や終了による業務の状況の変化と同様に伝票に反映されている。生産ラインで部品が加工や組立を待って滞留するのと同様に、伝票も滞留する。正確には、受注伝票に記載された一連のビジネス活動を必要とする要求が滞留する。このような滞留可能なビジネス要求を図 6-2 では開放長方形で示した。

6.2.2 ファイルシステムのデータモデル

ファイルシステムは帳票のデータモデルであり、いくつかの表とそれらの間の整合性条件から成る。第 3 章の説明のように業務システムをモデル化する場合には、ひとつの種類の帳票が DAE データモデルのひとつの管理実体型に対応する。たとえば、受注伝票は必ず受注概要と受注細目の 2 つの管理実体型に分かれる。ひとつの管理実体型はデータベース管理ソフトウエアで実現されるときにはひとつの表となる。管理実体型の属性は、対応する表の属性（データ項目欄）となる。DAE データモデルでは表は必ず主キーを持つ。主キーとは、たとえば顧客管理実体型で顧客に付けた番号のように、その値を指定すれば顧客を一意に指定できるような属性であった。また、ある納品書がどの受注に対するものであるかを示す場合のように、ある表の主キー欄と他の表のある欄との間の対応は、参照関係と呼ばれる 1 対多の関係がある。参照関係が指定されていれば、たとえば顧客番号が分かれば、顧客表から顧客名や届け先住所がわかり、納品書のアドレスとして使用される。データベース管理システムではこれを自動的に行なう仕組みをもっているし、手作業の場合でも検索が可能である。以下では光速通販のいくつかの管理実体型の属性を示す。管理実の属性を記述する方法は 3 章の図示法を簡略化して椿正明著「概念データモデル」に基づいて行う。各管理実体型のデータモデルの初めの属性はブラケットで囲まれ、それが主キーであることを表わす。実際には、主キーとしてはコード化された番号が用いられる。番号は＃で表わしている。参照関係は明示していないが、ほぼ自明にわかる。

（1）管理実体型

光速通販の受注台帳は、次の２つの表で管理している。これによって分納の管理が可能となっている。
受注管理実体型：誰からの注文をいつ誰が受けたか。
[受注＃] 顧客＃、日付、受付従業員＃

受注明細管理実体型：どの商品の受注を受けたか、どの受注書の明細か、を保持する。これを表として示すと図 6-3 となる。図で「受注明細」はこの表の名前である。
[受注明細＃] 商品＃、受注＃、数量、単位、納期、配送済み区分

受注明細						
受注明細＃	商品＃	受注#	数量	単位	納期	配送済み区分

定義データ

表に格納されるデータ

図 6-3 受注細目管理実体型の表による実現

上の属性の「配送済み区分」とは、在庫引き当ての状況を表現する欄である。なお、属性の順番は表の定義では論理的に無関係である。

顧客管理実体型
[顧客＃] 顧客名、住所１、住所２、電話、ファクス、与信クラス＃、職業、勤め先、生年月日

従業員管理実体型
[従業員＃] 氏名、住所、電話、職位級、部署名、入社年月日、生年月日

納品明細管理実体型　：納品明細は受注明細と実質的には同一である。伝票としてはカーボンコピーを用いたワン・ライティングによって、受注明細作成時に作られる。

[納品明細＃] 受注明細＃

納品管理実体型：納品は原則として受注ごとに行なう。だからいくつかの商品を一度に受注した場合は、受注明細を受注ごとにそろえてから納品する。例外的に分納も行なう。
[納品書＃] 受注＃、商品＃、数量、顧客＃、日付、納品担当者＃

直送依頼管理実体型
[直送依頼＃] 受注明細＃、直送メーカー＃、依頼日付、発送日付、納品書＃、顧客受領書＃

顧客受領書管理実体型：ワン・ライティングで納品と兼用されるので省略する。

直送メーカー管理実体型
[直送メーカー＃] メーカー名、連絡先ファクス、振込口座、電話、住所

商品管理実体型：商品マスターファイルと、自社倉庫在庫ファイルを兼ねる。
[商品＃] 商品名、直送か自社在庫かの区分、単価、重量、大きさ、荷姿

他の管理実体型については省略する。

（2）管理実体型からの伝票の作成
　管理実体型の表の中の１つの行が、ビジネスにおける１枚の帳票の中のデータに対応する。人間の目にデータがふれる場合には、たとえば、顧客コードや商品コードという主キーの値ばかりでなく、それらの主キーによって定まる顧客名や顧客の配達先住所とか、あるいは商品名や商品単価などが、必要に応じて表示される。たとえば、光速通販の与信未調査受注明細管理実体型と受注管理実体型の属性は以下のようになる。
[与信未調査受注細目＃] 商品＃、数量、納期、配送済み区分、受注＃
[受注＃] 顧客＃、日付、受付従業員＃
であるが、これを人間に表示するときは次のようにする。

[与信未調査受注細目番号] 商品＃、(商品名、単価)、数量、(金額)、納期、配送済み区分、受注＃、(顧客＃、(顧客名)、日付、受付従業員＃、(担当者氏名))

実体の世界でのモノやコトを正確に記述できるように、主キーや参照関係などによってデータモデルを正しく作ってあるので、どのような取引が大量におこってもこのような表示が可能になっていることに注意してほしい。

6.3　データフローダイアグラムの意味

データフローダイアグラム(DFD)は 1950 年代から言語に代わって数多くの情報システム方法論で用いられている分析技法である。データフローダイアグラムと業務取引システムの原始静的構造との間に自然な対応を定め、その対応の下でデータフローダイアグラムのモデル化能力を検討する。これにより、業務の並行処理を可能とするようなデータとプロセスの相互関係に注目して業務システムをモデル化したものが、データフローダイアグラムによるモデルであることが証明される。その帰結として、データ設計を別に行う必要が理解され、また、データフローダイアグラムの記述の拡張が提案された。

DFD は歴史的にはヨードン、デマルコ、ゲインズ&サーソンらの構造化分析によって広まった。ビジネスシステムの全体を描いたりするのに便利であるので、いまでも多くの情報システム方法論で用いられている。これまでは、DFD はビジネスシステムのイメージを表現するのに使われてきたが、ビジネスシステムのどういった部分のモデルであるかを決定することがこの考察の目的である。その結果、DFD によってビジネスプロセスをモデル化する際のガイドラインが得られ、また、ビジネスプロセスのモデル化についての理解が深くなる。

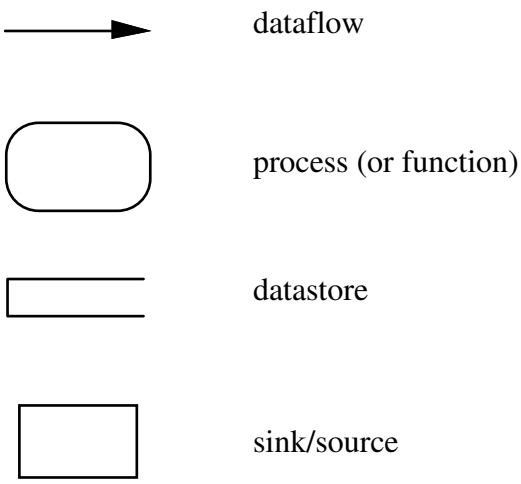


図 6-4　データフローダイアグラムの構成要素

図 6-2 の光速通販の物流管理部の業務取引システムを、データフローダイアグラムで描くと図 6-5 のようになる。直感的にいって、業務取引システムの原始静的結合構造と DFD はかなり似ている。

形式的な定義は 6.4 節にあるが、業務取引システムの基本静的結合構造とは、DFD のデータフローに対応するところに必ず管理実体型（概念ファイルや表とも呼ぶことがある）が存在することである。

業務取引システムについて第 4 章に示した通り、それが静的結合構造をもち、それらのトランザクションがひとつの DEVS として離散事象システムに対応し、全体として離散事象の時間変化のメカニズムで動く一個の離散事象システムであることが証明された。したがって、もし一定の形式的な対応関係があることを示せれば、その対応によって、DFD が業務取引システムのどの部分を描くか、モデル化するかが決定される。つまり、DFD がビジネスシステムをどこまで決定しているのかという意味が、業務取引システムの理論から証明される。一般に、似たものどうしの正確な類似点を確定するのはトリビアルではない。たとえば、数学においては、たくさんの自然な数学的構成が同型であることはカテゴリー理論が発見されるまでは明確でなかった。本節ではバーバルに DFD の意味づけを行う。精密な形式的証明は次節にある。

図 6-5 において、まる四角 round rectangle はあるひとまとまりの業務活動を示す。四角は注目している業務以外のシステムであって、外部の会社や、外部の部署である。矢印はデータフローと呼ばれ、帳票やデータ通信やファクスによ

るビジネスデータが流れる仮想的なパイプラインを表現する。それらは業務と業務、または業務と外部組織の間でやりとりされる。

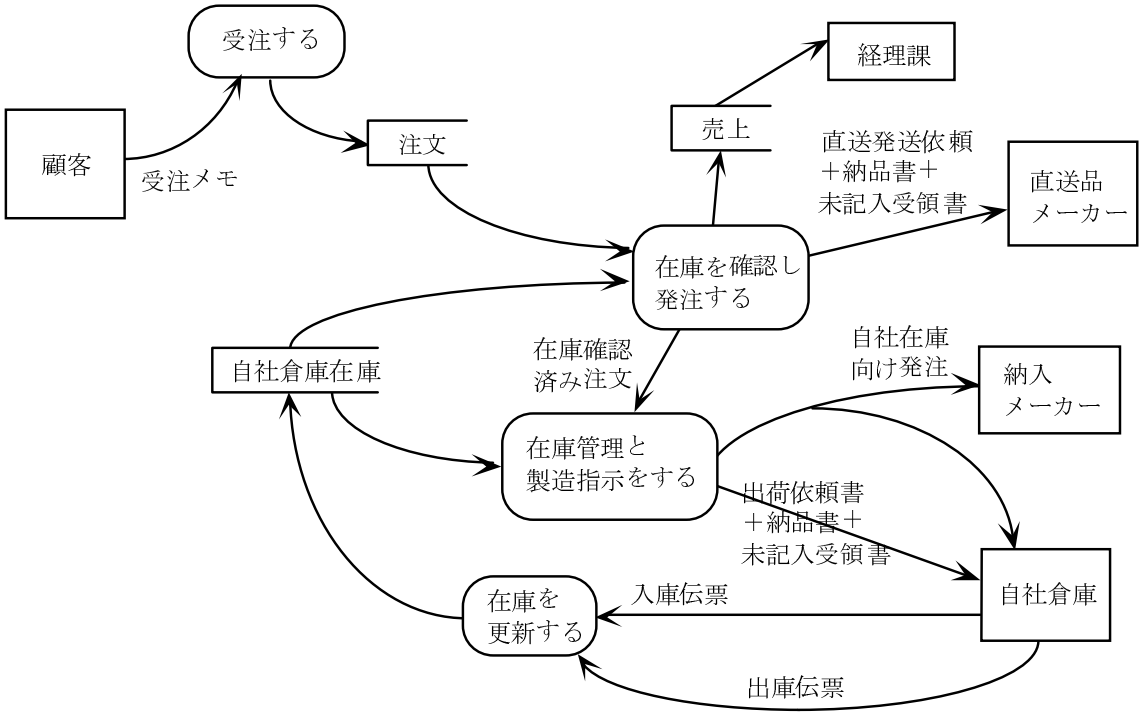


図 6-5　光速通販の DFD

通常は DFD は階層的な関係にある何枚もの DFD としてグループで使われる。各階層では解像度が異なる。グループ全体として、ビジネスシステムのトランザクション処理の相互依存性を表現する。

DFD と業務取引システムとの対応関係は、図 6-6 と図 6-8 のようになる。業務取引システム（の基本静的結合構造 PSS）から DFD への対応 G は図 6-6 で示される。

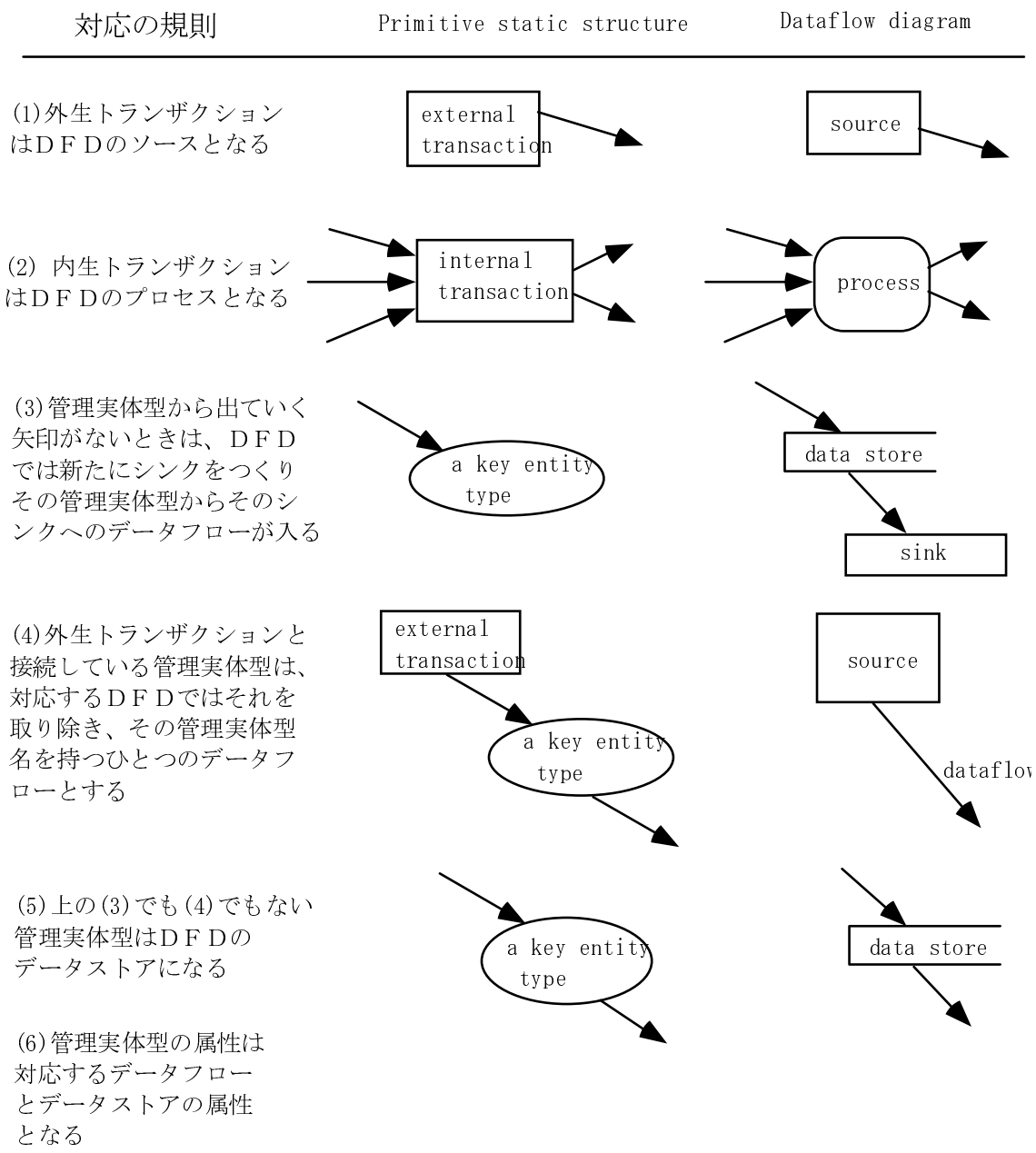


図 6-6 PSS から DFD への対応 *G*

DFD と業務取引システムの基本静的結合構造の差異は、DFD のデータフローに対応するところに必ず管理実体型が存在することである。これは光速通信の自社倉庫の場合のように、自社倉庫をデータフローごとに機能を分解してモデル化していることになる。

図 6-2 の基本静的結合構造を *G* によって DFD にすると図 6-7 の DFD となる。

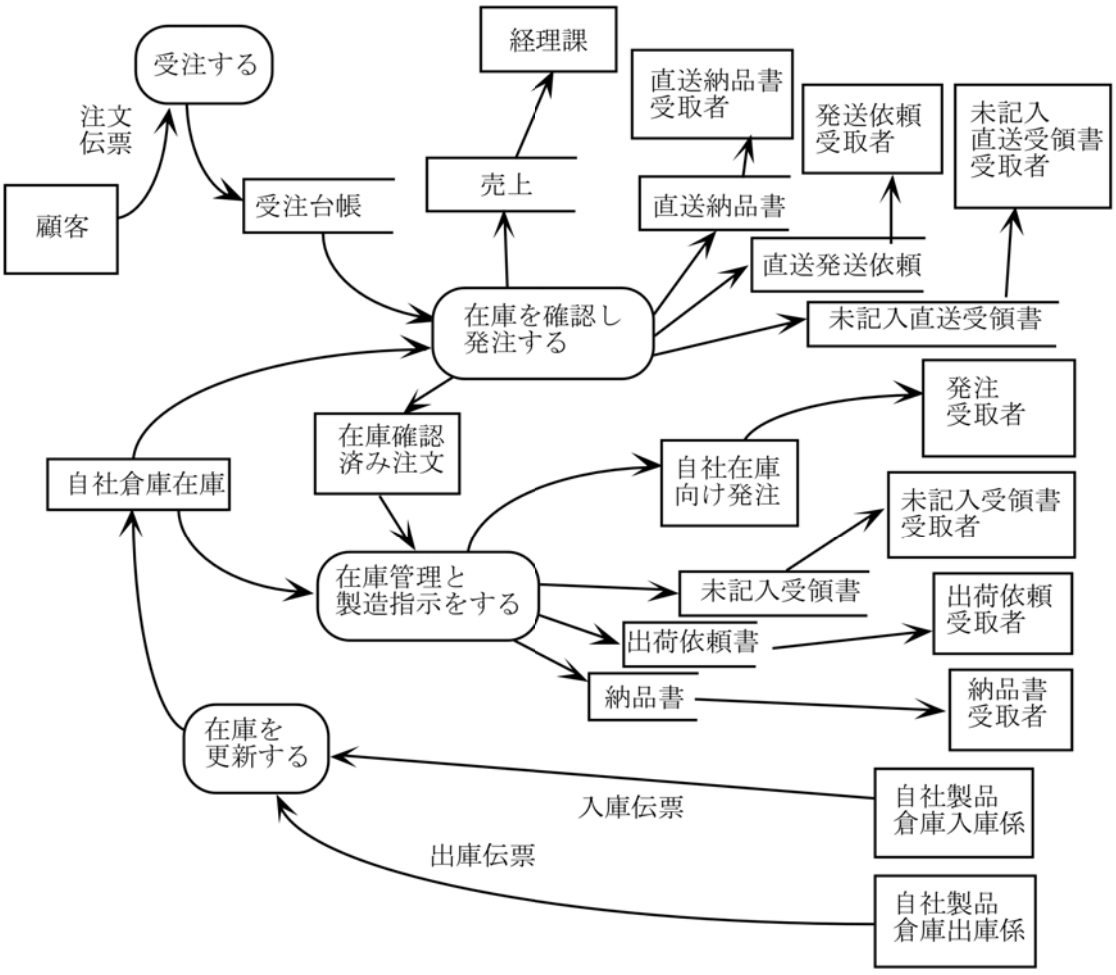


図 6-7　PSS から対応 *G* で得られた DFD

もし *G* によって基本静的結合構造から得られる DFD が基本静的結合構造の持つ情報を失うならば、基本静的結合構造のほうがより表現力を持っているとか、あるいは、基本静的結合構造というモデル化技法が冗長であるという可能性がある。幸運にも、以下のように、DFD から基本静的結合構造への実質的に逆の対応 *F* があることを示すことができる。

DFD から PSS への対応 *F* は図 6-8 で示される。

Dataflow diagram	Primitive static structure
ソース	外生トランザクション （ソースからのデータフローごと）
プロセス	内生トランザクション
データストア	管理実体型
データフロー	データフロー名の管理実体型を はさんだ結合の矢印
シンクとシンクへの矢印	消える
データフローと データストアの属性	対応する管理実体型の属性

図 6-8　DFD から PSS への対応 *F*

対応 *F* によって、図 6-7 の DFD は図 6-2 の基本静的結合構造に変換される。ただし、図 6-2 において直送品メーカー、納入メーカー、自社製品倉庫とそれへの矢印は除いたものに対応する。

対応 *F* は DFD のシンクを消す。ひとつのデータ源泉からいくつかのデータフローが出ている場合には、各データフローに対応して外生トランザクションを作る。DFD のソース（データ流入）は、ソースからのデータフローごとに分割する。物流管理部の例では、自社製品倉庫というひとつのソースが、入庫担当と出庫担当の２つのソースとして分けられる。この理由は、ソースは外部のものだが、異なるデータフローをだすのは異なる機能を持っているからで、トランザクション発生機能ごとにソースを外生トランザクションとして認識するということである。DFD のデータフローは枝分かれのあるなしにかかわらず管理実体型に対応する。最後に、データストアへ入ったり出ていったりするデータフローはそのデータストアの管理実体型と対応する。

基本静的結合構造から DFD を作る変換 *G* と、逆向きの変換 *F* が定義された。これらの変換は、いくつかの可能性があり、それらの中には、PSS の重要な情報を欠落して DFD を作るものがあるだろう。たとえば、ひとつの極端な変換として、PSS の持つトランザクションや管理実体型をすべて忘れて全体をただ一つの箱で表わしてしまう変換も PSS から DFD への変換としてはありうる。そのような変換はほとんど何の意味ももたらさない。DFD から PSS への変換も同様な問題を含む。ここまでに用意した２つの変換は、ある意味で実質的に互いに逆の

変換であることを示すことができる。

定義

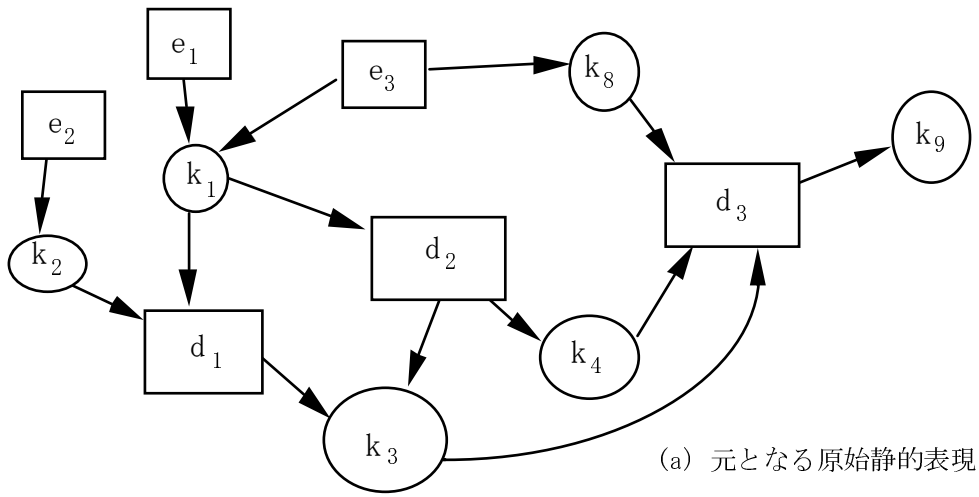
２つの原始静的構造があったとして、その間の名前付けの違いを無視した上で、それらが１対１に対応すれば等価である。

原始静的構造を構成する際に、外生トランザクション名、内生トランザクション名、管理実体型名、それらの定義に使われた属性名の４種類の名前付けが違うことがありうる。２つの原始静的構造が等価であるとは、これらの名前の付けかたが異なるだけで１対１に対応付けが可能なことと、トランザクションへの入力とトランザクションからの出力がその対応の元で全く同じことである。したがって２つの原始静的構造が等価な時には、２つの絵は位相的に（結合関係が）同じ関係を表わしていることである。

次節にあるように、次のことを証明できる。

命題 6-1

任意の原始静的構造を*G*によって DFD に変換しさらにそれを*F*によって原始静的構造に変換しても、それらは同値である。（図 6-9、図 6-10）



（a）元となる原始静的表現

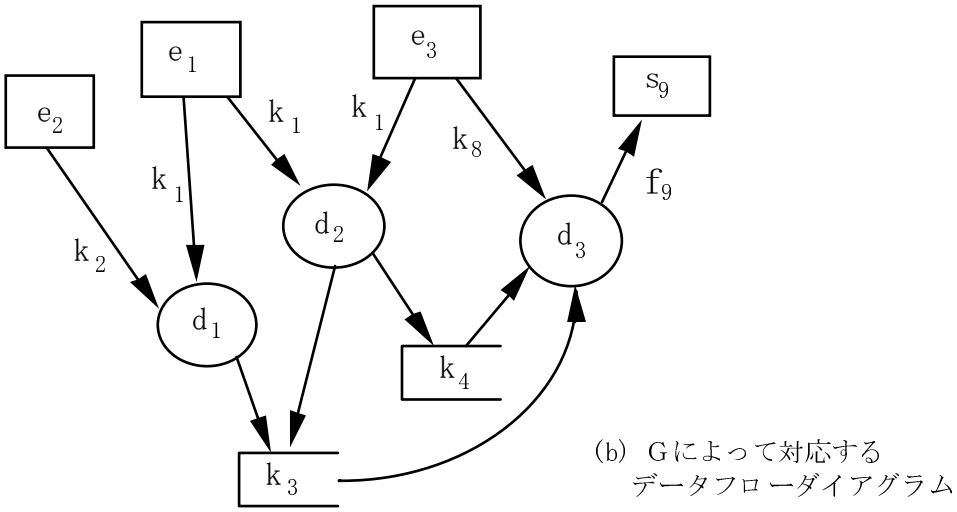

e1
e2
e3
s9
k1
k1
k8
f9
k1
d2
d3
k2
d1
k4
k3
(b) Gによって対応する
データフローダイアグラム

図 6-9　*G*の図的説明

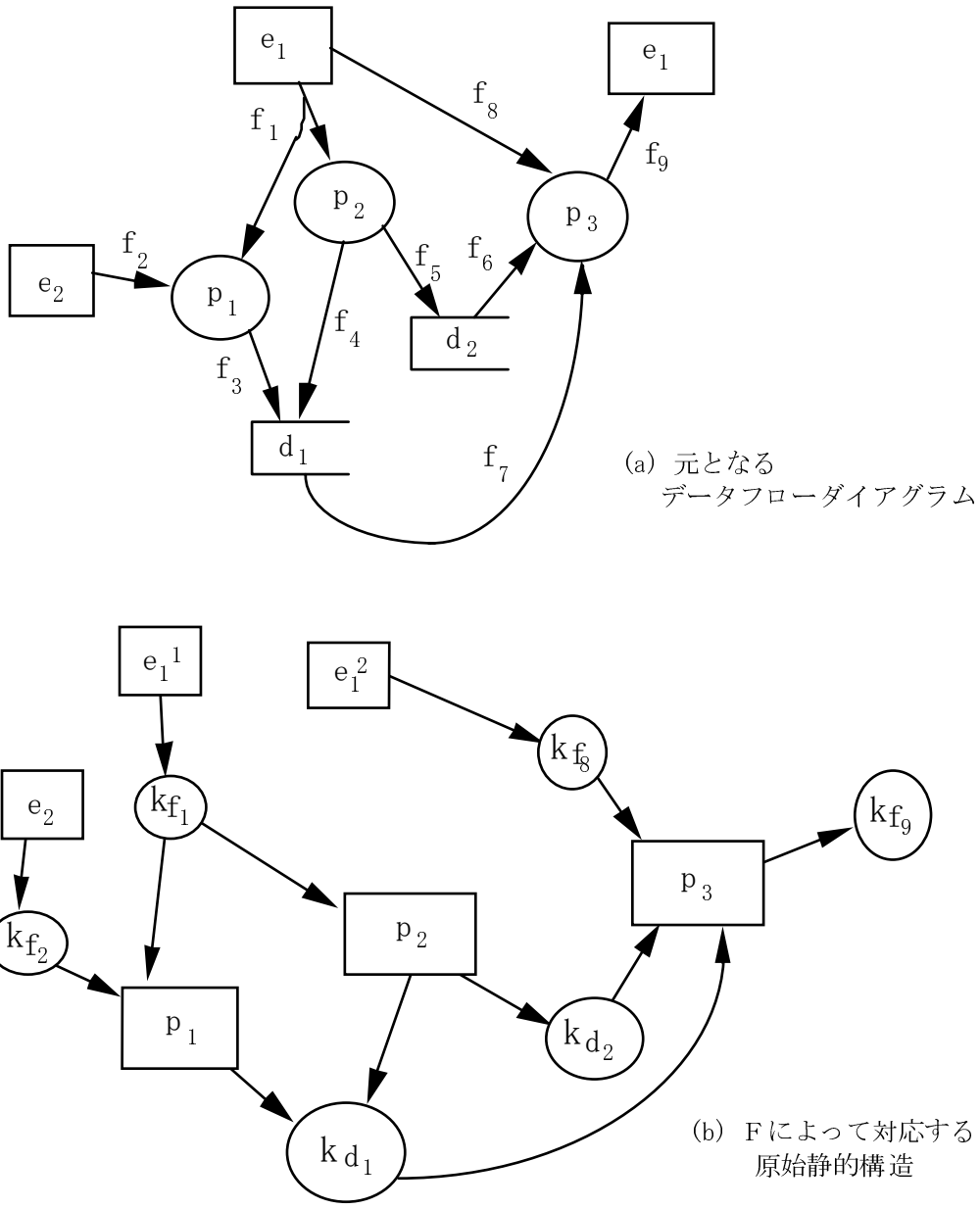


図 6-10　*F* の図的説明

DFD を *F* によって変換したものをさらに *G* によって DFD に変換したものは、もとの DFD とは等しくはない。光速通販の例の自社倉庫のように、基本静的結合構造では、外部トランザクションがデータを付け加える管理実体型はひとつだからである。

しかし、DFD を *F* によって変換した原始静的構造の分解能レベルで定型業務システムを考えるときには、それらが同じものである。

命題 6-2

任意の DFD を F で変換した業務システム S を考える。その得られた基本静的結合構造を G で変換した DFD をさらに F で PSS に変換した $F(G(S))$ は、元の基本静的結合構造の S と同値である。

上の命題 6-1 と 6-2 から、G と F は DFD を G で変換して得られる原始静的構造に対して互いに逆の対応となっていることがわかる。DFD と原始静的構造の間は実質的に 1 対 1 の関係があり、この意味で、両方ともに重大な情報を見逃すことのない自然なモデルであるといえる。

DFD によるモデル作りのために、業務取引システム理論と G と F という変換が与える示唆をつぎのようにまとめることができる。

（1） F によって DFD を原始静的構造に変換すると DFD のデータ吸収は消えてしまう。これはつまり、データ吸収が業務取引システムの動的側面とは関係がないからである。定型業務システムとしては、たとえば直送品メーカーに対して、直送発送依頼、直送納品書、未記入直送受領書を作成して届ければ、それ以後の作業は外部の組織である直送品メーカーが自らの責任で行うというビジネスルールになっている。したがって、直送納品書の作成以後のことは光速通販としては行う必要がない。

（2）DFD によって定型業務をモデル化するときに、内部活動の開始に関係のないデータフローは除いてもよい。なぜなら、業務取引システムが離散事象システムであるということと、F という変換とを通じて見えるのは、DFD のデータフローとデータストアが内部活動の開始条件を決めていることである。そして、その仕組みが組織内の定型業務の並行処理を可能にしているということである。

（3）データストアだけではなくデータフローもまた、実質的な管理実体型としてデータを保持することが言える。これは、データフローで表現されたデータが業務処理要求として蓄積され、その結果、業務取引システムの動作を決めるからである。

（4）データモデルの設計は DFD から自動的に出てくることはない。なぜなら、AID による原始静的構造と DFD の同型的対応によって、DFD のデータフローもデータストアも、原始静的構造の管理実体型を表していることがわかり、かつ、原始静的構造の定義が示す以上にはそれぞれの管理実体型の完全な属性情報は

持っていないからである。データモデルはDFDによるビジネスプロセスの外観設計と関連はあるが、DFD がデータモデルを完全に決定することはないので、第３章で述べたことがらを踏まえて、データモデルの設計は現実世界のイベントや関連資源をとらえるために必要な属性をモデル化する必要がある。
（５）DFD によるモデル化について、シンクやソースが直接にはデータストアと接続することを禁じている場合がある。その理論的理由はなく単に主張されている。しかし現実を見ると、たとえば注文のファクスが重なってはきだされていることもあるだろうし、ネットワークを介した EDI 伝票によって、引き合いや注文がデータベースにストアされることがある。支払い関連の業務でも同様である。こうしたことに加えて、本節で示した理論的な同型性からするとその主張は直接接続を禁止する必要はない。つまり、外部のシンクやソースがデータフローを介してデータストアに直接に接続することは自然なモデルであって問題はない。

6.4 データフローダイアグラムと業務取引システムとの自然な対応の証明

前節はDFDの意味論であった。つまり、互いに同型になるようなアクティビティ・インタラクション・ダイアグラム（AID）と DFD のクラスを提示し、命題 6-1 と 6-2 によって同型性を表現した。本節で情報システム方法論のツールの意味論の例として、集合論を用いてそれらの命題の形式的証明を行う。同時に、情報システムツールの意味を論じるときの例題としての位置づけを持つ。AIDと DFD の相互関係について実践的な説明は前節ですんでいるので、証明に興味のない場合には省略し、本章 5 節のエンティティ・ライフ・ヒストリーの意味論に進んでよい。

6.4.1 業務取引システムの原始静的構造

第４章４節で定めた業務取引システムの構成にある条件を課して、後にデータフロー・ダイアグラム（DFD）と同型になるようなアクティビティ・インタラクション・ダイアグラムを定める。

定義　業務取引システムの原始静的表現 (promitive static structure)

ファイルシステムの一部である管理実体型と属性集合、および外生トランザクション、内生トランザクションからなる組み$\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA}\rangle$が次の３つの条件を満足するときに業務取引システムの静的表現という；

1)外生トランザクションと内生トランザクションの影響の分離：

すべての$e \in E$と$a \in A$ に対して$f_{EK}(e) \cap f_{AK}(a) = \phi$ (空集合)である。

2)任意の管理実体型は外生トランザクションか内生トランザクションの影響をうける：

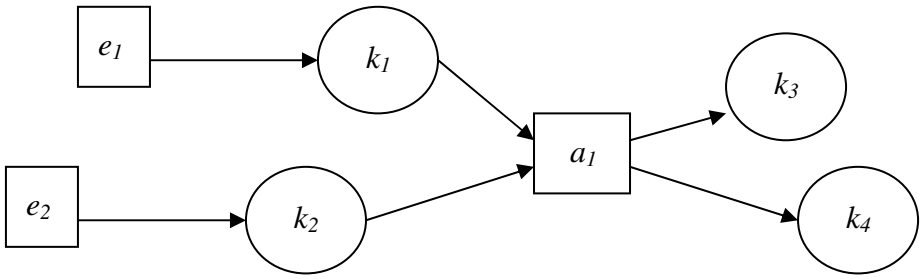
任意の$k \in (KS - \{k' | k' \in f_{EK}(e), e \in E\})$に対してはある$a \in A$があって$k \in f_{AK}(a)$が成立する。

3)外生トランザクションの発生が記録される管理実体型は活動開始に影響をあたえる管理実体型である：

任意の$k \in \{k' | k' \in f_{EK}(e), e \in E\}$に対してある$a \in A$があって $k \in f_{KA}(a)$である。

業務取引システムの静的表現は、トランザクション処理としての業務処理に関わる外部や内部の存在と、使われるデータファイルの構成を記述するものであった。原始静的表現では、データが満たすべき参照関係と継承関係という整合性条件を持つファイルシステム（$File$）やファイル更新関数（$f_{FinFileVal}$と$f_{StFileVal}$）、ファイルの値から割り当てトランザクションの開始を指定する関数（内部活動選択関数$f_{Dispatch}$ ）は無視されている。また、原始静的表現が満たすべき３つの条件はいわば当たり前の条件であって表現が内生と外生のトランザクションが異なる管理実体型に記録されることと、無駄な管理実体型をもたないことを要請している。

静的表現は動的表現が定める離散事象システムで使われる情報の外観を定めるものである。業務取引システムの原始静的表現はアクティビティ・インタラクション・ダイアグラムで表現できる（図 6-11）。管理実体型の属性は図では表現されない。



$$E=\{e_1,e_2\},A=\{a_1\},f_{EK}(e_1)=\{k_1\},f_{EK}(e_2)=\{k_2\},$$
$$f_{KA}(a_1)=\{k_1,k_2\},f_{AK}(d_1)=\{k_3,k_4\}$$

図 6-11 原子静的構造(PSS)

6.4.2 データフローダイアグラム

6.4.2.1 データフローダイアグラムの特徴

データフローダイアグラムは４種類の基本要素と付随するデータによって記述される有向グラフである。プロセス、データストア、データ源泉、データ吸収を表す図形には著者によってちがいがあり統一されていないが、４種類の図形で各々の基本要素を表現するということは同じである。

実際にデータフローダイアグラムを使う場合には、階層化して何枚かのデータフローダイアグラムによってひとつの業務システムを表現する。たとえば、まず大まかに機能がプロセスとして取り出されデータフローによって結合される。更に詳細化される必要のあるプロセスが、いくつかのプロセスから成るデータフローダイアグラムにブレークダウンされる。最終的には、論理的に１枚の大きなデータフローダイアグラムとなるようにある整合性をもって階層化されたデータフローダイアグラム群が、全体として１つの業務システムを記述する。図 6-12 は病院の医療材料管理業務を記述した例である。

本節ではデータフローダイアグラムの使用上の利点などを論じるのではなくそれが定型業務システムの何を記述するのかを考察するので、データフローダイアグラムはこうした最終的な詳細レベルの全体的な「１枚の」図的表現を意味するものとし、階層構造は考えない。

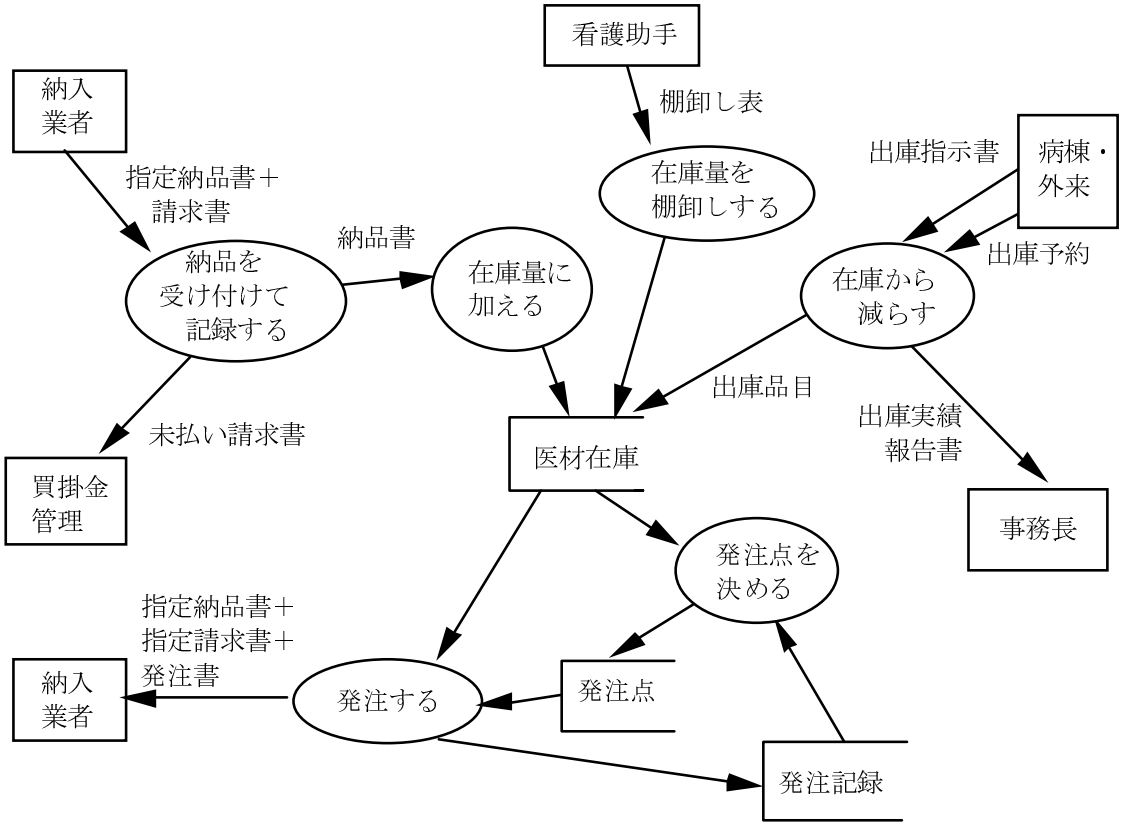


図 6-12　医材管理業務のデータフローダイアグラム

以下にデータフローダイアグラムの特徴を、DeMarco (1979), Gane and Sarson (1979), Page-Jones(1980), ピーターズ(1987) , Cutts (1987), ダウンズ他(1988), McDermid (1990) から簡単にまとめる。

（１）業務システムをどの観点から何を記述するかについて：

ある状態を人や組織（たとえばデータ処理担当者）の観点からではなくデータの観点から表示する。

業務システムの有効な分割を表現する。

業務システムの中で自動化の範囲を決定する。

データフローに帳票名を入れた物理データフローダイアグラムとして業務システムをモデル化することもでき、また、データ項目を入れた論理データフローダイアグラムとしても使うことができる。

データフローはデータという荷物を送るパイプと考えてよい。数学的抽象化ではなく、配管を扱っている。

「このトランザクションが発生するには何が必要か」を考えてデータフローやプロセスをかく。

（２）分析作業上の利点と意義について：

　データフローダイアグラムは、図形表示されている、階層化されている、分割されている、多次元的である、データの流れを強調している、コントロールの流れに重点をおいていない。

図形的表現なのでユーザがわかる。したがってデータフローダイアグラムによるモデルをウォークスルーによって検査できる。

補完的にデータ構造の記述が必要とされるため、データディクショナリが用いられる。そこでは、データフロー、データフローの構成要素、データストア、プロセスが定義される。

（３）記述しないもの：

データフローダイアグラムはデータの流れを示すものであってコントロールの流れを示すフローチャートではない。また、ある処理を開始するタイミング、エラー処理、例外処理は記述されない。

ある時にしか使われないデータもデータフローダイアグラムに記述される。

6.4.4.2　データフローダイアグラムの定式化

データフローダイアグラムは、外部のデータ源やデータ吸収を含む最も詳しいレベルの全体的記述を考える。

定義　データフローダイアグラム (data flow diagram)

Sink ：データ吸収の集合で有限とする。

Source ：データ源泉の集合で有限とする。

Process ：プロセスの集合で有限とする。

Datastore ：データストアの集合で有限とする。

Attribute: 属性の集合で有限とする。その要素を属性と呼ぶ。

以上の５つの集合は互いに共通要素を持たないものとする。

$Arrow \subseteq (Source \times Process) \cup (Process \times Process) \cup (Process \times Sink) \cup$
$(Process \times Datastore) \cup (Datastore \times Process)$

：矢印の集合であって次の条件を満足する。

1)任意の $s \in Sink$ に対して $(p, s) \in Arrow$ となるプロセス p がある；

2)任意の $s \in Source$ に対して $(s, p) \in Arrow$ となるプロセス p がある；

3) $Process = \{p \mid (\exists f, f' \in Arrow)(\exists j, j' \in Process \cup Source \cup Sink \cup Datastore)$
$(f = (j, p) \wedge f' = (p, j'))\}$ が成立すること。

4) $Datastore = \{j \mid (\exists f, f' \in Arrow)(\exists p, p' \in Process)(f = (p, j) \wedge f' = (j, p'))\}$ が成立すること。

$a : Arrow \cup Datastore \to BusinessPapers$, $Businesspapers = P(Attribute)$

関数 a は矢印が持つ属性の組と、データストアが持つ属性の組みを指定する関数であって属性関数と呼ばれる。

任意の $f = (p,d) \in Arrow \cap (Process \times Datastore)$ に対して $a(f) \subseteq a(d)$ という条件を満足する。

$DataFlow = \{(f, a(f) \mid f \in Arrow\}$: データフローは矢印と対応する属性の組みである。

データフローダイアグラムは

$\langle Sink, Source, Process, Datastore, Arrow, DataFlow, a \rangle$

という組である。

6.4.3 データフローダイアグラムから業務システムの原始静的構造への対応

dfd=$\langle Sink, Source, Process, Datastore, Arrow, DataFlow, a \rangle$ に対して以下のように定められる業務取引システムの原始静的構造 pss= $\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA} \rangle$ が存在する。この対応を F(dfd) = pss と書き、F(dfd)をdfdが定める原始静的構造と呼ぶ。

(1) $\hat{E} = \{e_f \mid f \in Arrow, f = (s, p) \in Source \times Process\}$ という集合を考え、その上の同値類を次のように定める：

$(e_f, e_{f'}) \in \equiv$ iff ある $s \in Source$ と $p, p' \in Process$ があって $f = (s, p) \in Arrow$ かつ $f' = (s, p') \in Arrow$ かつ $a(f) = a(f')$ であること。

このとき外部トランザクション集合を次で定める。

$E = \hat{E} / \equiv$

：データ源泉から出ていく矢印のうち、属性が異なる矢印の数だけの外部トランザクションがあることになる；

(2)

$KS = \{k_d \mid d \in Datastore\}$

$\cup \{k_f \mid f \in Arrow \wedge f \in (Process \times Process) \cup (Process \times Sink)\} \cup \{k_{[e]} \mid [e] \in E\}$

：データストアと、データストアに関係しないデータフローと、外部トランザクションの各々に対応して管理実体型がある。

(3) $AS = Attribute$

(4) $A = Process$

(5) 管理実体型が持つ属性：

(i) $d \in Datastore$のとき $attr(FT(k_d)) = a(d)$

(ii) $f \in Arrow \wedge f \in (Source \times Process) \cup (Process \times Process) \cup (Process \times Sink)$のとき $attr(FT(k_f)) = a(f)$

(iii) $[e] \in E$に対して$e_f \in [e]$ のとき、 $attr(FT(k_{[e]})) = a(f)$ とする。

(6) f_{EK}の定義

任意の$[e] \in E$に対して$f_{EK}([e]) = \{k_{[e]}\}$：異なるデータフローに対して異なる発生源があるような業務システムの静的構造として考える。

(7) f_{AK}の定義

任意の$p \in A$に対して

$f_{AK}(p) =$

$\{k_d \mid \exists d \in Datastore, (p,d) \in Arrow\} \cup \{k_f \mid \exists j \in Process \cup Sink, f = (p,j) \in Arrow\}$

：活動$p \in A$によって影響を受ける管理実体型は、プロセスpの出力となっているデータストアやアローに対応するものである。

(8)　f_{KA}の定義

任意の$p \in A$に対して

$f_{KA}(p) = \{k_d \mid \exists d \in Datastore, (d,p) \in Arrow\}$

$\cup \{k_f \mid \exists j \in Process, f = (j,p) \in Arrow\}$

$\cup \{k_{[e]} \mid \exists s \in Source, f = (s,p) \in Arrow \wedge e_f \in [e]\}$

：活動$p \in A$の開始を決める管理実体型は、プロセスpの入力となっているデータストアやアローやソースに対応するものである。

上記定義において、管理実体型の属性は $[e] \in E$ に対して$e_f \in [e]$ のとき、

$attr(FT(k_{[e]})) = a(f)$ と well-defined に定まる。実際、$e_f, e_{f'} \in [e]$ とすると $f = (s,p), f' = (s,p') \in Arrow \cap (Source \times Process)$ であって $a(f) = a(f')$ である。

図 6-6 と図 6-10 は F を説明したものであった。データ吸収($Sink$) は F によってなくなる。データフローダイアグラムでは外部存在が物理的にひとつであることから、ひとつのデータ源からいくつかのデータフローがでてくることがありうるが、静的構造では 1 つのデータ発生機能ごとにひとつの外生トランザクションとして対応づける。ただし枝分かれするものは同一視する（図 6-10 の f_1）。また、f_3, f_4 の場合のようにデータフローの属性はデータストアの属性として一括されてしまう。

つぎに示すように、F による対応は確かに原始静的構造である。

命題 6-3

任意の $\mathtt{dfd} = \langle Sink, Source, Process, Datastore, Arrow, DataFlow, a \rangle$ に対して $F(\mathtt{dfd})$ は原始静的構造である。さらに、任意の $e \in E$ に対して $f_{EK}([e])$ はただひとつの要素だけからなる。

（証明）

任意に $\mathtt{dfd} = \langle Sink, Source, Process, Datastore, Arrow, DataFlow, a \rangle$ をとる。$F(\mathtt{dfd})$ が原始静的構造であるためには、F によって定まる $KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA}$ が以下の 3 つの条件を満たすことを示せばよい。

1)すべての $e \in E$ と $p \in A$ に対して $f_{EK}(e) \cap f_{AK}(p) = \phi$ (空集合)である。

2)任意の $k \in (KS - \{k' \mid k' \in f_{EK}(e), e \in E\})$ に対してはある $p \in A$ があって $k \in f_{AK}(p)$ が成立する。

3)任意の $k \in \{k' \mid k' \in f_{EK}(e), e \in E\}$ に対してある $p \in A$ があって $k \in f_{KA}(p)$ である。

1)について：

任意の $[e] \in E$ と $p \in A$ をとる。$f_{EK}([e]) = \{k_{[e]}\} \subseteq \{k_{[e]} \mid [e] \in E\}$ である。

$\hat{E} = \{e_f \mid f \in Arrow, f = (s,p) \in Source \times Process\}$, $E = \hat{E}/\equiv$ であり、一方、F の定義から $f_{AK}(p) =$

$\{k_d \mid \exists d \in Datastore, (p,d) \in Arrow\} \cup \{k_f \mid \exists j \in Process \cup Sink, f = (p,j) \in Arrow\}$

なので、$f_{EK}(e)$と$f_{AK}(p)$の要素はもとのデータフロー図において必ず異なったデータフローである。したがって$f_{EK}(e)\cap f_{AK}(p)=\phi$ である。

2)について：

任意の$k\in(KS-\{k'|k'\in f_{EK}(e),e\in E\})$をとる。

$k\in\{k_f\mid f\in Arrow\wedge f\in(Process\times Process)\cup(Process\times Sink)\}$とすると、ある$p,p'\in Process$と$s\in Sink$があって$k=k_f$となるような$f=(p,p')\in Arrow$ または$f=(p,s)\in Arrow$ があるので、$k\in f_{AK}(p)$を得る。

また、$KS=\{k_d\mid d\in Datastore\}$とすると、$k=k_d$となるような$d\in Datastore$があり、データフローダイアグラムの矢印集合の条件から、$(p,d)\in Arrow$ となる$p\in Process$がある。したがって$k\in f_{AK}(p)$を得る。

3)について

任意の$k\in\{k'|k'\in f_{EK}(e),e\in E\}$をとる。つまり、ある$e\in E$があって$f_{EK}(e)=\{k_e\}$である。$E$の定義から、$[e_f]=e$となるような$f=(s,p)\in Arrow\cap(Source\times Process)$がある。$F$の$f_{KA}$の定義から$k_{[e]}\in f_{KA}(p)$となる。（証明終）

6.4.4　業務システムの原始静的構造からデータフローダイアグラムへの対応

業務取引システムの原始静的表現 pss=$\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA}\rangle$に対して以下のように定められる dfd=$\langle Sink$，$Source$，$Process$，$Datastore$，$Arrow$，$DataFlow, a\rangle$が存在する。この対応を$G$(pss)=dfdと書き、dfdをpssが定めるデータフローダイアグラムと呼ぶ。

(1) $Source=E$,

(2) $Process=A$,

(3) $Sink=\{s_k\mid k\in KSterminal\}$, ただし$KSterminal=\{k\mid\forall p\in A, k\notin f_{KA}(p)\}$と定める。

(4) $Datastore=KS-(KSterminal\cup K_E)$, ただし$K_E=\{k\mid\exists e\in E, k\in f_{EK}(e)\}$と定める。また、任意の$d\in Datastore$に対して$a(d)=attr(FT(d))$と定める。

(5) $Attribute=AS$

(6) 以下の条件を満足する集合として$Arrow$と属性関数aを定める。

6-1) $f_{EK}(e)=\{k_1,k_2,...,k_n\}$のとき、$k_i$に対してある$p\in A$があって$k_i\in f_{KA}(p)$であるとき、$f=(e,p)\in Arrow\wedge a(f)=attr(FT(k_i))$とする。

6-2) $p \in A$に対して$f_{AK}(p)=\{k_1,k_2,...,k_n\}$のとき、各$k_i$について次のように定める：

(i) $k_i \notin KSterminal$ならば$f_i=(d,k_i) \in Arrow \wedge a(f_i)=\phi$（空）と定める。

(ii) $k_i \in KSterminal$ならば、$f_i=(d,s_{k_i}) \in Arrow \wedge a(f_i)=attr(FT(k_i))$である。末端をデータ吸収にするということである。

6-3) $p \in A$に対して$f_{KA}(p)-K_E=\{k_1,k_2,...,k_n\}$のとき$f_i=(k_i,a) \in Arrow$, $a(f_i)=\phi$, $a(k_i)=attr(FT(k_i))$と定める。

定義の意味は次のようである。(1)外生トランザクションはデータ源泉である。(2)内生トランザクションはプロセスである。(3) 内生トランザクションへ f_{KA}を示す矢印が出ていない管理実体型があれば、それに対応するデータ吸収がある。(4)データストアはデータ吸収でもなく外生トランザクションの出力でもないような管理実体型である。(5)属性集合は変わらない。(6-1) 外生トランザクションから f_{EK}で関係付けられた管理実体型があるとき、その関係に対応してデータ源泉からデータストアへのデータフローがある。データフローの属性は管理実体型のそれである。(6-2-i) 内生トランザクションがf_{AK}によって影響（出力）する管理実体型があれば、その関係に対応してプロセスからデータストアへのデータフローがある。(6-2-ii) さらにその管理実体型から出ていく矢印がないときは、(3)で定まるデータ吸収へのデータフローがある。データフローの属性を適切に定める。(6-3) 管理実体型から内生トランザクションp へ入る矢印はデータフローである。データフローの属性は空である。

命題 6-4

任意の原始静的構造pssに対してG(pss)はデータフローダイアグラムである。さらに$G(\mathtt{pss})=\langle Sink, Source, Process, Datastore, Arrow, DataFlow, a\rangle$とするとき、$Arrow \cap (Process \times Process)=\phi$（空集合）である。

（証明）

任意にとった原始静的表現$\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA}\rangle$に対して$G$が定める各集合$Sink, Source, Process, Datastore, Arrow, DataFlow, a$がデータフローダイアグラムの条件を満たすことを示せばよい。

$Arrow$に関する条件が成立することを示す。

(1) *Source*については、関数 f_{EK} が $f_{EK}: E \to P(KS)^+$ なので *Arrow* に関する条件を満たす。

(2) *Sink*について：任意の $s_k \in Sink = KSterminal$ をとる。すべての $p \in A$ について $k \notin f_{KA}(p)$ である。したがって、原始静的構造の条件 3)より $k \notin \{k' \mid k' \in f_{EK}(e), e \in E\}$ である。原始静的構造の条件 2)からある $p' \in A$ があって $k \in f_{AK}(p')$ 成立する。したがって G の定義 6-2)の ii) から $f = (p', s_k) \in Arrow$, $a(f) = attr(FT(k))$ となる。

(3) *Process*について：

$Process = \{p \mid (\exists f, f' \in Arrow)(\exists j, j' \in Process \cup Source \cup Sink \cup Datastore)$
$(f = (j, p) \wedge f' = (p, j'))\}$ が成立することを示せばよい。

(3-1) 任意の $p \in Process$ をとる。 $f_{KA}(p) \neq \phi$ だから、 $k \in f_{KA}(p)$ であるような $k \in (KS - KSterminal)$ がある。ここで $K_E = \{k' \mid \exists e \in E, k' = f_{EK}(e)\}$ とおく。

$k \in K_E$ のときは、ある $e \in E$ について $k = f_{EK}(e)$ なので
$f = (e, p) \in Arrow, a(f) = attr(FT(k))$ である。

$k \in (KS - KSterminal - K_E)$ のときは、 $f = (k, p) \in Arrow$ かつ
$a(f) = \phi \subseteq a(k) = attr(FT(k))$ である。

また、 $f_{AK}(p) \neq \phi$ より、 $k \in f_{AK}(p)$ となる $k \in (KS - K_E)$ がある。 $k \in KSterminal$ のときは $f = (p, s_k) \in Arrow \cap (Process \times Sink)$, $a(f) = attr(FT(k))$ である。

$k \in (KS - K_E - KSterminal)$ のときは $f = (p, k) \in Arrow \cap (Process \times Datastore)$,
$a(f) = \phi \subseteq attr(FT(k))$ である。

(3-2) 逆の包含関係も明らかに成立する。

(4) *Datastore*について：

$Datastore = \{j \mid (\exists f, f' \in Arrow)(\exists p, p' \in Process)(f = (p, j) \wedge f' = (j, p'))\}$ が成立することを示せばよい。

(4-1) 任意の $k \in Datastore$ をとる。 $Datastore = KS - (KSterminal \cup K_E)$ だから、
$k \in f_{KA}(p)$ であるような $p \in Process$ がある。したがって、 $(k, p) \in Arrow$,
$a(f) = \phi \subseteq attr(FT(k))$ である。

また、原始静的構造の条件 2)からある $p \in Process$ があって $k \in f_{KA}(p)$ が成立する。したがって G の定義から $f = (p, k) \in Arrow$, $a(f) = \phi \subseteq a(k) = attr(FT(k))$ である。

(4-2) 逆の包含関係も明らかに成立する。

(5) 以上から、G(pss)はデータフローダイアグラムである。さらに、Gの定義か

ら $Arrow \cap (Process \times Process) = \phi$ である。
（証明終）

6.4.5 データフローダイアグラムの意味

定型的業務システムは業務取引システムという状態表現を考えることで離散事象システムとしてとらえることができた。業務取引システムの静的構造からさらに情報を減らした原始静的構造とデータフローダイアグラムの関係が F と G という対応で示された。本節ではさらに、が F と G との関係を明かにし、データフローダイアグラムを業務取引システムから意味づける。

定義　同形な原始静的構造

2 つの原始静的構造 pss= $\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA} \rangle$ ， pss'= $\langle KS', AS', attr', E', A', f'_{EK}, f'_{AK}, f'_{KA} \rangle$ が同形であるとは、以下の 4 条件を満たすような 1 対 1 対応 h_0, h_1, h_2, h_3 が存在することである：

$h_0 : AS \to AS'$， $h_1 : KS \to KS'$， $h_2 : E \to E'$， $h_3 : A \to A'$

1) $h_0 \cdot attr \cdot FT = attr' \cdot FT' \cdot h_1$、このとき $attr \cong attr'$ とかく；
2) $h_1 \cdot f_{EK} = f'_{EK} \cdot h_2$、このとき $f_{EK} \cong f'_{EK}$ とかく；
3) $h_1 \cdot f_{AK} = f'_{AK} \cdot h_0$、このとき $f_{AK} \cong f'_{AK}$ とかく；
4) $h_1 \cdot f_{KA} = f'_{KA} \cdot h_0$、このとき $f_{KA} \cong f'_{KA}$ とかく；

pss と pss'が同形なときに pss≅pss'とかく。

同形な原始静的構造は名前の付け方が表面上異なるだけであり、実質的に同じものである。

命題 6-5

任意の原始静的構造 pss= $\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA} \rangle$ において、任意の $e \in E$ に対して $f_{EK}(e)$ がただ 1 つの要素からなるとする。このとき、$F \cdot G(\text{pss}) \cong \text{pss}$ である。

（証明）

任意に原始静的構造pss=$\langle KS, AS, attr, E, A, f_{EK}, f_{AK}, f_{KA}\rangle$をとる。$G(\mathrm{pss})=\langle Sink, Source, Process, Datastore, Arrow, DataFlow, a\rangle$，$F\cdot G(\mathrm{pss})=\langle KS', AS', attr', E', A', f'_{EK}, f'_{AK}, f'_{KA}\rangle$とする。$F$、$G$の定義から$A=A'$，$AS=AS'$である。

$E\cong E'$ を示す。

$h_2: E\to E'$を$h_2(e)=[e_f]$ と定める。ただしある$p\in A$ があって$f=(e,p)$，$f_{EK}(e)\subseteq f_{KA}(p)$である。実際このような$[e_f]$は常に存在する。任意の$e\in E$をとり、$f_{EK}(e)=\{k\}$とする。ある$p\in A$があって$k\subseteq f_{KA}(p)$である。したがって$G$による対応で $f=(e,p)\in Arrow\cap(Source\times Process)$，$a(f)=attr\cdot FT(k)$である。これから、$F$によって$[e_f]\in E'$ が存在する。

h_2 が well- defined であること、つまり代表元e_f のとりかたに依存しないことを示す。 $p,p'\in A$ があって$f=(e,p)$，$f'=(e,p')\in Arrow\cap(Source\times Process)$とする。どちらも$a(f)=a(f')=attr\cdot FT(k)$であるから$[e_f]=[e_{f'}]$ である。

h_2が全射であることを示す。

任意の$[e]\in E'$ をとる。すると$[e]=[e_f]$ となるような

$f=(e,p)\in Arrow\cap(Source\times Process)$がある。このデータフロー$f$ はGによって生じたのだからある$s\in E$があって$f_{EK}(s)=\{k\}$かつ $k\in f_{KA}(p)$である。よって$h_2(s)=[e_f]=[e]$である。

h_2が単射であることを示す。$h_2(s)=h_2(s')$とする。すると$h_2(s)=[e_f]$，$h_2(s')=[e_{f'}]$となるような$f=(s,p)$，$f'=(s',p')\in Arrow\cap(Source\times Process)$ がある。いま$[e_f]=[e_{f'}]$ だから E' の同値関係の定義より$s=s'$ である。以上よりh_2は１対１対応である。

$KS\cong KS'$ を示す。

$K_E=\{k\mid\exists e\in E, k\in f_{EK}(e)\}$とする。$Datastore=KS-(KSterminal\cup K_E)$である。$F$の定義より

$KS'=\{k_d\mid d\in Datastore\}\cup\{k_f\mid f\in Arrow\cap(Process\times(Process\cup Sink))\}$

$\cup\{k_{[e]}\mid[e]\in E'\}$であるが、命題２より$G(\mathrm{pss})$においては

$Arrow\cap(Process\times Process)=\phi$だから

$KS'=\{k_d\mid d\in Datastore\}\cup\{k_f\mid f\in Arrow\cap(Process\times Sink)\}\cup\{k_{[e]}\mid[e]\in E'\}$

である。KS, KS' を構成する３つの集合は互いに共通部分を持たないことから、

$h_1: KS\to KS'$を$\psi_1: KS-\{KSterminal\cup K_E\}\to\{k_d\mid d\in Datastore\}$，

$\psi_2: KSterminal\to\{k_f\mid f\in Arrow\cap(Process\times Sink\}$，$\psi_3: K_E\to\cup\{k_{[e]}\mid[e]\in E'\}$の

組み合わせで定める。ここで、

$h_1(d)=\psi_1(d)=k_d$, $d\in Datastore$のとき；

$h_1(j)=\psi_2(j)=k_f$, $j\in KSterminal$のとき。ただし、ある$p\in Process, k_j\in Sink$があって$f=(p,k_j)$；

$h_1(j)=\psi_3(j)=k_{[e']}$, $j\in K_E$のとき。ただし、ある$e\in E$があって$f_{EK}(e)=j$かつ$h_2(e)=[e']$である。

上の定義でψ_2,ψ_3は well-defined である。実際、任意にとった$j\in KSterminal$に対して$f=(p,s)\in Arrow\cap(Process\times Sink)$はただひとつだけ存在するから$\psi_2$ は well-defined である。また、任意にとった$j\in K_E$に対して $f_{EK}(e)=j$となる$e\in E$はただひとつだけ存在するからψ_3は well-defined である。

h_1が１対１対応であることを示すには、ψ_1,ψ_2,ψ_3 がそれぞれ１対１対応であることをいえばよい。ψ_1は明かに１対１対応である。ψ_2 が単射であることを示すため、$k_f=\psi_2(j)=\psi_2(j')=k_{f'}$とする。このとき、ある$f=(p,k_j), f'=(p',k_{j'})\in Arrow\cap(Process\times Sink)$があって$k_f=k_{f'}$である。したがって$f=f'$であり$j=j'$となる。$\psi_2$ が全射であることを示すため、任意に$k\in\{k_f \mid f\in Arrow\cap(Process\times Sink)\}$をとる。するとある$j\in KSterminal$があって$f=(p,s_j)\in Arrow\cap(Process\times Sink)$, $k_f=k$である。したがって$\psi_2(j)=k_f$である。

ψ_3が単射であることを示すため、$\psi_3(j)=\psi_3(j')$とする。ある$e,e'\in K_E$があって$f_{EK}(e)=\{j\}$, $f_{EK}(e')=\{j'\}$かつ$h_2(e)=h_2(e')$である。このとき、ある$f=(e,p), f'=(e',p')\in Arrow\cap(Source\times Process)$があって、$h_2(e)=[e_f]$, $h_2(e')=[e_{f'}]$である。したがって同値関係の定義から$e=e'$ を得るので、$j=j'$ となる。

ψ_3 が全射であることを示すため、任意に$k_{[e']}\in\{k_{[e]} \mid e\in E'\}$をとる。ある$e\in E$があって$h_2(e)=[e']$となるので、$f_{EK}(e)=\{j\}$とおく。すると$\psi_3(j)=k_{[e']}$である。

以上からh_1は１対１対応である。

$A\cong A'$を示す。

任意の$k\in KS$ に対して$attr'\cdot FT'(h_1(k))=attr\cdot FT(k)$を示せばよい。

$K_D=KS-\{KSterminal\cup K_E\}$とおくとき、$KS=K_D\cup KSterminal\cup K_E$ であり、各集合は互いに共通部分を持たない。

任意の $d\in K_D$に対して$attr'\cdot FT'(h_1(d))=attr'\cdot FT'(k_d)=a(d)=attr\cdot FT(d)$である。

任意の$j\in KSterminal$に対して$attr'\cdot FT'(h_1(j))=attr'\cdot FT'(k_f)$である。ただし、$f=(p,s_j)\in Arrow\cap(Process\times Sink)$である。$F$と$G$の定義から

$attr' \cdot FT'(k_f) = a(f) = attr \cdot FT(j)$である。
任意の$j \in K_E$に対して$attr' \cdot FT'(h_1(j)) = attr' \cdot FT'(k_{[e']})$である。ただし、ある$e \in E$, $p \in Process$があって$[e'] = [e_f]$, $f_{EK}(e) = \{j\}$,
$f = (e, p) \in Arrow \cap (Source \times Process)$である。$F$と$G$の定義から
$attr' \cdot FT'(k_{[e']}) = a(f) = attr \cdot FT(j)$である。以上から$A \cong A'$である。
$f_{EK} \cong f'_{EK}$を示す。任意の$e \in E$をとる。$f'_{EK}(h_2(e)) = f'_{EK}(e')$である。ただし、ある$f = (e, p)$ があって$e' = h_2(e) = [e_f]$, $f_{EK}(e) = \{k\} \subseteq f_{KA}(p)$である。すると$h_1$の定義から$h_1(k) = k_{e'}$ である。一方、Fの定義より$f'_{EK}(e') = \{k_{e'}\} = \{h_1(k)\} = h_1 \cdot f_{EK}(e)$を得る。つまり$f_{EK} \cong f'_{EK}$を得る。
$f_{AK} \cong f'_{AK}$を示す。
任意の$p \in D$をとり、$f_{AK}(p) = \{k_1, k_2, ..., k_n\}$とする。一般性を失うことなく、ある$j, 1 \le j \le n$, に対して、$1 \le i \le j$のような i について$k_i \in Datastore$であり、$j+1 \le i \le n$のようなiについて$k_i \in KSterminal$としてよい。$k_i \in Datastore$の場合は$f_i = (p, k_i) \in Arrow$であり, $k_i \in KSterminal$の場合は$f_i = (p, s_{k_i}) \in Arrow$である。命題２によって$Arrow \cap (Process \times Process) = \phi$だから、
$f'_{AK}(p) = \{h_1(k_1), h_1(k_2), ..., h_1(k_n)\}$ $= h_1(f_{AK}(p))$を得る。
$f_{KA} \cong f'_{KA}$を示す。
任意の$p \in D$をとり、$f_{KA}(p) = \{k_1, k_2, ..., k_n\}$とする。一般性を失うことなく、ある$j, 1 \le j \le n$, に対して、$1 \le i \le j$のような i について$k_i \in K_E$であり、$j+1 \le i \le n$のようなi について$f_i = (k_i, p) \in Arrow$としてよい。命題２によって
$Arrow \cap (Process \times Process) = \phi$だから、 $f'_{KA}(p) = \{h_1(k_1), h_1(k_2), ..., h_1(k_n)\}$
$= h_1(f_{KA}(p))$を得る。（証明終）

一般のdfd に対して$G \cdot F$(dfd)はdfdと同形にならないが（命題２参照)、Fを作用させた原始静的構造としては同型である。

命題 6-6

任意のデータフローダイアグラム dfdに対して$F \cdot G \cdot F$(dfd) $\cong F$(dfd) が成り立つ。

（証明）

任意にdfdをとる。命題 6-3 より、F(dfd) は命題 6-5 の原始静的構造の条件を満たす。(証明終)

命題 6-5 と 6-6 によって、F と G がデータフローダイアグラムと原始静的構造の情報をトリビアルなやり方ではなくかなりの程度「自然に」対応させていることがわかる。さらに、このような対応がデータフローダイアグラムと原始静的構造にあることから、データフローダイアグラムは離散事象システムとしての業務システムの外観をどこまで定めるツールであるかが明確となった。

6.5 エンティティ・ライフヒストリーの意味

4 章定理 4-2 に示されたように、業務取引システムの静的結合構造だけでは、業務取引システムは一意に決定されない。動的な部分があるからである。つまり、ひとつの静的結合構造に対して動的部分も含めてただ一つの業務取引システムを確定するためには、外生トランザクション、内生トランザクション、ファイル更新関数、活動決定関数が定まらなければならなかった。ビジネスシステムの動的ふるまいを記述するひとつのツールがエンティティ・ライフ・ヒストリ（ELH）である。ELH が表現するのは各管理実体型に対するデータの記録の完全性であって、生成・参照・更新変更・削除がなされることを分析段階で保証し、業務をモデル化する上でもれをなくしようとするものである。本節では ELH が業務取引システムをどこまで決定するのかを考察する。結論的には、いくつかの特別な業務取引システムについては静的結合構造とともに ELH がそれを完全に決定するが、一般のビジネスプロセスを完全には決定しない。

動的な側面を記述する図的なツールは、情報システム方法論においていくつか提案されてきた。ペトリネットもそうだし、UML のシーケンス図やアクティビティ図や状態図もそうである。構造化分析で定められてた ELH の定義は何人かの著者によって多少異なる。いずれにせよ活動が発火したときの順序に注目する。そこで我々は ELH を定義することから始めなければならない。

図 6-2 の光速通販の業務取引システムを考える。各四角ではそれ自体がひとつの離散事象システムであり、内部状態が満期時間が来たことで離散的に変化する。ひとつの離散事象システムの満期時間には、そのシステムが起動する。そ

れらの起動した離散事象システムの名前をイベントとして記録することを考える。たとえば、「顧客」からの注文とか「受注する」というイベントが起こる。これは「顧客」や「受注する」というシステム名が発生していることである。さて、説明の簡単のために一時点ではひとつの発生しかないものとする。複数のトランザクションが発生する場合でも以下の議論は同様に成立する。業務取引システムでは、将来どのトランザクションが発生するかは、業務取引システムの現在の状態に依存し、その情報だけから決まる。状態は、データモデルで表されたファイルシステムの値と、各トランザクションの離散事象システムの疑似内部状態との組合せであるから、結局、ひとつの初期状態に対して、時間の経過とともに順々に発生したトランザクションの列がただひとつ定まる。たとえば、倉庫に非常に豊富な製品在庫を抱えている状態に対して、つぎつぎにそれらへの注文が来た場合には、この初期状態に対する生成列は、顧客、受注、在庫確認発注、在庫管理と製造指示、在庫更新の５つのトランザクションが適当にシャッフルされた列ができる。もちろん、品物によって在庫確認や在庫管理にかかる時間が違う場合には、顧客の注文する品物が異なれば、同じ初期状態からでも、異なるトランザクション列が発生する。顧客が注文する品物が異なるというのは、顧客を表現する離散事象システムの疑似内部状態の初期値が異なるということである。したがって、初期状態が定まると唯一のトランザクション列が発生する。したがって、業務取引システムに対して、このようなトランザクション列をこの初期状態に対応する生成列という。また、ある業務取引システムが与えられれば、初期状態をいろいろ変えることで種々の生成列を得られる。すべての初期状態に対応する生成列を集めた集合を、この業務取引システムの生成列が作る言語と呼ぶ。適当なシンボルの列を言語と呼ぶことはコンピュータ科学ではポピュラーな概念である。

さて、一般の業務取引システムを考える前に、特殊な構造だがわりとよく見られる業務取引システムを取り上げる。それはブラックホール業務取引システムである。

定義

業務取引システムがブラックホール型であるとは、どの管理実体型の値に対しても記録が増えていくだけで使われず、どのトランザクションも起動しないときである。

ブラックホール業務取引システムでは、データがファイルに追加されることしか起こらない。業務全体ではなく一部分だけをみると、ブラックホール型になっている。たとえば、顧客からの注文を電子的に受け取る業務活動や、アンケートを集計して記録する業務である。また、なぜ、この記録を集める業務が行なわれているのか誰も理由を分からず、社内で調べた結果、以前に社長の命令である報告書を作ったことがあり、そのときの仕事が残って定型化した業務。これも例だ。

ブラックホール業務取引システムでは、内部活動を起動することがないのだから、４章図 4-4 で示した DEVS 結合システムの状態遷移を表すフローチャートで、「実行可能な内部活動（内部 DEVS）があるか?」という判定が常に No となる。したがって、発生して記録されるトランザクションは、外生トランザクションか、初期状態と内部トランザクションの両方で決まる内生トランザクションの終了のみである。したがって、次のことが成立する。

定理 6-7

同じひとつの静的構造を持つ２つのブラックホール業務取引システム B と B' が同じ外部取引と同じ内部取引を持つとする。このとき、B と B'は生成列が等しい。つまり同じ初期ファイル値に対応するそれぞれの生成列が等しい。

つぎにフランチャイズビジネスを考える。Pickle,H.B. and Abrahamson,R.L.著「Small Business Management」(fifth edition, John Wiley & Sons, 1990) によれば、フランチャイズには２種類ある。「商品とトレードマーク提供型フランチャイズ(the product and trade name franchise) 」と「ビジネス全体フォーマット提供型フランチャイズ」である。我々が考えるのは後者であり、フランチャイザー本部とフランチャイジー店舗は全面的な関係を結び、フランチャイジーは標準化された商品やサービスを顧客に提供する。そのためにフラチャイザーによって確立されたマネジメントの方法をフランチャイジーはフランチャイザーが提供する教育によって身に付ける。さらに、フラチャイザーは財務管理や在庫管理などのほとんどの運営管理を提供する。日本でのコンビニは典型である。顧客に対してフランチャイジーの全店舗が見かけ上同一の製品とサービスを提供できるように展開している。

定義

２つのビジネストランザクションシステムが、あるひとつのフランチャイザーのフランチャイジーであるとは、同じ静的結合構造を持ち、同じ内生トランザクションと、同じファイル更新機能を持つことである。

つまり、それらは同じ静的構造を持つということから同じ伝票を使い、それを基盤としてビジネス活動が行なわれている。したがって 4 章定理 4-2 から、この２つは同じ顧客や同じ納入会社をもつならば、ただ、いつどういう活動をするかのビジネスルールを表現した活動選択関数のみが異なる。そのような例は、たとえば、光速通販の物流部門が独立オーナーの店として２つのフランチャイジーを持つとき、自社在庫製品にする品目はフランチャイザーが決めているとしても、再発注のための在庫水準の値はそれぞれの店長が決めることになっている場合である。コンビニエンスストアの弁当の在庫補充は実際の例である。

フランチャイジーの２つのビジネストランザクションシステムは生成列が等しいこととある条件によって、同じシステムとなる。

命題 6-8

２つのフランチャイズ業務システム B と B' が同じ外部取引システムを持つとする。このとき次は同値である：

（１）離散事象システムとして２つは等しい；

（２）

(2-i) それぞれの対応する離散事象システムが生成列同値である。かつ、

(2-ii) 内部トランザクションの開始が B と B' で異なることがあるようなファイルの値があるなら、システム全体としての内部トランザクションの開始が B と B' で異なることがあるようなファイルの値がある。

命題 6-8 の証明はすこしだけテクニカルである。まず離散事象システムとして B と B' の２つがひとしいと仮定すると、（２）の２つの条件はいずれも成立する。次に（２）から（１）を示せることを、対偶によって証明する。つまり（１）でなければ（２）であることを示す。このために（１）でないことと（２）であることを仮定して、矛盾が生ずることを示す。（１）でないことから、定理 4-2 により内部トランザクションの開始がして B と B' で異なる。これはそのような

ファイルの値があるということである。すると(2-ii)により、システム全体としての内部トランザクションの開始がしてBとB'で異なることがあるので、そのファイルの値を初期値とする生成列は当然異なったものになる。これは条件(2-i)に矛盾する。したがって（１）でなければ（２）でないことが言えた。

生成列同値は離散事象システムとしての同値より弱いわけである。

さて、一般の業務取引システムではどうだろうか。静的結合構造はDFDとデータディクショナリ（ファイルのデータモデル）で完全に決定されるから、もし動的なふるまいをELHで完全に定められれば、DFDとデータモデルとELHで業務取引システムを定義できる。命題6-4はフランチャイジーとしての業務取引システムが決定されることを示している。しかし、結論をいえば、以下の分析が示すようにこれは部分的にしか成立しない。

まず、ELHを定義する。以前に、業務取引システムの生成列や言語を考えた。ひとつの生成列について、各管理実体型に関係する部分だけをトランザクションの発生順序をくずすことなく「抜き出して」各管理実体ごとの生成列を定めることができる。たとえば、顧客、受注、在庫確認発注、在庫管理と製造指示、在庫更新の５つのトランザクションからなる列に対して、受注という管理実体型に関係するのは受注と在庫確認発注の２つだから、それらを順序を崩さずに抜き出して得られるひとつのトランザクション列が、受注管理実体型のひとつの生成列である。したがって、

定義

ひとつの管理実体型にとって、業務取引システムの言語からさまざまなトランザクション列が得られる。これをその管理実体型の entity life history (ELH)と呼ぶ。

定義

同一の静的結合構造に対して２つのBとB'があって、それぞれの管理実体型のELHが等しいときには、それをELH同値と呼ぶ。

次の命題はELHの定義から明らかに成立する。

命題6-9

同じひとつの静的構造を持つ２つの業務取引システムBとB' が同じ外部取引と同じ内部取引を持つとする。このとき、対応する離散事象システムが生成列同値ならば、ELH 同値である。

系

同じひとつの静的構造を持つ２つの業務取引システムBとB' が同じ外部取引と同じ内部取引を持つとする。このとき、対応する離散事象システムが ELH 同値でなければ生成列同値でない。

残念ながら、ELH 同値であっても、B と B' が等しいとは限らない。業務取引システムを完全に定義するには、静的結合構造とデータ辞書と ELH だけでは不足なのである。

実際に生成列同値が ELH 同値より強いことを光速通販で例示する。これは業務取引システム B と B' という異なるシステムおいて、それぞれの異なる生成列から同一の ELH が生ずる可能性があることを示せばよい。このようなことが生ずる原因はデータがファイルシステムの形で保持されていて、商品入荷と同時に現在庫量が更新された値が参照関係という整合性を保つメカニズムを通じて、"瞬時に"共有できるからである。B も B' も同じ静的構造を持つとする。図 6-11 のような入荷と在庫引き当ての業務を考える。

B と B' は注文に対して引き当て可能な目標在庫量が異なると仮定する。同じ入庫量に対しても B はそれをすぐに注文に引き当てるが、B' は在庫をある程度確保できてから一度に出荷準備をするという理由ですぐには引き当てずにもう少し入庫を待つものとする。これは、活動選択関数が異なるということである。

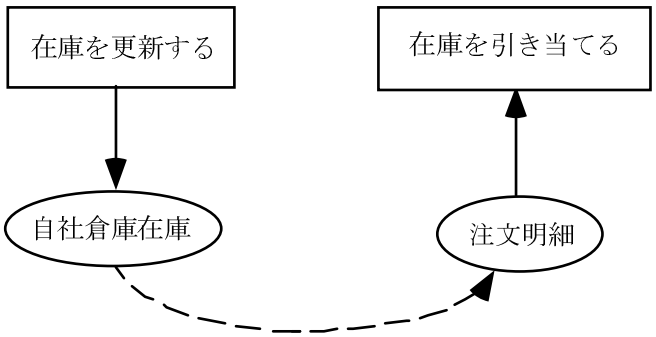


図 6-11　出荷が在庫不足によりペンディングされている

このときに B では「検収する」とすぐ後に「出荷準備する」という活動が起こるので、２回入庫があると、交互に２つの活動が起こる。しかし B' では２回入庫があってから在庫を引き当てることになる。したがって B ではこの２回の入庫に対して発生する生成列は「検収する」を a とし「出荷準備する」の終了を b とすると abab である。一方 B' のほうは aabb であるので、２つは生成列同値ではない。ところが、B においても B' においても、商品在庫の管理実体型に対する ELH は aa であるし、注文という管理実体型に対する ELH は bb であり、それぞれ B にも B' にも共通である。

Cutts も MacDermid も、ELH をトランザクションの簡単化された生成規則として図を使って定義している。本書での定義は、生成された列である。

自習課題

IDEF について、IDEF0, IDEF1X, IDEF3 を調べなさい。

6.6 UML：シーケンス図とコラボレーション図

本節以降で取りあげる図的ツールは特にビジネスプロセスの動的側面のモデル化に効果があるといわれている。業務取引システム理論を使ってツールの意味を明らかにし、モデル化の発想を豊かにするために、ツールと AID モデルとの対応関係を簡単に論じている。

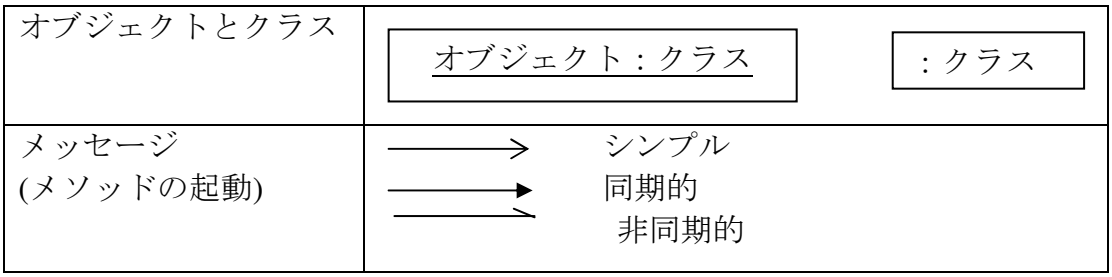


図 6-12　シーケンス図の構成要素

本節以降で取り上げる図的ツールは特にビジネスプロセスの動的側面をモデル化する目的に用いられている。業務取引システム理論を使ってこれらのツールの意味を明らかにするためにはアクティビティ・インタラクション・ダイアグラム(AID）との対応を論じる。これらは形式的な証明は存在せず厳密な議論ではないが、モデル化の際の発想を豊富にして促すために 6.6 節から 6.8 節においた。

要求分析や仕様分析は、実現できない機能を定義する可能性や、冗長な機能が入っている場合などがあって、いきなり正確なものができるわけではない。むしろ、たたき台を作ってそれを練り上げていく。練り上げのプロセスは、処理ロジックの推敲や、プロトタイプ作成と並行して検討することで行われる。

UML (unified modelling language)はオブジェクト指向のツール群からなる。したがって、なんらかの方法論によって開発プロジェクトの進め方を規定し、ツール群の中の適切なツールによって適宜ビジネスプロセスと情報システムがモデル化される。

UML に含まれるツールは、ユースケース、ユースケース記述、クラス図、シーケンス図、コラボレーション図、ステートチャート図（状態図）、アクティビティ図、コンポーネント図、配置図がある。

シーケンス図とコラボレーション図は、同じ内容を異なるあらわし方で表現している。本節では、シーケンス図の意味を、AID を用いて考える。

情報システムがアクター（活動実行者）に提供する機能がユースケースによって表現される。シーケンス図は、横軸にオブジェクトを四角で示す。ユースケースを実現するのに必要なクラスとメッセージの流れを順序立てて示す。つまり、図の上から下に向かって時間軸が設想定されている。シーケンス図のクラスごとに描かれた縦軸はライフラインと呼ばれる。縦軸にそって、うえのメソッドや活動が順に実行される。

図 6-12 はシーケンス図の構成要素であり、図 6-13 はシーケンス図によるビジネスプロセスのモデル例である。参考文献[吉野君卒業論文]の中に示された、カスタマイズドウエアの見積書をとるための Web サービスと Web サイトと連携のようすを示している。「apparel」のところをなんらかの離散組み立て型製造業に見立てれば、製品仕様のコンフィグレーションをユーザが指定し、見積書を得るまでのプロセスを書いているとみなせる。メーカーの Web サービスは、メーカーの Web サーバにあらかじめ他のクライアントから起動できるようなプログラムを用意してお

いて RPC（remote procedure call--他社の遠隔コンピュータからのプログラム実行）と同じように起動するものである。代表的な例は、製品データベースからの製品 ID をパラメータとして受け取って、その製品情報を検索して返すものである。検索する際にさらに他社の Web サーバを通じて Web サービスを使うというような必要に応じて、Web サービスを何段か接続して使うようなことも可能である。図では:
(1) 顧客がメーカーを指定し、そこで製造可能な製品をブラウザ上に表示させ、
(2) そこから、製品を指定して、可能なコンフィギュレーションを表示させる。
(3) さらに、カスタマイズ項目として選択項目を指定し、希望数量とともに送ると、
(4) 見積書が得られる。

このシーケンス図は、データベースや GUI や Web サービス間のメッセージによる連携の手順を示している。手順として事項されるのは、人間による活動であったり、Web サービスのプロセスの実行である。

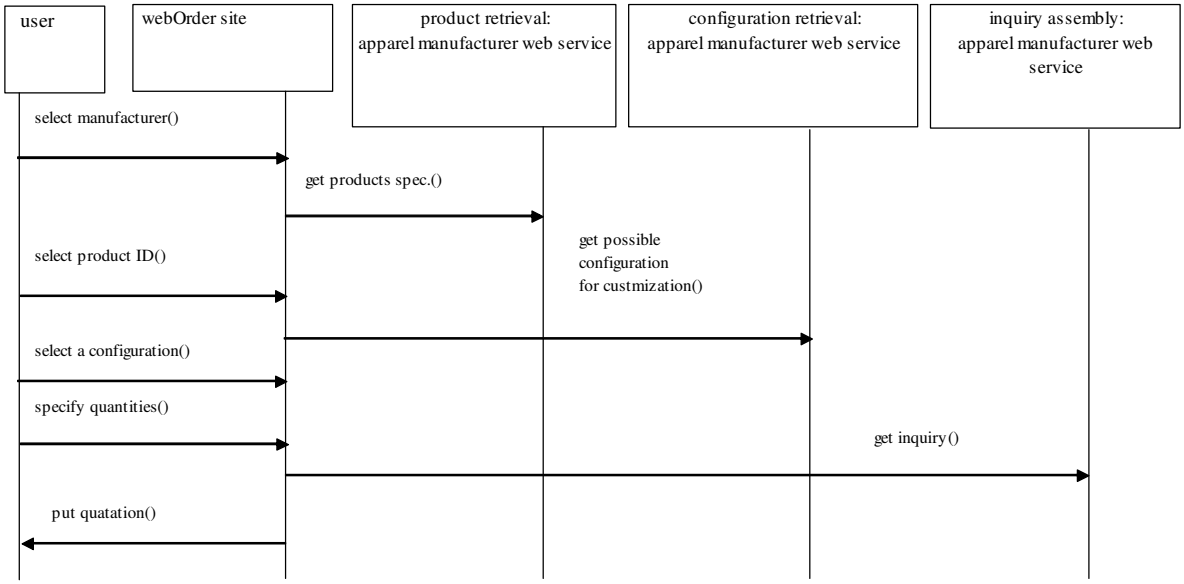


図 6-13　Web サービスによる連携のシーケンス図モデル

Web サービスと GUI と人間によって強調しながら実行されるビジネスプロセスも、業務取引システムであるわけだから、AID による構造を持つはずである。実際この図が表現しているビジネスプロセスは図 6-14 のようになる。つぎのような対応規則によってシーケンス図（あるいはコラボレーション図）から AID を導くことができる。

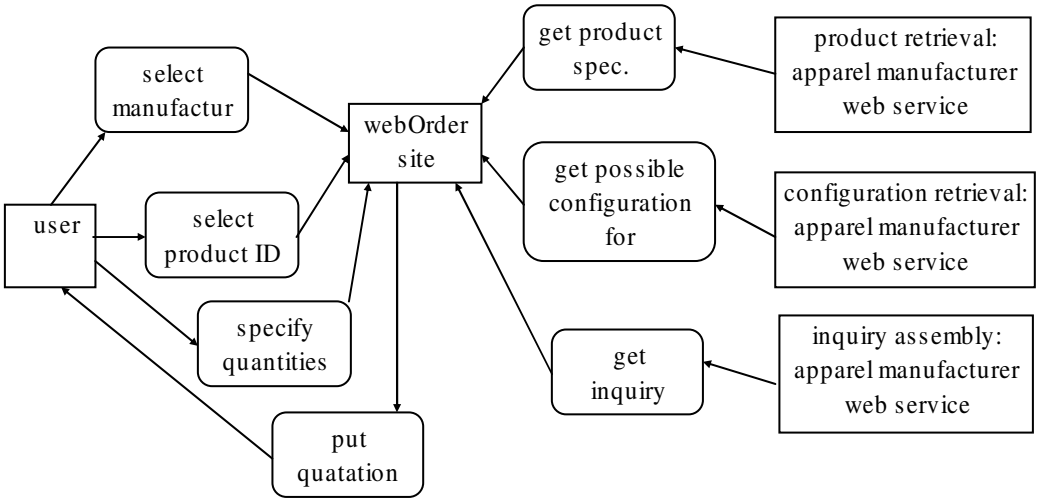


図 6-14　Web サービスのシーケンス図から得られる AID 構造

シーケンス図・コラボレーション図から AID を得るためのルール

シーケンス図	アクティビティ・インタラクション・ダイアグラム（AID）
オブジェクト（クラス）	管理実体型か、外部活動
メソッド	活動
メッセージの矢印	AID の矢印

データストアに対する検索のための get メソッドは、データフローの矢印の向きを逆にする。
データフローには名前をつかないので、人工的な形式的名称をつける。

シーケンス図作成への提言・ガイドライン

シーケンス図およびコラボレーション図は、ユースケースに対応している。メソッドに対して活動を対応させるが、活動の内容を見ることによって、入力としているデータと出力データを知ることができる。シーケンス図がスケルトン的な構造概念図として使われるには、対応する AID 構造もスケルトン的であって、特にデータフローが明示されない。逆に、データフローを明示するように、AID を作成した際に調べることによって、もとのシーケンス図の正当性をチェックできる、また、AID が状態繊維できるかどうかを確認する中で、必要なデータが活動の入力になっているかを調べて、必要なデータは新しい入力項目として AID に取り込む。このとき、クラス図の正当性チェックも同時に行う。

シーケンス図では、オブジェクトＡからＢへメッセージを送ること矢印を書くことで、ＡがＢのメソッドを起動することを表す。メッセージの矢印には３種類定義されている（シンプル、同期、非同期)。AID ではすべての矢印が「シンプル」に対応し、同期性には焦点を当てていない。

シーケンス図では、クラスとその具体的オブジェクトを認識して、箱で表現するが、AID ではクラス（管理実体型）として認識している。クラスは概念ファイルであり、実装においては DBMS のテーブルとなる。オブジェクトの実装はテーブル内のデータになる。

6.7 ARIS と EPC

ＥＲＰの最もポピュラーな SAP R/3 では、ビジネスプロセスを記述するのに、EPC(event-driven process chain)ダイアグラムを用いる（Keller and Teufel, 1998)。

EPC の構成要素は図 6-15 の通りである。

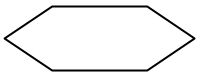
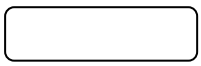
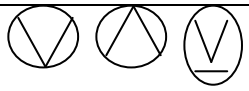
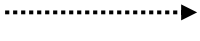
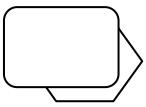
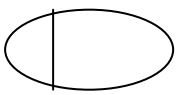
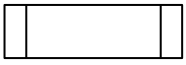
要素名	シンボル	説明	要素の例
イベント		処理を必要とするオブジェクトに関連する時刻	注文の到着； 生産オーダのリリース
機能 (活動)		入力から出力への変換	注文の見当; 組立て
論理結合・分岐記号		OR; AND; XOR	
コントロールフロー		処理の順序	
外部のプロセス		プロセスへの外部入力; プロセスから外部へ出力	生産管理システム；顧客
組織ユニット		部署; 機械; 担当者	営業部; 組立ライン
情報オブジェクト		データ；表	受注; 従業員一覧
情報フロー		生成; 読み出し; 更新; 削除	

図 6-15.　EPC の構成要素

EPC から AID を得るためのルール

イベント駆動プロセス図 (EPC)	アクティビティ・インタラクション・ダイアグラム (AID)
イベント	管理実体型
情報オブジェクト	管理実体型
機能 (活動)	活動
コントロールフロー、情報フロー	結合の矢印
結合要素 (AND, OR, XOR)	(暗黙的で明示しない)

図 6-16(a)は受注生産における個別受注のコスト計算のプロセスの例である (Scheer, 1994)。イベントは、直接的には文字通りある時刻のことであって、本来は何かの作業が終わったとか、注文が到着したことを表している。EPC では、イベントの時刻に発生したたとえば到着した注文自体もイベントとして表現するので、処理が必要なオブジェクトもイベントに含まれる。

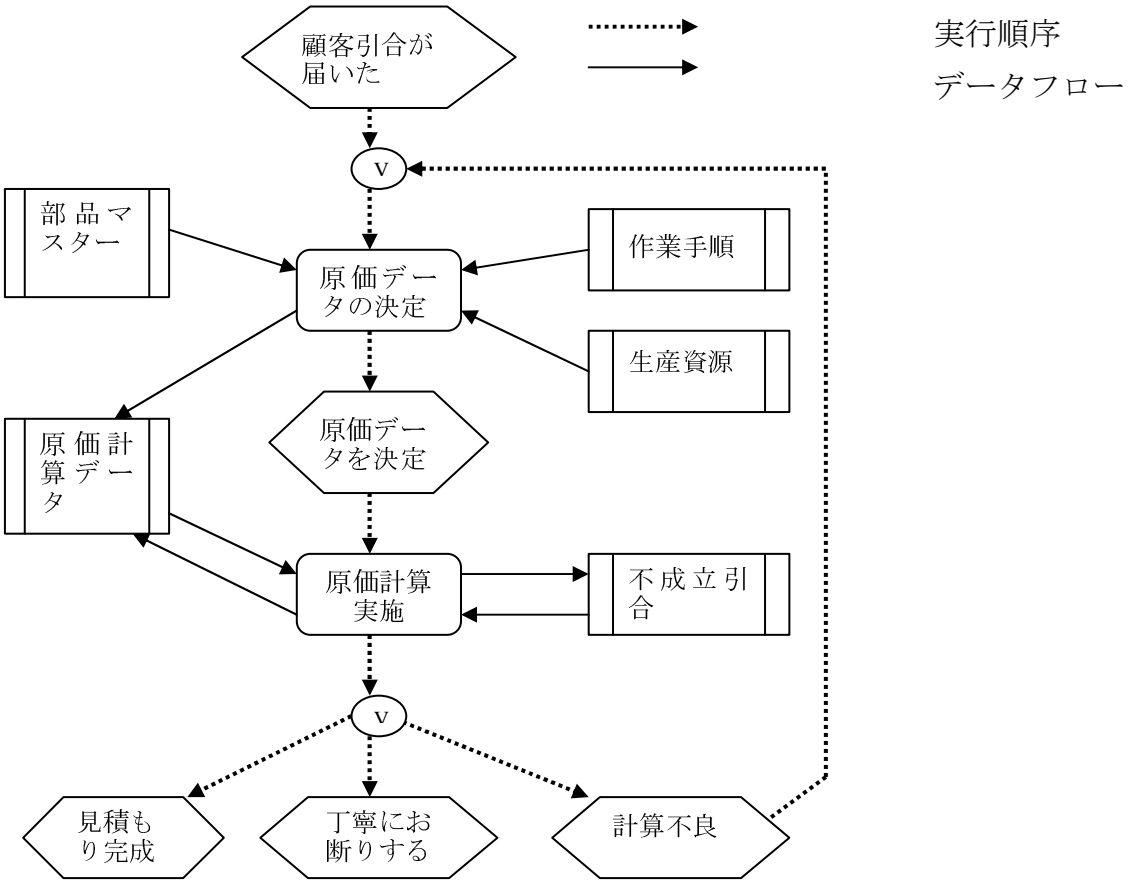


図 6-16(a) 個別受注の原価計算の EPC モデル

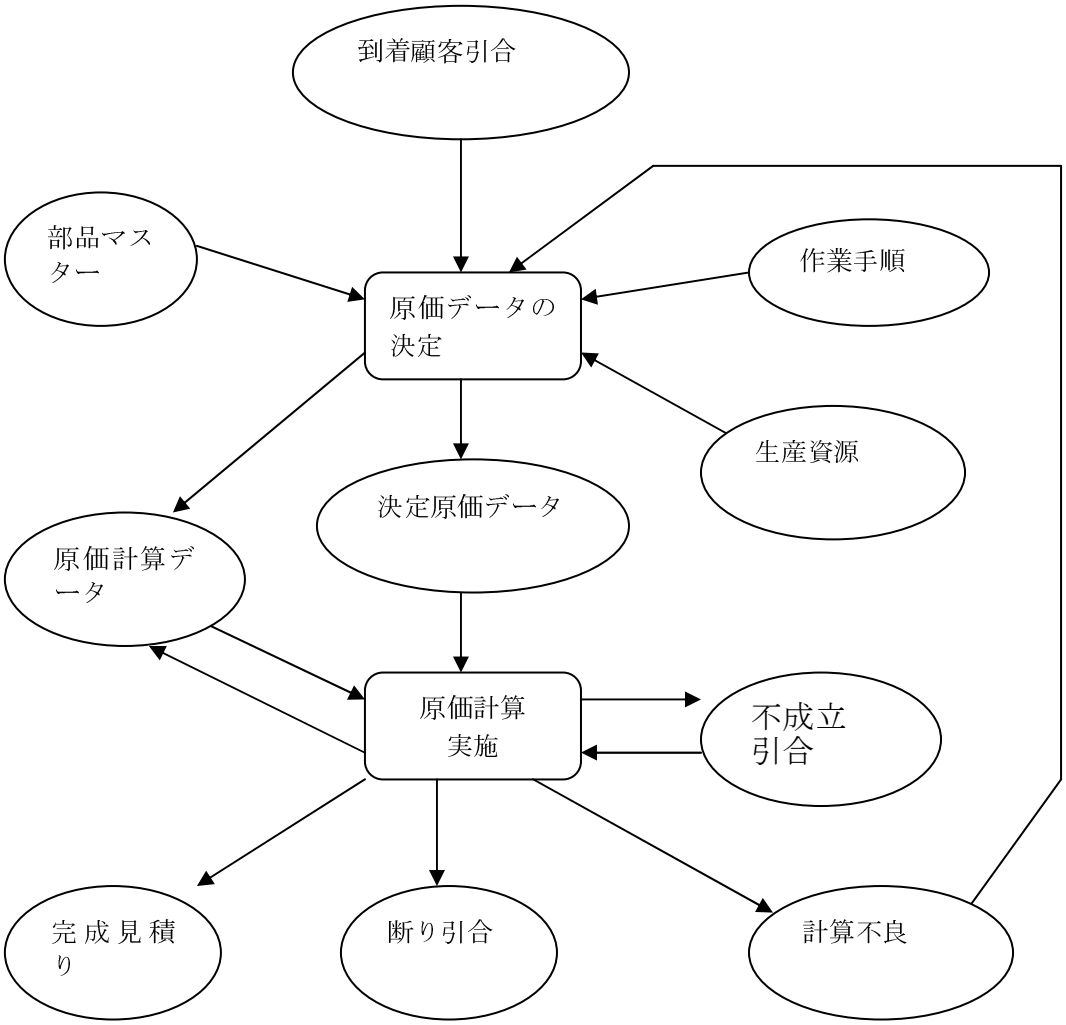


図 6-16(b) 個別受注の原価計算 EPC から得られる AID モデル

EPC 自体は、業務取引システムの構造を明示的に意識していないので、EPC の整合性チェックするときに、その EPS から得られた AID について、それが状態遷移の基礎構造を決めている点からの検討が可能である。

活動を開始するための必要な情報がすべてモデル化されているか。

まとめ

ビジネスプロセスモデルの方法論として３つを取り上げて、それらの意味を業務取引システムとの関係から論じた。ＵＭＬのシーケンス図とコラボレーシ

ョン図のメソッドの呼び出しを、ＡＩＤの活動である。同時に、オブジェクトはデータストアとしての働きを持ち、活動開始のデータは暗黙的にシーケンス図で表現されている。また、ＥＰＣの機能はＡＩＤの活動であること、ＥＰＣのイベントや外部プロセスが持つデータが暗黙的にモデル化されている。これらのことから、次の 2 つのことが言える。

(1) 作成するモデルのチェック機能 : プロセスモデリングにおいて、活動が開始できるかどうかを決定できるだけのデータを入力に用意する必要がある。充分に入力データがあるか

(2) この考察によって、業務取引システムの静的結合構造であるＡＩＤを標準構造として利用することで、異なるモデル間の自動変換を実現し、モデル構築の支援やプロセスモデル再利用の道を開く。

6.8　プロジェクト・マネジメントのプロセスモデル

プロジェクト管理のために有効に使用されている図的ツールとして WBS（work breakdown structure: 作業詳細構造）がある。一方、生産管理や生産計画に使われている部品表（BOM- bills of materials）は、作業手順とあわせて表現することによってプロセスとプロセス成果物とを表すことができる。

WBS とはプロジェクトを完成させるために詳細化した一連の活動である[9]。WBS の個々の活動は次の表のように書き上げられて、図 6-16 のような形で構造化されることが多い。

活動の間にさらに詳細化の階層があって、図 6-17 のような部品表と作業手順として捉える。プロジェクトを部品表としてとらえたモデルを WBS 部品表と呼ぶこととする。

対応する AID は省略する。

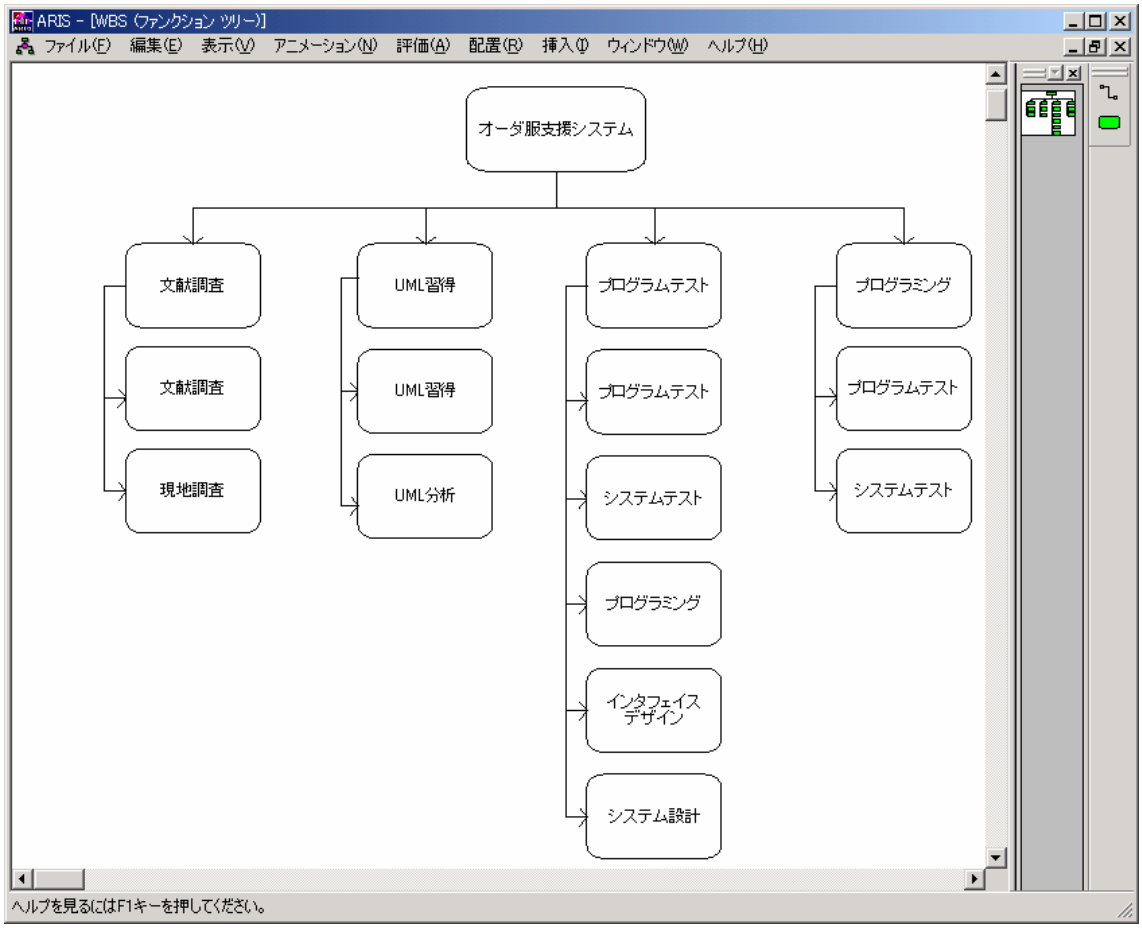


図 6-16　WBS 構造の図示（ARIS Toolset™ による）

開発フェーズ	作業コード	作業名	作業時間（日）
調査	C1	文献調査	30
	CE1	現地調査	25
システム分析	E1	UML習得	10
	E2	UML分析	10
製作	PE1	資源準備	10
	P1	技術習得	20
	P2	プログラミング	30
	D1	UIデザイン	10
	PDE1	システム設計	15
テスト	PE2	プログラムテスト	1
	PE3	システムテスト	1

表　WBS の構成要素となる活動と作業リソースと所要時間

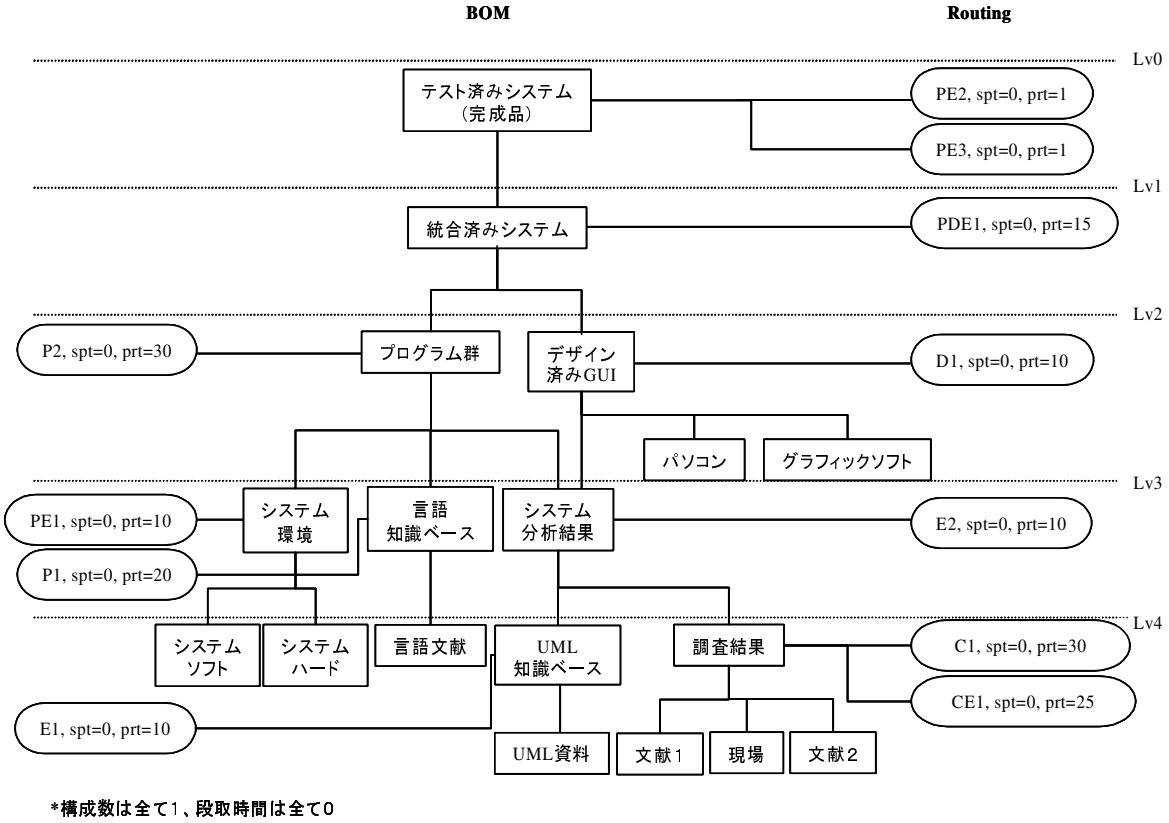


図 6-17　WBS 部品表と作業手順（佐藤他,2003）

WBS 部品表を用いることで、生産管理の場合と同じようにして、ガントチャートを用いて WBS の全活動をスケジュールできるという特徴を持つ。AID では活動開始条件を活動への入力とする。WBS においても活動担当のリソースもモデルの中に表現すれば、それを用いて実行可能性の高いスケジュール作成と動作シミュレーションをすることが可能となる。

第 6 章章末問題 1

ビジネスプロセスの現状分析と改善設計案作成の練習

モデル, M.E 著「第 1 線技術者のための実践システム分析」の第 20 章事例研究のケースの記述や、カッツ著「情報システムの分析と設計」の第 3 部のケース記述を読んで、活動や組織やデータを拾いだし、アクティビティ・インタラクション・ダイアグラムとしてモデル化する。記述の詳細さは、最初の分析を

終えてビジネスの状況を上司に報告するような荒さであり、ひとつのケースが10 ページ程度で記述されている。

ここでのビジネスプロセスのモデル化・プロセス設計とは、端的には AID とデータモデルを作成することである。AID ではなくとも AID との対応関係がある UML や ARIS の概念や WBS（作業手順）によってプロセスを記述してもよい。その際に、各活動を開始するために必要なデータがそろっているかどうかに注目し、不足していたら自分で追加する。また、たとえば生産計画データを使う活動があれば、そのデータを出力している活動が必要だが、それもなければ追加する。このように、AID の構造を持つようにモデル化して行くことで、ビジネスプロセスモデルの全体整合性のチェックができる。

なお、市販の DBMS と GUI 環境を使えば、AID＋データモデル＋簡単な GUI 設計によって、プロトタイプを試作できる。実際に大学の演習科目として行っている。

章末問題 2

情報システム分析では、新しいビジネスプロセスを設計することを要求されていることが多い。新たなビジネスのためにビジネスプロセスの新たな機能の方向性を定めるのは多面的な議論が必要であって簡単でない。そこで、ソフトシステム方法論（SSM）とか、イノベーション・アーキテクチャのような方法論を使うことになる。方法論には工学的なモデルではなく、組織的な手順とその基盤概念が説明される。方法論は工学的原理とともに用いられることが多い。SSM の文献（チェックランド,1985;ウイルソン,1992;ローゼンヘッド，1996;McDermid,1990 など）やイノベーション・アーキテクチャの文献によって Sauber and Tschirky(2006)調べてまとめなさい。

第７章　ビジネスプロセスの動特性設計への応用

ビジネスプロセスの動特性設計として、ビジネスプロセスのリードタイムを計算する方法を考える。たとえば、新たに受け付けた注文が、それ以前に受けた注文が処理されている間は待ち続け、すべての自分の前の注文処理が終わった後に処理され、顧客へ商品やサービスが送られて受け取られるまでの時間が顧客にとってのリードタイムである。ビジネスの実行形態はネットワークや IT の発展とともに止むことなく変化し、ビジネスプロセスのモデリング記法や概念がたくさん提唱され続けているが、ビジネスプロセスの構成と関連させてリードタイムを工学的に計算することや設計することはうまくいっていない。現実ビジネスではリードタイム短縮へのソリューション指向が強すぎる余り、原材料から顧客配送済み状態の製品に至る長いロジスティクスプロセスのどこかに在庫負担を押し付けるかによって実現されている感がある。

工学的設計のためには基本的に何を考えるべきか、どのような特性に注目することで設計可能になるかを本章で考察する。ポイントは、ビジネスプロセスが長期間にわたってある程度安定的に運転されている状態に注目することである。定常的な長時間にわたる平均を考えることで、ビジネスプロセスに対して普遍的に成立する事実があり、それがビジネスプロセスの過渡的適応力に対しても示唆を与える。CRM や需要予測などの計画情報システムの基本的役割はビジネスプロセスの定常的運転の実現であると考えられる。

7.1 リトルの定理

ビジネスプロセスを流れて行くオブジェクトに注目した時、下の図のように、時間的に平均すると、ほぼ一定の速度と個数で、オブジェクトが流れている状況を考える。

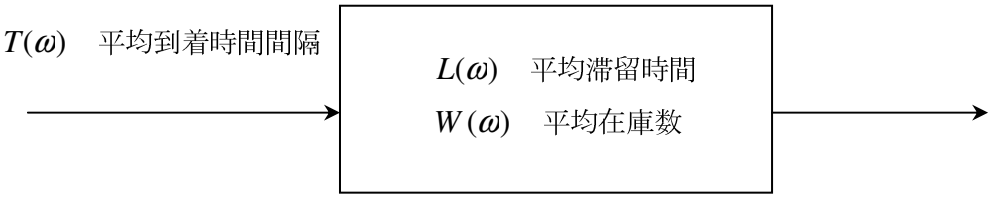


図 7-1　全体プロセスを１つの業務活動として近似する

リトルの定理[Little, J.D.C., "A proof for the queuing formula: $L = \lambda W$," *Operations Research*, 9, 383-387, 1961)]

Ω を適当な確率空間とする。強定常とする（確率が時間によらない）。次の３つの確率変数$W(\omega), T(\omega), L(\omega)$を定義する。独立な任意の分布を持つ。

$W(\omega) = \lim_{t\to\infty} \frac{1}{t} \int_0^t n_s(\omega)\, ds$　システム内のオブジェクトの平均個数 [個]

ただし $n_s(\omega)$ は時刻 s においてシステム内に存在するオブジェクト在庫個数（work-in-process)

$T(\omega) = \lim_{m\to\infty} \frac{1}{m} \sum_{j=1}^{m} \tau_j(\omega)$　オブジェクトの平均の到着時間間隔 [単位時間／個]

ただし、$\tau_j(\omega)$ は j 番目と j+1 番目のオブジェクトの到着時間間隔

$L(\omega) = \lim_{m\to\infty} \frac{1}{m} \sum_{j=1}^{m} \lambda_j(\omega)$

平均の滞留時間[sec, min, hour, day, week, month, 等の時間]

ただし、$\lambda_j(\omega)$ は j 番目に到着したオブジェクトがシステム内に留まる時間（lead time)

このとき、$L(\omega) = T(\omega)W(\omega)$が成立する。（確率１で等号が成立する）

定理の意味

図 7-1 に描かれているように入力が一定の割合で入ってくるので出力も同じ割合で出て行っている。

$L(\omega) = T(\omega)W(\omega)$を言葉で表すと：

時間平均値について

（システムのリードタイム）＝（到着時間間隔）＊（在庫オブジェクト数）

確率変数表現が見なれないしリトルの定理自体の証明は簡単ではないのだが、しかし、以下の例１ー３で見ていく通り、意味は直感的にごく当たり前のことである。また、本章では確率は使わないで考えていく。確率変数ではなく、十分長い期間での平均値として変数L, T, Wを考える。

$L = T \cdot W = W \cdot T$を変型すると$L = W / \frac{1}{T}$ であり、

スループット$TH = \frac{1}{T}$ [個／単位時間]として定めると、$L = W / TH$ とは次のよう

な変形と意味を持つ。

[1]（リードタイム）＝（在庫オブジェクト数）／（単位時間あたり到着数）
＝（平均在庫数）／（スループット）
[2]（スループット）＝（平均在庫数）／（リードタイム）
[3]（平均在庫数）＝（スループット）・（リードタイム）

到着時間間隔や作業時間が確率的な変動を持たず常に一定の定数であるときに、決定論的であるという。決定論的な場合には$L = T \cdot W$は自明に成立する。しかも、この世の出来事である限り、モノやオブジェクトが蓄積され流れていく状況では、この関係式から逃れることは起こり得ない。あまねく成立する原理的関係であって重大事項である。

ビジネスでは標準作業時間を考えて動的特性を設計する必要がある。まず、設計通りに動くとすると新規設計したビジネスプロセスがどの程度の性能を持つのかを考える手段が必要なのである。ビジネスプロセスの構造的な妥当性を検討するために必要なのである。

リトルの定理は決定論的なシステムの定常状態（平衡状態）に対して適用すると、リードタイムについての基本的な見通しを与え、ビジネスプロセスの設計に役立つ。「在庫する」のは注目するビジネスごとに変わる。部品や製品であることもあれば、また、顧客からの注文であったりサプライヤへの支払いであったりする。下記では「決定論的リトルの定理」としているが、決定論的状況は確率的状況の特例（確率が１）であるので、別に新しい事実ではなく安心して使える。

TH：1 時間あたりの生産速度（スループット）= 注文投入速度 TH [個／時間]

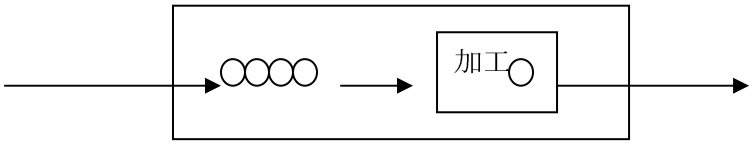
WIP：平均 WIP（work-in-process）在庫の個数　WIP [個]
各々の作業の前と作業中を合計した部品在庫（の時間平均値）。

LT：平均リードタイムを LT [時間]とすると、

（決定論的）リトルの定理　　LT = WIP／TH

例１　決定論的でシンプルな場合（１）

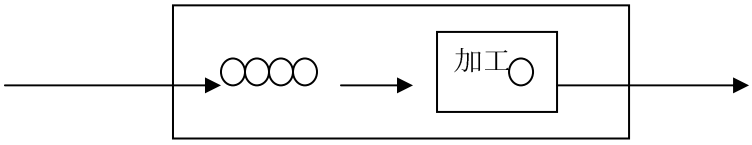
到着間隔が 7 分ごとで WIP が（平均） 5 個あるとする。出荷間隔は 7[分／個]である。定常状態（平衡状態）では、これらの平均値が一定であるとする。到着間隔と出荷間隔は一致している。もし一致しなければ平均在庫数は一定ではなく増加し続けたり、減少傾向を見せたりするはずだからである。



リードタイムを計算したい。LT[分] = 到着間隔[分／個]*WIP[個] = 35[分] である。決定論的だからこの関係が成立することは自明である。つまり、新たに到着したオブジェクトは WIP が作る待ち行列の最後に並び、自分の順番がくるとサービスを受けて加工されたりして完成する。そのときに、完成までの待ち行列全体を考えると平均して 5 個あるのである。新たに到着したオブジェクトは、その平均 5 個の中に含まれているので、リトルの定理で述べられる関係式が成立する。

例２　決定論的でシンプルな場合（２）

平衡状態において次のようだとする。WIP＝平均 5 個　加工時間＝6 分／個

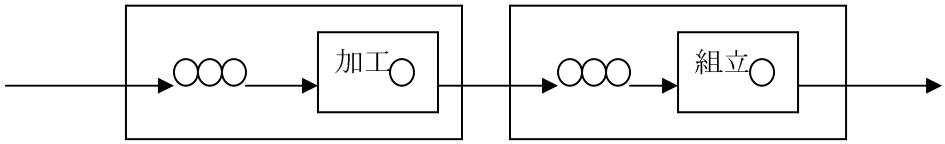


スループットを計算したい。つまり、１個当たりの加工時間に相当する時間が 6[分／個]であって、かつ平衡状態にあるのだから、出力も入力も同じ速度で動いているはずである。よって、TH =1/6[個／分]である。
なお、リードタイムは LT[分] = 到着間隔[分／個]*WIP[個] = 30[分] である。

例３　決定論的でシンプルな場合（３）

全体プロセスで見ても、ひとつの個別で見てもそれぞれについてリトルの定理が成立する。平衡的な状態になっているためである。

WIP 4 個、加工時間＝6 分／個；WIP 4 個、組立時間＝6 分／個



スループットは組立だけに注目すると、出力のスピードから 1/TH = 6 分／個。

リードタイム計算１　リトルの定理の適用１

$LT_{組立}$ = 4/TH = 4*6 = 24 分

$LT_{全体}$＝2*24 ＝48 分

リードタイム計算２

全体 WIP = 8 個

$LT_{全体}$ = WIP/TH ＝8*6 ＝48 分

注意 : 活動の時間が異なる時、全体のリードタイムは注意して扱う必要がある。特に平衡状態（定常状態）にならないとリトルの定理を使えない（成立しない）ことに留意する。

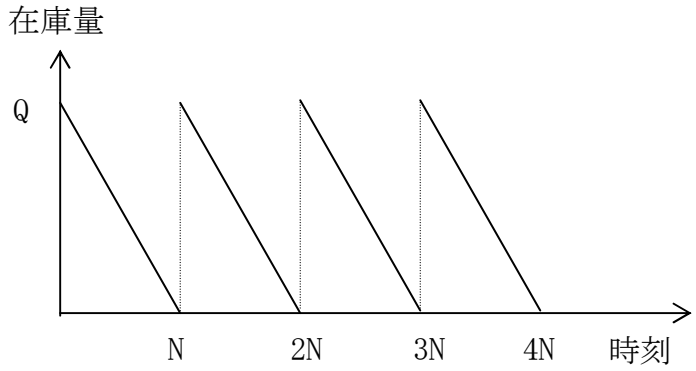
例４　WIP の定義と計算

WIP は時間平均の在庫量である。リトルの定理に記してある通りだが、一定のスピードで出荷がなされて在庫が減っていく場合について計算方法を確認しておく。実際は品物は離散的に減っていくのであるが、連続量によって離散的変化を近似すると理想的には次の図のようなノコギリ状の図形とみなせる。このとき、まず時間長さ N の期間での平均を考えると、

$$WIP = \frac{1}{N}\int_0^N (-\frac{Q}{N}t + Q)dt = \frac{Q}{N}[-\frac{t^2}{2N} + t]_0^N = \frac{Q}{2}$$

である。積分を実行しなくてもノコギリ図形の中のひとつの三角形をみて、その面積（期間 N の間の総在庫量）$\frac{QN}{2}$ を期間 N で割ると WIP になる。周期的な

在庫量の規則的変動なのでこの値が全期間に渡る平均値となる。



7.2 ビジネスプロセスのリードタイム計算：直列プロセス

シンプルな直列の作業手順をもつプロセス

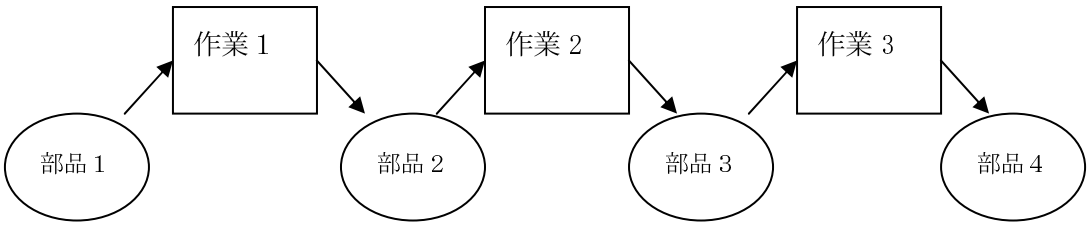


図 7-2 ３つの作業の直列プロセス

定常に運転されているものとする。つまり、長い期間に渡って平均をとると、一定の速さでジョブ（作業要求）が到着し、それと同じ速さで最終作業から完成品が出ていっているとする。各作業の作業者は一人づつとする。平均滞留在庫(WIP)、スループット(TH)、リードタイム（LT）を求める。

（ケース１）

プロセスの設定

作業１、２、３の標準作業時間　５分／個；

平均 WIP（work-in-process）在庫の個数　WIP（個）: 各活動部分に 0.5 個;
システム全体の平均 WIP 総数は 0.5*3 = 1.5 個。

考え方：

状態遷移表を用いて分析する。試しに次の初期状態を考える。もし定常状態になった場合は、Cust の満期時間（1 度発生してから次を発生するまでの時間間隔）が、1/TH [時間／個]になる。状態遷移表の様子を見ながら、発見的に決めていくことにする。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(1)	0	?	0	1(5)	0	1(5)	0	1(5)	0
(2)	5	?	0	Λ	1	Λ	1	Λ	1
(3)	5	?	0	Λ	0	1(5)	0	1(5)	1

上の（2）行目の p1Q は 0 である必要がある。さもないと、act1 の WIP 個数が 1 個となっていまい、0.5 個という状況ではなくなる。

さて、同じ理由で、time =10 においては、p1Q に 1 個入り、活動 1 が開始される必要がある。

よって time=10 では p1Q=1 である必要がある。：

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(3)	5	?	0	Λ	0	1(5)	0	1(5)	1
(4)	10	?	1	Λ	0	Λ	1	Λ	2

上の（４）行目で 1 になるために、Cust は 10 時間に 1 つ到着する必要がある。よって（１）行目の Cust は 1(10)とする。これを、まとめて時刻ゼロから逐次状態遷移をみる。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(1)	0	1(10)	0	1(5)	0	1(5)	0	1(5)	0
(2)	5	1(5)	0	Λ	1	Λ	1	Λ	1
(3)	5	1(5)	0	Λ	0	1(5)	0	1(5)	1
(4)	10	1(10)	1	Λ	0	Λ	1	Λ	2
(5)	10								.
(6)	15	1(5)	0	Λ	1	Λ	0	Λ	3
(7)	15	1(5)	0	Λ	0	1(5)	0	Λ	3
(8)	20	1(10)	1	Λ	0	Λ	1	Λ	3
(9)	20								.

問 7-1

上の状態遷移表の（５）行と（９）行を完成せよ。

偶然にも状態遷移表の（5）行目と（9）行目が完成品の個数を除いて同じである。ということは、(9）行目以降は同じ様子がくり返される。つまり、定常になったということである。そこで、(5）行目から（8）行目までの間の平均値をとって、リトルの定理を適用できる。

1) 部品 1 (p1Q)と活動 1 （act1）を合わせた平均 work-in-process 在庫である w1 の計算。

act1 に部品がある時間＝５　（時刻 10, 11, 12, 13, 14）

p1Q に部品がある時間＝０　（時刻 20 で一瞬入るが、止まることなく活動 1 に持っていかれる）

よって平均すると、(5 分間にわたり 1 個ずつ）／10 分間

w1＝５個分／10 分間＝5/10 ＝0.5 個　（幸運にも問題の条件と一致した）

2) スループット throughput

平均的に、Cust から入ってくるスピードで出ていく。定常状態だから。

よってスループットを TH[個／分]とすると、TH は到着間隔時間の逆数だから

到着間隔時間　10 分／個　＝1/TH

□　TH = 1/10 [個／分] = 60/10 [個／時間] = 6 [個／時間]

各活動が直列だから、活動 1 の TH がシステム全体の TH となる。

3) 全体のリードタイムを求める

3-1) p2Q, act2 の平均在庫 L2 が 05 個であることを確認する。

act2: 5　（時刻 15, 16, 17, 18, 19）

p2Q: 0

よって　w2＝(5+0)/10 = 0.5 個

3-2) p3Q, act3 の平均在庫

act3: 5　（時刻 10, 11, 12, 13, 14）

p3Q: 0

よって　w3＝(5+0)/10 = 0.5 個

3-3) 全体では　WIP = w1+w2+w3 = 1.5 [個]

よって、リトルの定理から、

平均リードタイム LT[時間] ＝WIP／TH＝1.5/6 = 1/4 [時間] =15 分

なお、リードタイムは状態遷移表をつかって追跡することでも求められる。また、直感的には、在庫が 0.5 個ということから、生産オーダが少なくてガラガラの状況なので、３つの加工プロセスを直列に経由する 15 分で終了できそうなことが想像できる。

リトルの定理の強烈な点は、上の例のようねひとつの定常な状況を調べると、おなじ定常条件を満たす「すべての」場合についても、たとえばリードタイムが同じ値になることを保証していることである。

（**ケース** 2）

作業１、２、３の標準作業時間　５分／個；

平均 WIP（work-in-process）在庫の個数が、それぞれ 4 個ずつある。

1 時間あたりの完成速度（スループット）は？

考え方：

WIP がそれぞれ４個なので、次の（１）行目の初期状態について考えてみる。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(1)	0	?	3	1(5)	3	1(5)	3	1(5)	0
(2)	5	?	(4)	Λ	1	Λ	1	Λ	1

上の（2）行目の p1Q は、w1＝4 を満たすために、4 である必要がある。

ということは、Cust が５分に１個を出力する必要がある。つまり、

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(1)	0	(1(5))	3	1(5)	3	1(5)	3	1(5)	0
(2)	5	1(5)	4	Λ	1	Λ	1	Λ	1

この後の状態遷移を調べ、条件を満たすかどうかを確認すればよい。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(1)	0	1(5)	3	1(5)	3	1(5)	3	1(5)	0
(2)	5	1(5)	4	Λ	4	Λ	4	Λ	1
(3)	5	1(5)	3	1(5)	3	1(5)	3	1(5)	1
(4)	10	1(5)	4	Λ	4	Λ	4	Λ	2

偶然にも（2）行目と（4）行目が完成品の個数を除いて同じであるので、定常になる。そこで、時刻５から時刻９までの５時点の間の平均値を取ることで、リトルの定理を適用できる。

1) 部品１(p1Q)と活動１（act1）をの平均 work-in-process 在庫 w1 の計算。

act1 に部品が 1 個ある時間＝５　（時刻 5, 6, 7, 8, 9）

p1Q に部品が 3 個ある時間＝5　（時刻 5, 6, 7, 8, 9）

よって平均すると、(5 分間にわたり 4 個ずつ）／5 分間

w1＝(4*5 個分)／5 分間＝20/5 ＝4 個　（問題条件と一致）

2) 他も同様。 W2 = 4, w3 = 4 [個]

3) したがって、上の状態遷移図は、問題の条件を満たすようなひとつの場合である。よって, TH = 1/5 [個／分]＝12 [個／時間]

4) 平均リードタイム

LT = W/TH = (4+4+4)/12＝ 1 [時間]

観察１：上のケース２は、直感的にもほぼ明らかである。常に中間在庫を多く持っているるので、各機械はフル稼働している。したがって、平均スループットは５分間に１個の割合であるし、リードタイムは、新たに投入される材料が完成するのは（その新規投入も含めて）合計 12 個の平均在庫の加工をすべて終わるときである。つまり

（５分／個）*１２個＝６０分

状態遷移表とリトルの定理の威力は、計算を厳密にできることと、ひとつの初期値からの定常状態を調べて、同じ定常状態になるようなすべての初期状態についても成り立つ性質を求められることである。

観察２：ケース１とケース２の比較。

（ケース 3）

作業１、２、３の標準作業時間　５分／個；

平均 WIP（work-in-process）在庫の個数が、それぞれ 60 個ずつある。

ケース２の考察を踏まえると、次の状態遷移表を得る。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(1)	0	1(5)	59	1(5)	59	1(5)	59	1(5)	0
(2)	5								1
(3)	5	1(5)	59	1(5)	59	1(5)	59	1(5)	1
(4)	10								2

問 7-2

上の状態遷移表の（２）行と（４）行を完成せよ。

よって、w1 = w2 = w3 = 60 個。TH = 1/5 [個／分]＝12 [個／時間]

平均リードタイム　　LT = W/TH = (60+60+60)/12＝15 [時間]

（ケース 4)

作業１、３の標準作業時間　３分／個；作業２の標準作業時間　５分／個

問題：いろいろな初期値から始めたとき、定常になった場合の平均 WIP 数と、そのときの平均スループットと平均リードタイムはいくつになりうるか。

４つの場合を調べて、結論を得る。

・基本的事実。

1) もし定常になるならスループットは Cust の満期時間である。最も遅い作業２が 5 分／個なので、もしそれより速く入ると、必ず長期的には作業２の前に部品在庫が拡大していく。よって、スループットの速さの最大値は 5 分／個である。

（ケース 4-1)

Cust の満期時間＝５分／個、初期の各作業の WIP を１とする。

time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
0	1(5)	0	1(3)	0	1(5)	0	1(3)	0
3	1(2)	0	----	1	1(2)	0	1(0)	0
3	1(2)	0	----	1	1(2)	0	----	1
5	1(0)	0	----	1	----	1	----	1
5	1(5)	1	----	1	----	1	----	1
5	1(5)	0	1(3)	1	----	1	----	1
5	1(5)	0	1(3)	0	1(5)	1	----	1
5	1(5)	0	1(3)	0	1(5)	0	1(3)	1
8	1(2)	0	----	1	1(2)	0	1(0)	1

時刻 0 から 4 までの間で平均を計算すればよい。

w1[個]=3/5 （□時刻 0, 1, 2 に在庫あり）。

w2: act2 は 1 [個]、p2Q は時刻 3, 4 にあるので 2/5[個]。w2 = 7/5[個]

w3 = 3/5[個]

よって WIP = 3/5 + 7/5 + 3/5 = 13/5 [個]

よって　平均リードタイム　LT = WIP/TH = 13/5 * 1/12 = 13/60 [時間]

（ケース 4-2）

Cust の満期時間＝6 分／個、初期の各作業の WIP を 1 個とする。

time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
0	1(6)	1	----	1	----	1	----	0
	.....（途中略）.....							
18	1(6)	0	1(3)	0	1(2)	0	----	4
20	1(4)	0	1(1)	0	----	1	----	4
20	1(4)	0	1(1)	0	----	0	1(3)	4
21	1(3)	0	----	1	----	0	1(2)	4
21	1(3)	0	----	0	1(5)	0	1(2)	4
23	1(1)	0	----	0	1(3)	0	----	5
24	1(6)	1	----	0	1(2)	0	----	5
24	1(6)	0	1(3)	0	1(2)	0	----	5

時刻 18 から 23 までの 6 時点の平均値を求める。

w1 = 3/6 = 1/2 [個]

w2 = 5/6 [個] （□時刻 18, 19, 21, 22, 23 に在庫あり）。

w3 = 3/6 = 1/2 [個]

よって　WIP = 11/6 [個]

よって　平均リードタイム　LT = WIP/TH = 11/6 * 1/10 = 11/60 [時間]

（ケース　4-3）

Cust の満期時間＝ 5 分／個、初期の各作業の WIP を 7 個とする。

time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
0	1(5)	6	1(3)	6	1(5)	6	1(3)	0
	.....（途中略）.....							
45	1(5)	0	1(3)	13	----	1	----	15
45	1(5)	0	1(3)	12	1(5)	1	----	15
45	1(5)	0	1(3)	12	1(5)	0	1(3)	15
48	1(2)	0	----	13	1(2)	0	1(0)	15
48	1(2)	0	----	13	1(2)	0	----	16
50	1(0)	0	----	13	----	1	----	16
50	1(5)	1	----	13	----	1	----	16
50	1(5)	0	1(3)	13	----	1	----	16

時刻 45 から 49 までの 5 時点の平均値を求める。

w1 = 3/5 [個]

w2: act2 は 1 [個]。

p2Q は時刻 45, 46, 47 に 12 個。時刻 48, 49 に 13 個あるので

12*3/5 + 13*2/5 = 62/5[個]。w2 = 72/5[個]

w3 = 3/5 [個]

よって　WIP = 78/5 [個] = 15+3/5 [個]

よって　平均リードタイム　LT = WIP/TH = 78/5 * 1/12 = 390/60 [時間]

（ケース　4-4）

Cust の満期時間＝6 分／個、初期の各作業の WIP を 7 個とする

time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
0	1(6)	6	1(3)	6	1(5)	6	1(3)	0
	....（途中略）....							
378	1(6)	0	1(3)	0	1(2)	0	----	82
380	1(4)	0	1(1)	0	----	1	----	82
380	1(4)	0	1(1)	0	----	0	1(3)	82
381	1(3)	0	----	1	----	0	1(2)	82
381	1(3)	0	----	0	1(5)	0	1(2)	82
383	1(1)	0	----	0	1(3)	0	----	83
384	1(6)	1	----	0	1(2)	0	----	83
384	1(6)	0	1(3)	0	1(2)	0	----	83

時刻 378 から 383 までの 6 時点の平均値を求める。

w1 = 3/6 = 1/2 [個]

w2 = 5/6 [個]　（□時刻 378, 379, 381, 382, 382 に在庫あり）。

w3 = 3/6 = 1/2 [個]

よって　WIP = 11/6 [個]

よって　平均リードタイム　LT = WIP/TH = 11/6 * 1/10 = 11/60 [時間]

これはケース 4-2 と同じ定常状態である。

7.3　買掛金プロセス

問 7-3

下の買掛金決済のビジネスプロセスにリトルの定理を適用できるか。理由は？

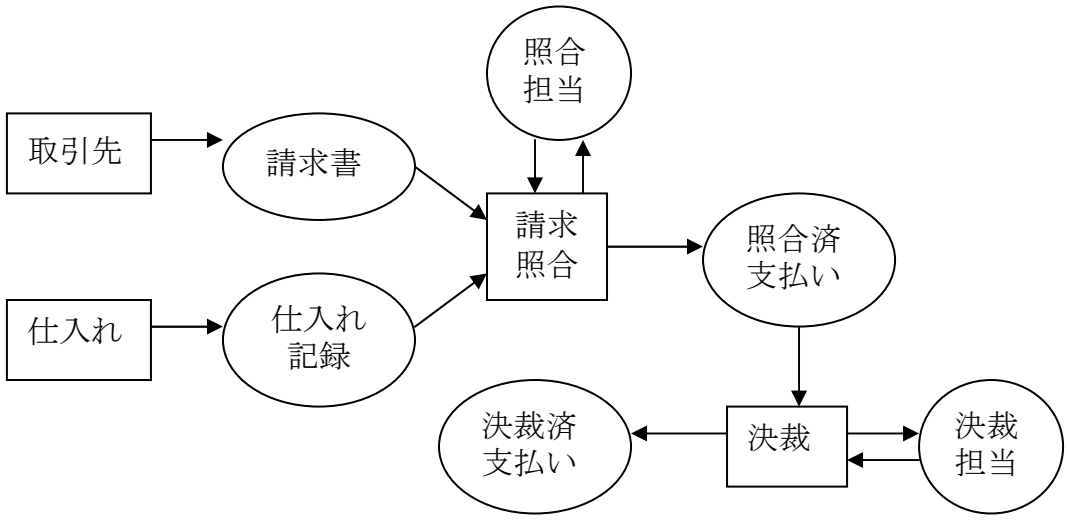


図 4-2. 買掛金決済プロセス（再掲）

時刻 ta	取引先 15	請求書 Q	仕入係 7	仕入記録 Q	照合担当	照合 10	照合済請求 Q	決裁担当 Q	決裁 29	決裁済請求 Q
220	(1,210)	0	(1,217)	115	1	-照合-	10	0	(1,203)	7
224	(1,210)	0	(1,224)	116	1	-照合-	10	0	(1,203)	7
225	(1,225)	1	(1,224)	116	1	-照合-	10	0	(1,203)	7
225	(1,225)	0	(1,224)	115	0	(1,225)	10	0	(1,203)	7
231	(1,225)	0	(1,231)	116	0	(1,225)	10	0	(1,203)	7
232	(1,225)	0	(1,231)	116	0	(1,225)	10	1	-決裁-	8
232	(1,225)	0	(1,231)	116	0	(1,225)	9	0	(1,232)	8

表 4-1. 買掛金決裁プロセスの状態遷移表の一部（再掲）

7.4　平衡状態と周期性

7.4.1　定義と命題

ビジネスプロセスの動的な特性をリトルの定理に基づいて考察するためには、システムの動作が「定常的」でなければならない。つまり、平均在庫や平均リードタイムなどの平均的な値が一定であることが必要である。この意味の定常状態を平衡状態と呼ぶ。

一般に、ランダム要素を含まない決定論的なシステムの動作に対してリトルの定理を適用するためには、システムが不動点に到達していたり、周期をもって繰り返し動作をしているなどの平衡状態にあることが十分条件である。つまり周期的動作をしていればリトルの定理を適用できる。

一般の業務取引システムの動特性を計算する方法はまだ開発されていないため、シンプル業務取引システムを定めてその周期性を分析する。

定義 1　シンプル業務取引システム

DEVS 結合システムであって次の 3 つの条件を満たすとき、シンプル業務取引システムとよぶ。

(1) 待ち行列変数への入力と出力の矢印はそれぞれ一つだけである;

(2) シンプル業務取引システムにおいては、各待ち行列変数が保持するオブジェクトをトークン（しるしとか偽コインという意味）と呼ぶことにする。トークンが移動する単位は 1 である。つまり、活動開始によって入力からとられるトークンは一度に 1 個ずつであり、また、活動終了の場合もトークンは一度に 1 個ずつ出力される。

(3) 孤立して入出力を持たない活動や待ち行列変数はない。AID は全体として必ず他とつなぐ矢印がある（連結性)。

また、業務取引システムとの対応関係を念頭において、待ち行列変数のことを概念ファイルとも呼ぶことがある。

図 7-3 はシンプル業務取引システムの AID の例である。

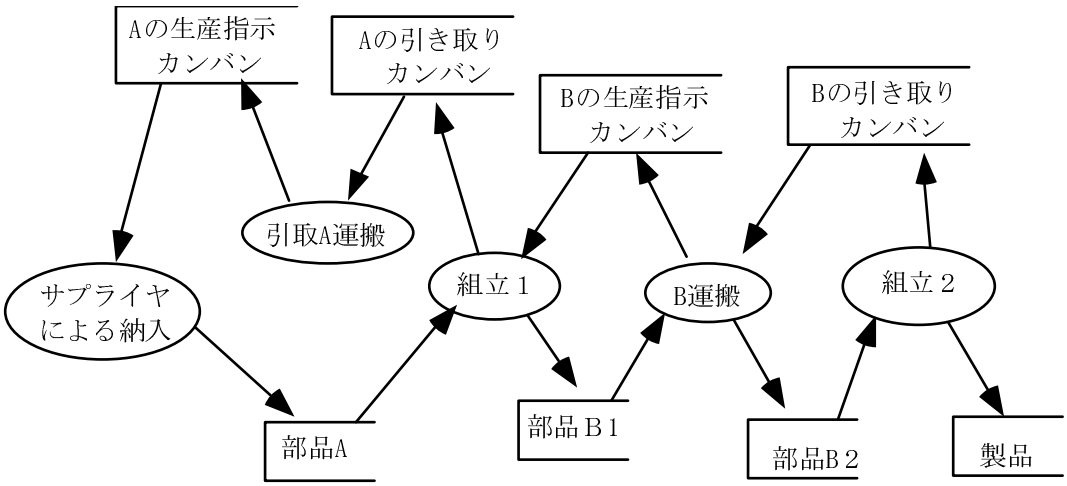


図 7-3　シンプル業務取引システムの例（カンバン方式による組立管理）

さらに活性という性質を考える。

定義２　活性

DEVS 結合システムが（したがって業務取引システムが）ある状態 s について活性であるとは、初期状態を s としたときの状態遷移表が永久に続いているものをいう。また、ある状態について活性なとき、DEVS 結合システムが活性であるという。

活性であることを形式的に定義しておく。第４章で定義した DEVS 結合システムは静的構造である AID と動的構造である $<S,\delta,ta>$ をもっているひとつの DEVS であった。Zeigler(1975）が用いた次の記号を導入する。

$d:S\times I^{+}\to S$,（I^{+}は 0 以上の整数）

$d(s,0)=s,\ d(s,k+1)=\delta(d(s,k))$（すべての $k\geq 0$ に対して）

$\Sigma:S\times I^{+}\to \Re^{\infty}$,（ここで $\Re^{\infty}$ は無限大も含む実数の集合）

$$\Sigma(s,0)=0,\ \Sigma(s,k)=\sum_{p=0}^{k-1} ta(d(s,p))$$（すべての $k\geq 0$ に対して）

つまり $\Sigma(s,k)$ は s を初期状態としたときの k 回目の内部遷移が発生する時刻である。初期状態 s について活性であるとは、

$(\forall k\in I^{+})(\Sigma(s,k)<\infty)$

が成立することである。さらに、状態 s について活性であれば、有限時間消費の

条件から、任意の時刻 t に対して次が成立することも分かる：

$$(\exists k, n \in I^{+})(\Sigma(s,k) \le t < \Sigma(s,k+n))$$

活性の時には、永久に開始しない活動があればそれを取り除いたシンプル業務取引システムを考える。取り除く場合に、その活動の入出力アローも同時に取り除く。

DEVS 結合システムは静的構造としての AID（activity interaction diagram)を持つが、AID のつながり方に注目する。AID は第 4 章の表現のとおり $<Q_{id}, E, A, f_{EK}, f_{AK}, f_{KA}>$ として表現される。AID におけるパス(path）とは矢印の向きに沿った $A \cup E$ の要素の列である。パスにおいて初めの活動と最後の活動が同じときサーキットと呼ぶ。この定義によれば孤立した活動もサーキットとなるが、本書で考える DEVS 結合システムは孤立点を含まないもののみを考える。また、元の AID の部分集合を部分 AID（または部分 DEVS システム）と呼ぶ。

定義　AID の強連結性

DEVS 結合システムの静的構造（AID）を $<Q_{id}, E, A, f_{EK}, f_{AK}, f_{KA}>$ とする。任意の活動記号 $a \in A$ と待ち行列記号 $q \in Q_{id}$ に対して、矢印の向きにしたがって到達できるような a から q へのパスと q から a へのパスがそれぞれあるとき、その静的構造は強連結という。また、強連結な静的構造を持つ DEVS 結合システムを強連結であるという。

図 7-3 は強連結なシンプル業務取引システムの例である。

つぎに静的構造のつながり合った部分を定義する。そのために、AID の中で活動からの出力の矢印を表現している関数 $f_{AK} : A \to P(KS)^{+}$ を関係と見なす。つまり、

$$(a,q) \in f_{AK} \subseteq A \times Q_{id} \leftrightarrow f_{AK}(a) = q$$

である。同様にして、活動への入力の矢印は

$$(a,q) \in f_{KA} \subseteq A \times Q_{id} \leftrightarrow f_{KA}(a) = q$$

次のように表せる。また、矢印を向きの順方向を守りながらつないだものを、その AID のパス（path）と呼ぶ。

定義　強連結な部分 AID

任意の静的構造（AID）を$<Q_{id},E,A,f_{EK},f_{AK},f_{KA}>$とする。

活動からの出力の矢印を表現する関数$f_{AK}:A\to P(KS)^{+}$の部分集合を$R_{AQ}\subseteq A\times Q_{id}$と表わす。活動への入力矢印を表す関数$f_{KA}:A\to P(KS)^{+}$については$R_{QA}\subseteq Q_{id}\times A$を用いて、

$$(q,a)\in R_{QA}\to f_{KA}(a)=q$$

を満たすものとする。

$G\subseteq A\cup Q_{id}$は活動名と待ち行列変数名の部分集合である。

AID の部分グラフ（部分構造）である$<G,R_{AQ},R_{QA}>$が元の AID のひとつの極大強連結部分であるとは、次の３つの条件を満たすことである。

(1) Gの任意の待ち行列変数と活動は、互いに矢印を順報告にたどって、つながっていること。つまり、任意の$q,a\in G\subseteq A\cup Q_{id}$について、$q$から$a$へのパスが$R_{QA}$の要素だけで構成でき、かつ、$a$から$q$へのパスが$R_{AQ}$の要素だけで構成できる。

(2) 任意の$q\in Q_{id}$に対して、ある活動$a\in G$があって、qからaへのパスとaからqへのパスとがf_{KA}とf_{AK}を適宜用いて構成できる場合には、それらのパスがすべて$R_{AQ}\cup R_{QA}$に含まれること、および、qも含めてそれらのパスに現れた活動と待ち行列変数もGに含まれること。

(3) 任意の$a\in A$に対して、ある待ち行列変数$q\in G$があって、qからaへのパスとaからqへのパスとがf_{KA}とf_{AK}を適宜用いて構成できる場合には、それらのパスがすべて$R_{AQ}\cup R_{QA}$に含まれること、および、aも含めてそれらのパスに現れた活動と待ち行列変数もGに含まれること。

上の定義の意図は図 7-4 に示すように簡単であり、要するにお互いにパスがある活動と待ち行列変数を、パスを構成する矢印ごと取り出したものである。定義により、もしある活動が（または待ち行列変数が）Gに含まれないということは、活動とGの中の待ち行列変数との間には、パスが存在しないか、一方向のパスだけが存在する。

一般にひとつの AID において強連結部分は複数あり、次の命題のように複数の強連結部分は一意に定まる。

パスの起点と終点が同一の活動（または同一の待ち行列変数）である場合に、そのパスをサーキットと呼ぶ。

図 7-2 で例を述べる。「顧客・注文１・組立１・作業者１・組立１・部品１」

はひとつのパスである。「顧客・注文１・組立１・部品１」は別のパスである。「作業者１・組立１・作業者１」はサーキットである。また、３つの強連結部分を持ち、「作業者１・組立１・作業者１」、「作業者２・組立２・作業者２」、「作業者 F・組立 F・作業者 F」はそれぞれ極大強連結部分である。

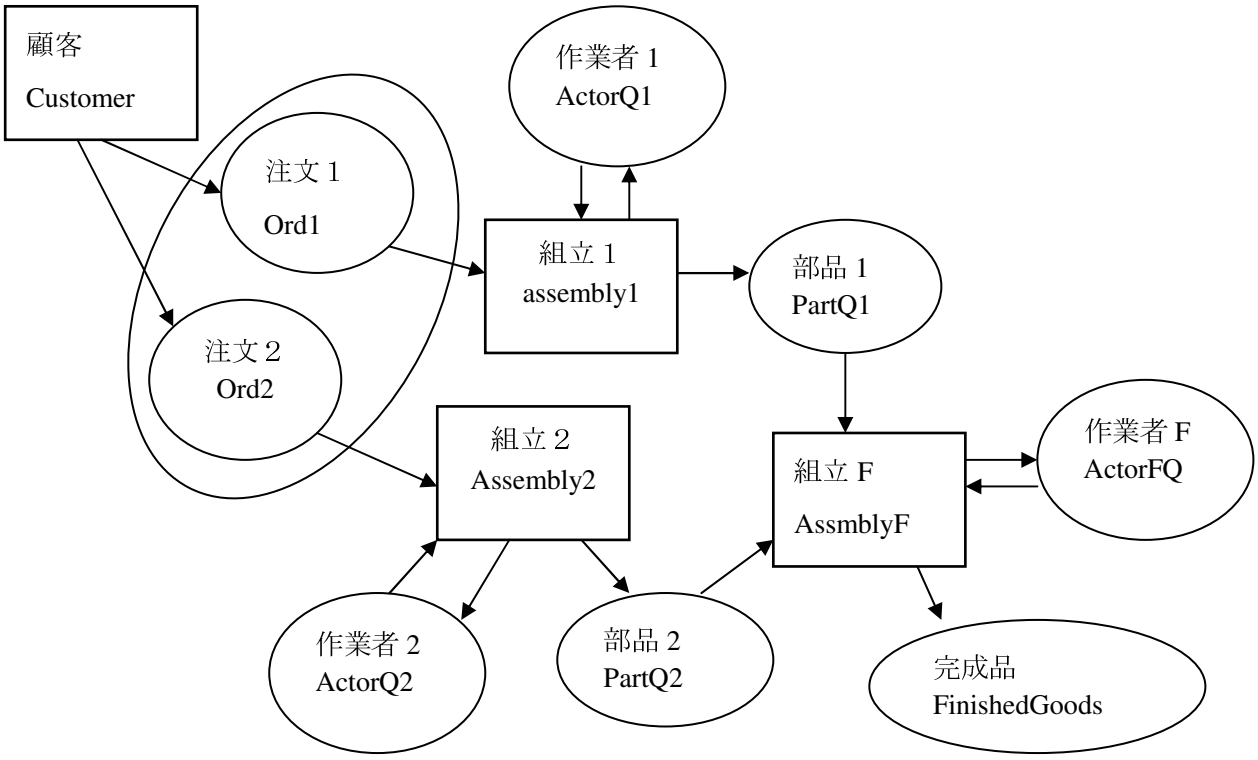


図 7-24 シンプル業務取引システムのサーキットと強連結部分

命題１

シンプル業務取引システムの静的構造は、有限個の極大強連結部分にユニークに分かれる。

（証明）

任意の AID をとる。その中に含まれるサーキットで、重複する矢印を持たないものを集めて $\{C_i \mid i=1,2,...,h\}$ とする。$A \cup Q_{id}$ の要素をノードと呼ぶ。$\{C_i \mid i=1,2,...,h\}$ を共通のノードを持たないグループに分ける。このようなグループ分けは常に可能である。k 個のグループに分けられたとする。つまり、

$$\{C_i \mid i=1,2,...,h\} = \{C^1{}_i\}_{i \in I_1} \cup \{C^2{}_i\}_{i \in I_2} \cup ... \cup \{C^k{}_i\}_{i \in I_k}$$

とする。このとき、$C,C' \in \{C^m{}_i\}_{i \in I_m}$ ならば、$\hat{C},\hat{C}' \in \{C^m{}_i\}_{i \in I_m}$ があって $C \cap \hat{C} \neq \phi$ かつ $C' \cap \hat{C}' \neq \phi$ である。異なるグループのサーキットは共通なノードを持たないことは明らかである。

各グループごとに、ノードと矢印を合わせると、ひとつの強連結部分ができる。実際、$\{C^m{}_i\}_{i \in I_m}$ を任意にとる。$\{C^m{}_i\}_{i \in I_m}$ のノードを集めた集合を G_m とし、$\{C^m{}_i\}_{i \in I_m}$ の各サーキットの矢印をすべて集めたものを R^m_{AQ}, R^m_{QA} とする。各サーキットは共通ノードを持つので、任意の２つのノード $g,g' \in G_m$ をとると、g を含む $\{C^m{}_i\}_{i \in I_m}$ のひとつのサーキットから始まって、いくつかの $\{C^m{}_i\}_{i \in I_m}$ のサーキットを経由して g' へ到達できる。逆向きのパスも同様に存在する。つまり、$< G_m, R^m_{AQ}, R^m_{QA} >$ は強連結である。極大性も明らかである。

（証明終わり）

命題２

シンプル業務取引システムのどの強連結部分にも含まれない待ち行列変数が保持するトークン数が、任意の時間において一定の個数以上にならないとする。このとき、シンプル業務取引システムが時間変化する過程で持ちうるトークンの総数は上限を持つ。

（証明）

命題１によって、システムが有限個の強連結部分を持つ。その中の任意の強連結部分 $< G_m, R^m_{AQ}, R^m_{QA} >$ において、任意の時刻で G_m のすべての待ち行列変数のトークン数が一定であることを示す。

G_m を簡単のために単に G とかく。サーキットに分解して $G = \cup\{C_i \mid i = 1,2,...,h\}$ とする。任意の $C \in \cup\{C_i \mid i = 1,2,...,h\}$ はシンプル業務取引システム内のサーキットであるので、C 上のトークン総数は、どんな活動の開始や終了によっても変わらずに一定数である。よって状態遷移の任意の時点で C 上のトークン総数を $|C|$ と書くことができる。また、$|\cup\{C_i \mid i = 1,2,...,h\}|$ も一定数である。

$G = \cup\{C_i \mid i = 1,2,...,h\}$ であるが、G 上のトークン総数を数える場合はいくつかのサーキットで重複して数える可能性があるので、各サーキットごとの総和よりは小さくなる。つまり、$|G| \leq |\cup\{C_i \mid i = 1,2,...,h\}|$ となる。

Q.E.D.

上の命題のような「シンプル業務取引システムのどの強連結部分にも含まれ

ない待ち行列変数が保持するトークン数が、任意の時間において一定の個数以上にならない」とき、その業務取引システムが有界結合であると言う。

補題１

活性で有界結合なシンプル業務取引システムをとる。ある正整数 K があって，任意の活動が任意の時刻から引き続いて開始（もしくは終了）するまでに発生しうるイベント数は K 以下である。イベントとは状態遷移表における行の数である。つまり、

$(\exists K)(\forall a \in A)(\forall n)(|event(a(n),a(n+1))| \leq K)$ である。

ただし A は活動の集合であり、また、$|event(a(n),a(n+1))|$ は状態遷移表において、活動 a の n 回目の開始時刻から n+1 回目の開始の間にある行の数である。

（証明）

１．Q_{id} を結合待ち行列変数の集合とする。命題２によって存在するトークン数の上限値を M とする。$|Q_{id}| \times |A| \times |M| << K$ なる K をとる。

２．任意の活動 a をとる。ある時刻で a が n 回目の活動開始をしたとする。

３．このとき、$|event(a(n),a(n+1))| > K$ と仮定する。

４．上の 3. の不等式を満たすイベントで、いちばん多く開始または終了している活動を s とし、その活動開始の回数を K' と書く。すると $K' \geq |event(a(n),a(n+1))|/(|P| \times |A|) > K/(|Q_{id}| \times |A|) >> M$ より、$K' >> M$ である。

５．命題 2 によって、どの活動や待ち行列変数も保持しうるトークン数は高々 M であるから、K'回の開始（または終了）があったということは活動 s は K'個のトークンを入力として取り込んだ。初期に s が持っていたトークン数を j とすると、$K' >> (j+M)$ 個以上のトークンが出力されているから、活動 s の開始が M 回以上起こっている。

６．すると、s のすべての入力概念ファイルにほぼ K'回にわたってトークンが補給されている。なぜなら、$K' >> (M+M)$ なので初期トークン数$(<M)$によらずに、入力概念ファイルは K'個のトークンが補給されている。

７．s の任意のひとつの入力概念ファイルを出力とする活動を s の直前活動と呼ぶ。入力概念ファイルは K'個のトークンが補給されたということは、直前活動が K'回出力したことである。ここで、$K' >> M$ であることから、直前活動はほぼ K'回開始した。

８．このようにして直前活動をたどっていくと、aを含む強連結部分の中で活動aにたどり着くことができ、同じ理由によって、aもほぼK'回開始していなければならない。これは、矛盾である。

９．よって、$(\exists K)(\forall a \in A)(\forall n)(|event(a(n),a(n+1))| \leq K)$ を得る。

Q.E.D.

命題３

活性で有界結合なシンプル業務取引システムの状態表において、活動の残り時間の値は，有限通りの場合しかない。

（証明）

ビジネスプロセス内の活動の個数を n とする。

活性なのである初期値からいつまでもイベントが起こり，状態表は無限に続いている。

シンプル業務取引システムの状態遷移の仕組みから，任意のイベントから次のイベントまでの時間は，活動の集合 A についての満期までの残り時間の最小値である。これは

$$min\{lag_i - e_i \mid 0 < lag_i - e_i \neq \infty, 1 \leq i \leq n\}$$

と表される。ただし、lag_iは活動iの保持時間、e_iは活動iが前回にあるトークンの保持を開始してからの経過時間である。（ひとつの活動が複数のトークンをk個持っているときには、トークンごとに$lag_i - e_i^m, 1 \leq m \leq k,$を考える。）

すべての活動の（トークンの）経過時間 e_i は，時刻０においては０であり，その後，満期に至らない間は，各イベント時刻において

$$min\{lag_i - e_i \mid 0 < lag_i - e_i \neq \infty, 1 \leq i \leq n\}$$

という量だけ減算されていく。満期には$0 = lag_i - e_i$となって終了イベントを発生し、開始されるときに経過時間は０に設定される。

したがって，トークンを持つときの満期までの残り時間は，各プレースの保持時間 $lag_i, 1 \leq i \leq n,$ を適当な回数だけ加減した式で表現できるものになる。つまり，トークンを持つ任意の活動pの終了までの残り時間は，任意の時刻において$a_1, a_2, ..., a_n$ を整数として， $a_1 lag_1 + a_2 lag_2 + ... + a_n lag_n$とかけ、イベントが起これば必ずどれかの$a_i$ の値が変化する。

補題１から、ある$K > 0$があって、任意の活動が任意の時刻から引き続いて開

始（もしくは終了）するまでに発生しうるイベント数はK以下でなければならない。つまり，任意の時刻において任意の活動の残り時間を表わす式である$a_1 lag_1 + a_2 lag_2 + ... + a_n lag_n$において，各$a_i$が変化する回数は$K$回以下である。すべての活動に対して最大値をとればこのようなKが共通に定まる。

さらに、どれかのa_iの絶対値が無限に大きくなることはない。なぜなら、a_iの絶対値が無限に大きくなるということは，活動p以外の活動が無限に開始と終了を行うことであり，この活動pが発火しないことになって，基本業務取引システムが活性であることに反するからである。

任意の活動について、満期までの残り時間として取りうる値は有限通りの場合しかない。

Q.E.D.

命題４

活性で有界結合なシンプル業務取引システムは周期を持つ。

（証明）

活性なのでいつまでもイベントが起こり，状態遷移表は無限に続いている。状態遷移表の各行には，各々の活動について，活動が保持しているトークンと，それらトークンが利用可能になる満期までの残り時間がかかれている。なお，活動以外のプレースである概念ファイル内のトークンは常に利用可能である（満期までの残り時間が常に０である）。

命題２より，ある数Jがあって、状態遷移表の１個の活動や概念ファイルに入りうるトークンは高々J個である。

命題３より，状態表に書かれる残り時間は，（各活動が複数のトークンを保持する場合も含めて）各活動に対して有限通りしかない。その最大値をKとする。

したがって，状態遷移表の活動や概念ファイルにかかれうる（トークン数，残り時間）という組は，トークンひとつひとつを識別して記述しても、高々$J*K$通りの種類しかない。一方、状態表は無限の長さをもつので，かならず同じ行が現われる。

Q.E.D.

7.4.2　周期的振る舞いとリードタイム計算

図 7-4 に示された２つの部品を組み合わせて製品にするためのシンプル業務取引システムの周期的な動きを観察してみよう。図において各活動の加工時間（所要時間）は次のように設定されている。

組立１：　3 分／個

組立２：　57 分／個

組立 F：　33 分／個

顧客からの注文の間隔の大小によって、シンプル業務取引システムが有界結合になることもあればならないこともある。有界結合であれば周期的状態において成立するリトルの定理によってプロセス全体のリードタイムが計算される。

（１）顧客からの注文間隔が 65 のときの平衡的動作

初期値として、顧客からの注文のうちで初期に未処理の注文 1 の個数が 2、注文 2 の初期未処理個数が 5、部品 1 と部品 2 のの初期在庫がそれぞれ 2、最終製品の初期在庫が 1 とする。図 7-4 のプロセスでは全体が強連結ではない。この初期値から状態遷移表によって動的変化を調べる。すると、有界結合であって周期状態となっている。時刻 2,405 から 2,470 の 1 周期のようすを示す。

time	Cust	Ord1	Ord2	A1Q	Asm1	Pt1Q	A2Q	asm2	Pt2Q	AcFQ	AsmF	FingsQ
2405	(1,65)	1	1	1	-asm1-	0	1	-asm2-	1	1	-asmF-	41
2405	(1,65)	0	1	0	(1,3)	0	1	-asm2-	1	1	-asmF-	41
2405	(1,65)	0	0	0	(1,3)	0	0	(1,57)	1	1	-asmF-	41
2408	(1,62)	0	0	1	-asm1-	1	0	(1,54)	1	1	-asmF-	41
2408	(1,62)	0	0	1	-asm1-	0	0	(1,54)	0	0	(1,33)	41
2441	(1,29)	0	0	1	-asm1-	0	0	(1,21)	0	1	-asmF-	42
2462	(1,8)	0	0	1	-asm1-	0	1	-asm2-	1	1	-asmF-	42
2470	(1,65)	1	1	1	-asm1-	0	1	-asm2-	1	1	-asmF-	42

表 7-1 注文間隔 65 分のときの 1 周期の状態遷移

プロセス全体を律しているのは、AID の構造から直感的に組立 F である。その組立 F の起動を決定しているのは、組立１の活動である。その様子は、上の状態遷移表で矢印とアミかけで示した部分である。

リードタイム計算１：組立１＋組立 F のパスについて。

TH は周期から決定される。TH = 1/65 [個／分]

リードタイムは状態遷移表から組立１＋組立 F のパスについて見て取れる。

Asm1(3)+AsmF(33) = 36

または、2441 - 2405 = 36 分である。

そのパスに沿った 1 周期間の平均在庫は、(Ord1, Asm1）と（PartQ1, AsmF）について 1 周期分を観察することにより、W_{1F} = (3+33)/65 である。

よってリードタイムは　LT = W_{1F}/TH = 36 [分]

この例では、状態遷移表が得られている時にリードタイムを調べるには、AID の各活動の可能な連鎖において、開始から終了までがいずれもブロックされない（待たされない）ような連鎖を見つけてそれらの活動の満期時間の和をとればよい。定常的な待ち行列（WIP）がないので、満期時間の和がすなわちリードタイムとなる。

まちがえてリードタイム計算を組立２＋組立 F のパスで行うと、W_{2F}=57/65 個。LT＝57[分]（誤り）となる。実際、状態遷移表でトレースすると、新たにオブジェクトが入った場合に 36 分で完成するため、57 分ではない。

（２）顧客からの注文間隔が 58 分のときの平衡的動作

この場合には、（１）と同様である。

time	Cust	Ord1	Ord2	A1Q	Asm1	Pt1Q	A2Q	asm2	Pt2Q	AcFQ	AsmF	FingsQ
26390	(1,58)	1	1	1	-asm1-	0	1	-asm2-	1	1	-asmF-	459
26390	(1,58)	0	1	0	(1,3)	0	1	-asm2-	1	1	-asmF-	459
26390	(1,58)	0	0	0	(1,3)	0	0	(1,57)	1	1	-asmF-	459
26393	(1,55)	0	0	1	-asm1-	1	0	(1,54)	1	1	-asmF-	459
26393	(1,55)	0	0	1	-asm1-	0	0	(1,54)	0	0	(1,33)	459
26426	(1,22)	0	0	1	-asm1-	0	0	(1,21)	0	1	-asmF-	460
26447	(1,1)	0	0	1	-asm1-	0	1	-asm2-	1	1	-asmF-	460
26448	(1,58)	1	1	1	-asm1-	0	1	-asm2-	1	1	-asmF-	460

表 7-2 注文間隔 58 分のときの 1 周期の状態遷移

（３）顧客からの注文間隔が 57 分のときの平衡的動作

注文間隔が１分違うだけで、業務取引システムの構造と共鳴するようなことが起こり、様子が大きく変わる。

time	Cust	Ord1	Ord2	A1Q	Asm1	Pt1Q	A2Q	asm2	Pt2Q	AcFQ	AsmF	FingsQ
26334	(1,57)	1	5	1	-asm1-	4	1	-asm2-	1	1	-asmF-	462
26334	(1,57)	0	5	0	(1,3)	4	1	-asm2-	1	1	-asmF-	462
26334	(1,57)	0	4	0	(1,3)	4	0	(1,57)	1	1	-asmF-	462
26334	(1,57)	0	4	0	(1,3)	3	0	(1,57)	0	0	(1,33)	462
26337	(1,54)	0	4	1	-asm1-	4	0	(1,54)	0	0	(1,30)	462
26367	(1,24)	0	4	1	-asm1-	4	0	(1,24)	0	1	-asmF-	463
26391	(1,0)	0	4	1	-asm1-	4	1	-asm2-	1	1	-asmF-	463
26391	(1,57)	1	5	1	-asm1-	4	1	-asm2-	1	1	-asmF-	463
26391	(1,57)	0	5	0	(1,3)	4	1	-asm2-	1	1	-asmF-	463
26391	(1,57)	0	4	0	(1,3)	4	0	(1,57)	1	1	-asmF-	463
26391	(1,57)	0	4	0	(1,3)	3	0	(1,57)	0	0	(1,33)	463
26394	(1,54)	0	4	1	-asm1-	4	0	(1,54)	0	0	(1,30)	463
26424	(1,24)	0	4	1	-asm1-	4	0	(1,24)	0	1	-asmF-	464
26448	(1,0)	0	4	1	-asm1-	4	1	-asm2-	1	1	-asmF-	464
26448	(1,57)	1	5	1	-asm1-	4	1	-asm2-	1	1	-asmF-	464

表 7-3 注文間隔 57 分のときの 2 周期間の状態遷移

1 周期が 57 分の周期状態において（平衡状態において）矢印とアミかけで示した部分がプロセス全体を律している。つまり、オブジェクトを処理して行く活動の連鎖があって、その連鎖においてはどの活動も開始をブロックされて待たされるということがない。そのようなパスは組立 F と組立 2 である。

リードタイム計算：組立 2 ＋組立 F のパスについて。

TH は周期から決定される。TH = 1/57 [個／分]

リードタイムは状態遷移表から組立 2＋組立 F のパスについて見て取る。

平均在庫の計算：

Asm2(57)+AsmF(33) = 90[個分]（1 周期間）

および　OrdQ2 の在庫の 4 個が消費されるのに 4 周期が必要だから、4*57[個分]。

合計を 1 周期の長さで平均すると

W_{2F} =（90+4*57）/57

よってリードタイムは　LT = W_{2F}/TH =90+4*57 = 318[分]

（4）顧客からの注文間隔が 56 分のときの状態遷移

この場合は、（遅い方の）加工速度 57 より到着速度 56 が速いので、決して平衡状態にはなり得ない。Ord2 の値が無限に増えていき有界結合ではない。

time	Cust	Ord1	Ord2	A1Q	Asm1	Pt1Q	A2Q	asm2	Pt2Q	AcFQ	AsmF	FingsQ
73192	(1,56)	1	27	1	-asm1-	26	0	(1,53)	0	0	(1,29)	1284
...												
75656	(1,56)	1	28	1	-asm1-	27	0	(1,40)	0	0	(1,16)	1327
75656	(1,56)	0	28	0	(1,3)	27	0	(1,40)	0	0	(1,16)	1327
75659	(1,53)	0	28	1	-asm1-	28	0	(1,37)	0	0	(1,13)	1327
75672	(1,40)	0	28	1	-asm1-	28	0	(1,24)	0	1	-asmF-	1328
75696	(1,16)	0	28	1	-asm1-	28	1	-asm2-	1	1	-asmF-	1328
75696	(1,16)	0	27	1	-asm1-	28	0	(1,57)	1	1	-asmF-	1328
75696	(1,16)	0	27	1	-asm1-	27	0	(1,57)	0	0	(1,33)	1328
75712	(1,56)	1	28	1	-asm1-	27	0	(1,41)	0	0	(1,17)	1328

表 7-4 注文間隔 56 分のときの状態遷移

（５）初期値による違いを見るため、（１）と同じプロセスの構造でいくつかの初期値を変えてどのような平衡状態になるかを観察する。

初期値：注文１（Ord1）=20 個、注文２（Ord2）=5 個、部品１（Pt1Q）=12 個、部品 2(Pt2Q) = 12 個、完成品（FingsQ)= 1 個。

time	Cust	Ord1	Ord2	A1Q	Asm1	Pt1Q	A2Q	asm2	Pt2Q	AcFQ	AsmF	FingsQ
2080	(1,65)	1	1	1	-asm1-	18	1	-asm2-	0	0	(1,25)	45
2080	(1,65)	0	1	0	(1,3)	18	1	-asm2-	0	0	(1,25)	45
2080	(1,65)	0	0	0	(1,3)	18	0	(1,57)	0	0	(1,25)	45
2083	(1,62)	0	0	1	-asm1-	19	0	(1,54)	0	0	(1,22)	45
2105	(1,40)	0	0	1	-asm1-	19	0	(1,32)	0	1	-asmF-	46
2137	(1,8)	0	0	1	-asm1-	19	1	-asm2-	1	1	-asmF-	46
2137	(1,8)	0	0	1	-asm1-	18	1	-asm2-	0	0	(1,33)	46
2145	(1,65)	1	1	1	-asm1-	18	1	-asm2-	0	0	(1,25)	46
2145	(1,65)	0	1	0	(1,3)	18	1	-asm2-	0	0	(1,25)	46
2145	(1,65)	0	0	0	(1,3)	18	0	(1,57)	0	0	(1,25)	46
2148	(1,62)	0	0	1	-asm1-	19	0	(1,54)	0	0	(1,22)	46
2170	(1,40)	0	0	1	-asm1-	19	0	(1,32)	0	1	-asmF-	47
2202	(1,8)	0	0	1	-asm1-	19	1	-asm2-	1	1	-asmF-	47
2202	(1,8)	0	0	1	-asm1-	18	1	-asm2-	0	0	(1,33)	47
2210	(1,65)	1	1	1	-asm1-	18	1	-asm2-	0	0	(1,25)	47

表 7-5 注文間隔 65 分、多量初期受注残のときの平衡的状態遷移

表の状態遷移をみると、初期にあった注文 1 の受注残は平衡状態においては部品 1 の在庫に姿を変えており、矢印で示したように、プロセス全体のスピードを決定しているのは注文 2 から完成品への活動の連鎖であることが分かる。

問 7-4

上の（5）の場合のリードタイムを求めよ。

7.5　計画機能を持つビジネスプロセスの動特性

7.5.1 計画機能と組み立てプロセス

計画機能を持ったビジネスプロセスの例として、図 7-5 の組み立てプロセスを考える。この組み立てプロセスは、3 種類の製品 A, B, C を生産する。生産は製造オーダによって開始されるが、製造のための作業手順はそれぞれ異なる。製品 B は２つの作業からなるが、製品 A と C は 2 つの部品製造と一つの最終組立工程からなる。製品 B の２つの作業は operationB と operationF_B である。製品 A の３つの作業は operationA_1, A_2, F_A であり、製品 C も同様である。部品表は明示されていないが、各作業から生まれる部品をそれぞれ一つ用いて、次の段階の部品や製品が製造されるものとする。また、生産オーダの流れを書いていて、最初の活動で使われる材料を示してないが、材料は何らかの方法によって都合よく用意されているか、大量に用意された材料の倉庫から自由に使うものとしている。このようにして多品種のプロセスを扱うことができるが、顧客オーダは個数や納期の多様性を仮定しておらず、1 個 1 個の製品を要求するオーダが来る状況を考えている。

四角で示されている各活動に付随している数値は、1 個を加工や組み立てるのに必要な時間である。段取り時間やかたずけ時間は含まないと仮定している。また各作業の作業者数(actor)は１人である。

計画機能は生産開始を指示する生産オーダ発行機能だけを果たしており、通常は計画に含まれる在庫引き当てによる在庫調整とか需要や調達の予定変更に伴う再計画機能は含まれていない。計画サイクルが加工等の加工時間に対応するので、計画サイクルが全体のビジネスプロセスのパフォーマンスに影響を与

える。

業務取引システムは、状態遷移という普遍的な動作記述方法を使えるので、動的な性質を考察することができる。

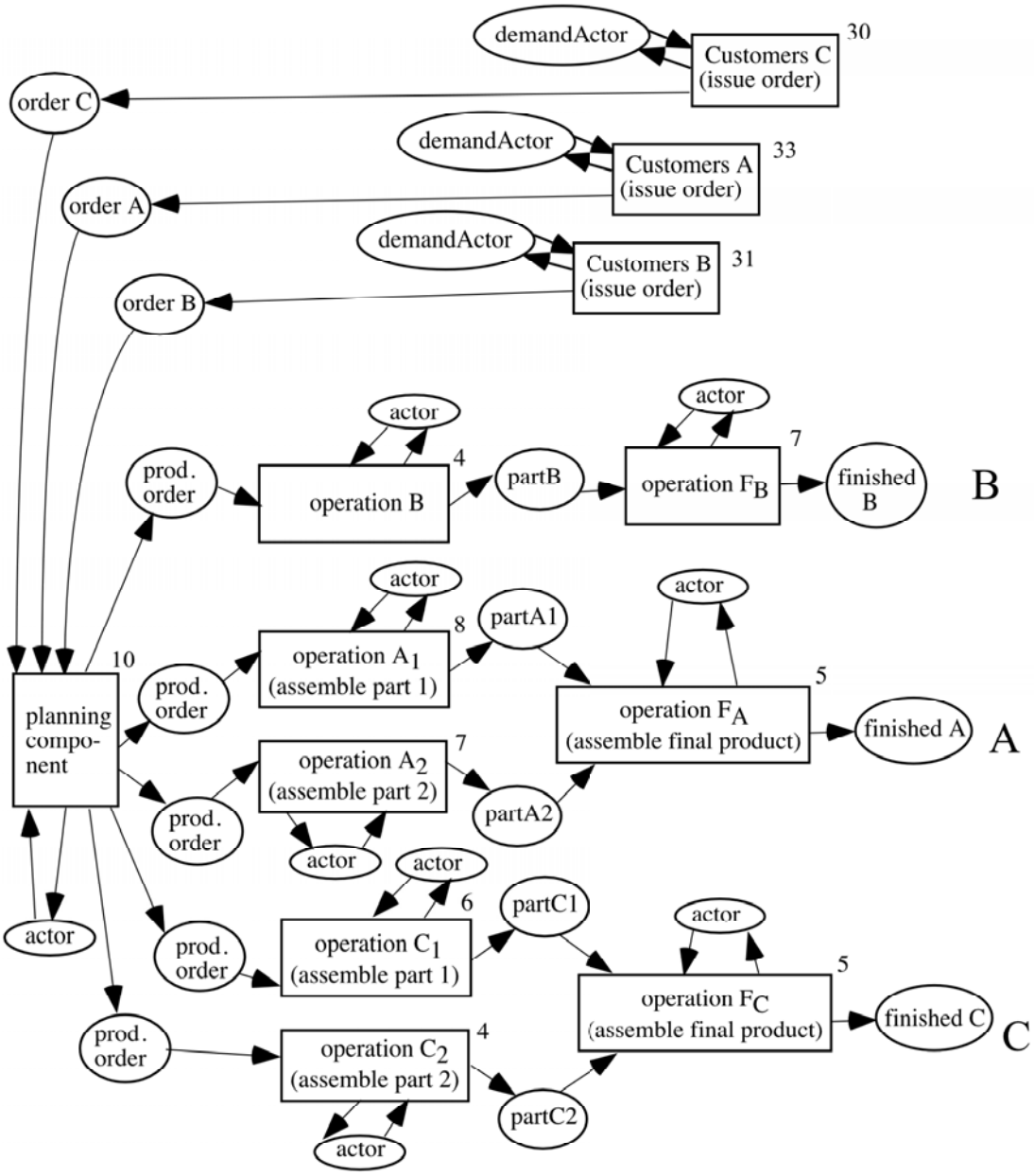


図 7-5　3 種類の作業手順からなるプロセス

7.5.2　計画リードタイムと待ち時間の変動の例

計画の影響を考えるために、計画作成・リリースの頻度と周期的動作の関連を考えることが可能である。

命題４を使うと、計画システムも活動として含めて考えたシンプル業務取引システムが周期的ふるまいを示し、平衡状態を持つ条件が分かる。

周期構造があらわれて定常状態（平衡状態）となるとリトルの定理が使える。

リトルの定理：

平均の滞留時間＝平均のシステム内のトークン個数×平均の到着時間

プロセスが周期 S で周期状態になると、1 周期内に入ってくるトークンが一定値 k になる。するとリトルの定理を使って、リードタイム L は　L = W・(S/ k) で計算される。

初期条件として顧客からの注文を A が２個、B が１個、C が２個としたとき、図 7-5 のプロセスの周期は下記の表のようになる。

time	AdmQ	ACust	BdmQ	BCust	CdmQ	CCust	pnAct	A1mQ	A1Act	A1pQ	A2mQ
99436	1	1(20)	1	1(104)	0	1(14)	1(5)	0	-A1aQ-	0	0
178606	1	1(20)	1	1(104)	0	1(14)	1(5)	0	-A1aQ-	0	0

time	A2Act	A2pQ	AfAct	AfQ	B1mQ	B1Act	B1pQ	BfAct	BfQ	C1mQ
99436	-A2aQ-	0	-AfaQ-	2368	0	-B1aQ-	0	-BfaQ-	947	0
178606	-A2aQ-	0	-AfaQ-	4253	0	-B1aQ-	0	-BfaQ-	1701	0

time	C1Act	C1pQ	C2mQ	C2Act	C2pQ	CfAct	CfQ
99436	-C1aQ-	0	0	-C2aQ-	0	-CfaQ-	2551
178606	-C1aQ-	0	0	-C2aQ-	0	-CfaQ-	4581

表 7-6 計画システムがリリース管理を行うビジネスプロセスの平衡状態

周期は 79,170 (=178606-99436) である。なお、A,B,C それぞれの完成品累計を示す AfQ, BfQ, CfQ は、周期的動作とは関係しない。

第８章　ビジネスプロセスの Max-Plus 方程式

8.1 動的特性の設計へのアプローチ

情報システムによるフィードバックを採用し、フィードバック構造を持つ業務取引ペトリネットによってビジネスプロセスの時間特性を分析する方法を考えよう。

受注から納品までの納期、平均在庫量、在庫期間、スループット（処理速度）といった量は、ビジネスシステムの重要な時間的特性である。これらは、現実に一定期間測定して平均値を求めることはできるが、ビジネスプロセスのどの部分をどういうふうに変更すると特性がどう変わるかが分かりにくい。リードタイムを例にとる。顧客リードタイムとは、顧客が発注してから製品や商品を受け取るまでの時間であり、製造リードタイムとは、生産指示オーダを製造部門に発注してから、製品が完成するまでにかかる時間である。一般的に製造リードタイムは以下のように説明される：

製造リードタイム＝加工時間＋段取り時間＋移動時間＋待ち時間

待ち時間は大きな部分を占める要素としてかならず入る。たいていは、待ち時間と段取り時間の和が、製造リードタイムの９割以上を占めていることもめずらしくないといわれている。

製造リードタイムはいちおうは定義式のように書かれるが、待ち時間が生産システムだけで定まるものでないことは実は自明である。多くの注文を抱えているところに新たに発注されたオーダの待ち時間と、生産能力に余力のある状態でのオーダの待ち時間は大きく異なる。注文や生産の混み具合は管理システムによって大きく変わるので、結局、待ち時間はビジネスプロセスの分業の仕組みと管理システムに依存している。したがって、ビジネスプロセスのふるまいを制御するためには、システムの構成要素である機能別活動のふるまいと活動を媒介する物やデータなどの在庫の蓄積状況や生産指図発行のしくみを設計する必要がある。さらに、生産の一部をアウトソーシングしている場合はもちろん、自社内で閉じている場合でも、下請けやザプライチェーンの生産能力の影響も一般には受けている。RFID（IC タグ）の記録能力によってビジネスプロセスで１個流しの管理をすることは、理論的には可能である。また、将来的に RFID に計算能力を持たせて RFIS(無線情報システム）とするかどうかは、ビジネス全体にとっての有効な使い方次第である。

生産計画ではリードタイムの値を必要とするし、また、物流計画でも管理ポイントごとのリードタイムを必要とする重要な特性である。しかし、リードタイムの値がどういう条件でどうなるかについては、その仕組みが理解されているというよりは、経験的に設定され遵守される側面が大きいといえる。

周期性を追求することで動的特性の見通しが得られる。工学原理として充分に発達している段形ではないが、特別な場合には特性を計算できる。業務取引システムの周期性を分析するために、業務取引ペトリネットというモデルを導入し、max-plus 代数という特別な「線形代数」による行列方程式を使った分析法を説明する。この行列方程式は将来ビジネスプロセス方程式として発展させられるべきものである。

８.２ 業務取引ペトリネット

本書ではビジネスプロセスについての AID や DFD を基盤としたモデルを立て、その分析のために業務取引システムやペトリネットによるモデルと max-plus 方程式の適用を行う。その目的のために、AID を静的構造として持つ業務取引システムと同じ動作をする時間ペトリネットとして業務取引ペトリネットを提唱する。これによって、ビジネスプロセスを分析する時にペトリネットの分析方法を利用する可能性が開ける。なお、本章の付録１に本章の記号の一覧表を示した。

業務取引システムの構成要素は、AID や DFD で記述できるものである。以下に定める業務取引ペトリネットはシンプル業務取引システムに対応するペトリネットである。一般的なペトリネットは村田（『ペトリネットの解析と応用』，近代科学社 1992）にしたがって次のように定められる。

定義１　ペトリネット

ペトリネットは組　$PN=(P,T,F,W,\mu_0)$　である。ここで

$P=\{p_1,p_2,...,p_x\}$　プレースの有限集合

$T=\{t_1,t_2,...,t_y\}$　トランジションの有限集合

$F\subseteq(P\times T)\cup(T\times P)$　アーク（矢印）の集合

$W:F\rightarrow\{1,2,3,...\}$ アークの重み

$\mu_0:P\rightarrow\{0,1,2,...\}$　初期マーキング

$P\cap T=\phi$ （共通のシンボルは無い）

上の定義において、xとyはそれぞれプレースの個数とトランジションの個数を表す。マーキングμ_0は任意の特定のプレースp_iについて$\mu_0(p_i)$によってp_iの中にあるトークンの個数を表す。したがって、マーキングはPから自然数の集合への関数として表されている。マーキング（トークンの配置）がされていないペトリネット構造（グラフ）をJで表わし、初期マーキングμ_0が規定されているものは(J,μ_0)で表わす。

ペトリネットの図的表現には図 8-1 のシンボルが使われる。プレースは楕円で、トランジションは線分で、アークはプレースとトランジションを接続する矢印で描かれる。アークには重みと呼ばれる正整数が附記される。重みが 1 の場合には、図的表現では省略する。

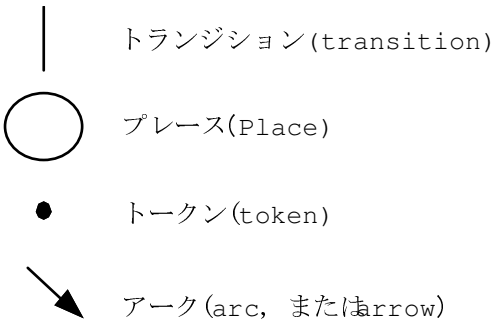


図 8-1 Petri net の構成要素

図 8-2 はある瞬間の 1 台の機械による加工のペトリネットモデルである。図でトークンは加工待ちの材料の個数を表現したり、また、機械が稼働中か空きかを示すために使われている。図 8-3 は加工開始後のトークンの様子である。

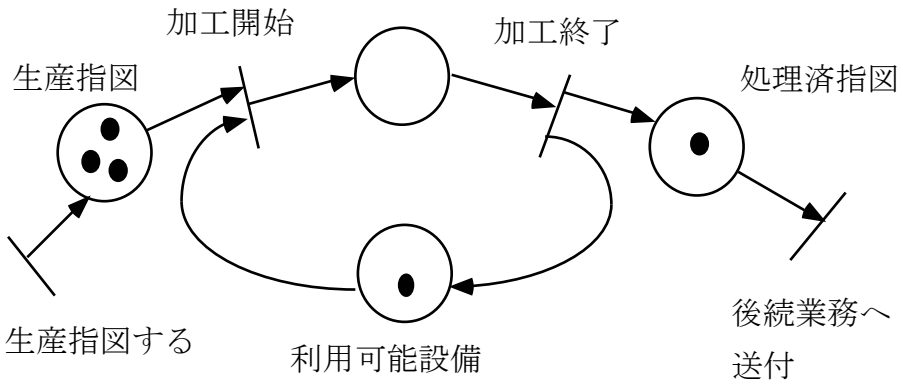


図 8-2　機械加工のペトリネット（Peterson (1981)を変更）

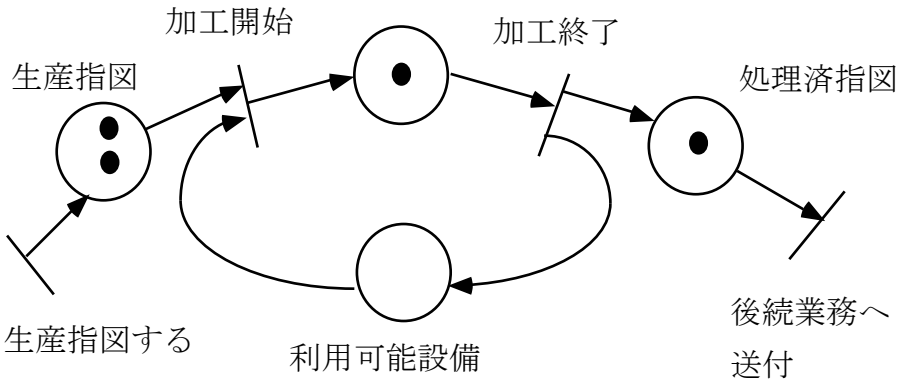


図 8-3 「加工開始」発火後のマーキングの変化

マーキングは各プレースへのトークンの分布（配置）である。マーキング μはプレースの集合から非負整数への写像と考えることができる。つまり、$\mu: P \to \{0,1,2,...\}$ である。トランジションの発火によって、トークンが重みの数だけ移動する。プレース pからトランジション tへ向かうアークがあるとき pを tの入力プレースといい $p \in I(t)$とかく。またトランジション tからプレース pへのアークがあれば pを tの出力プレースといい $p \in O(t)$とかく。

トランジション発火ルール

発火条件：トランジション tの各々の入力プレース pに $w(p,t)$ 個以上のトークンがあれば tは発火可能であるという。ここで $w(p,t)$ は pから tへのアークの重みである。
発火によるトークンの移動：tが発火すると、tの入力プレースすべてからそれぞれの重み $w(p,t)$ の分だけのトークンが取り去られる。tの出力プレースそれぞれには、重み $w(t,p)$に応じたトークンが増える。$w(t,p)$は tから pへのアークの重みである。

通常のペトリネットではトランジションの発火についての時間的な情報は組み込まれていない。プレースがトークンを保持する時間が定まっていて、トークンがそのプレースに入ってきてからその保持時間が経過すれば、トークンがそのプレースを入力とするトランジションによって利用可能になるようなペトリネットを時間ペトリネ

ット(timed Petri net) と呼ぶ。本書で扱う時間ペトリネットは、バチェリら(Baccelli, F.L., Cohen, G., Olsder, G.J., and Quadrat, JP, “Synchronization and Linearity - An algebra for discrete event systems, ” John Wiley,1992)にしたがって、トランジションのすべての入力プレースのトークンが利用可能になり発火条件を満たせば、トークン移動をともなって直ちに発火するものとする。

時間ペトリネットのサブクラスとして、業務取引ペトリネット (business transaction Petri net: BTS-PN) を次のように定める。

定義２　業務取引ペトリネット(BTS-PN)

あるペトリネット構造が業務取引ペトリネットであるとは、ペトリネットが次の５つの条件を満たすことである。

(1) |→○→| という２つのトランジションとひとつのプレースから成る部分を「活動部分」と呼ぶ。中のプレースを busy プレースと呼ぶ。活動部分の入力とは busy プレースへの入力トランジションであり、出力も同様に定義される。busy プレースへの入力アークと出力アークはそれぞれ一つしかない。それらのアークの重みは各々１とする。

(2) BTS - PN の活動部分以外は、→○→のように入力アークと出力アークをそれぞれ一つだけ持つプレースから成り、「結合部分」と呼ばれる。このプレースを結合プレースと呼ぶ。各アークの重みは１以上である。さらに、活動部分に対しては busy プレースのアークと逆向きのアークによって接続された結合プレースがある。このプレースは活動が idle 状態にあることを表現することを意図されているので、idle プレースと呼ぶ。

(3) 活動部分への入力アークは、busy プレースの入力トランジションにだけ入れる。（出力アークは入力からも出力からも出る。）

(4) ペトリネットグラフの任意のパスにおいて、活動部分と結合部分は交互に現われる。つまり BTS -PN は交代性をもつ。（活動部分の|→○→|という組み合わせをひとつの単位とみなしたときに交代性が成り立つ。）

(5) BTS -PN の busy プレースのトークン保持時間が定まっており、正の実数で表す。結合プレースは保持時間０である。

任意のプレース p について、p へ入るアークの重みを w_{pi} と書き、出るアークの重みを w_{po} と書くことにする。たとえば、busy プレース q にとっては $w_{qi} = w_{qo} = 1$ である。

図 8-4 は２つの組立工程からなる組立プロセスをアクティビティ・インタラクション・ダイアグラム(AID)として表現したものである。カンバンを使ってコントロールされている。

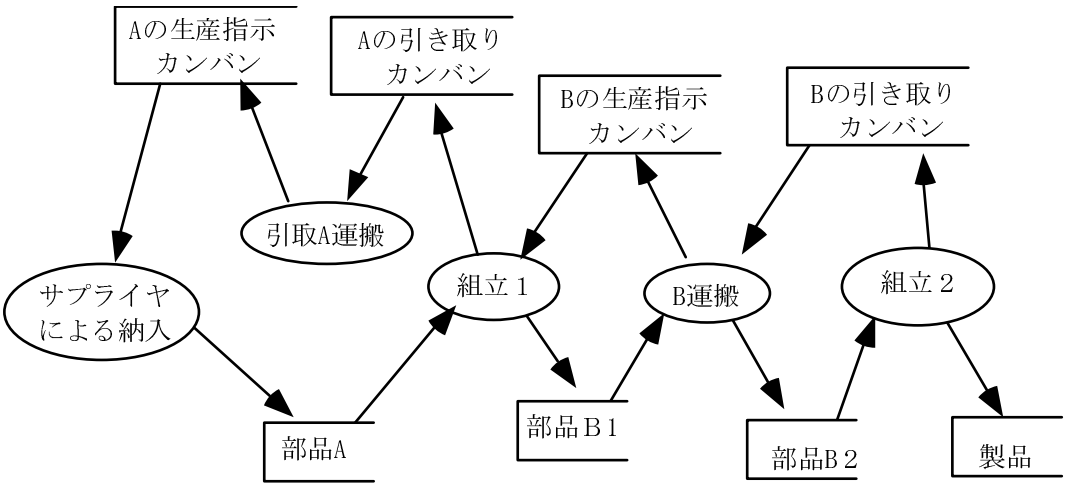


図 8-4　カンバン方式による組立プロセスの管理

図 8-5 が図 8-4 の AID から得られる業務取引ペトリネットである。シンプル業務取引システムと業務取引ペトリネットの同型的な対応は後の節で示されるが、図をみれば直感的にも納得できるものである。

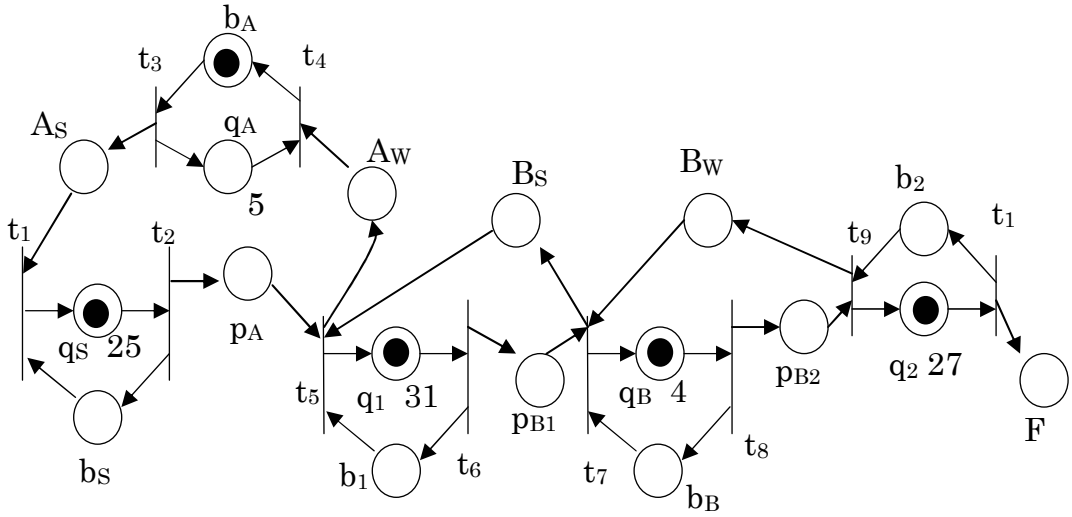


図 8-5.　図 8-4 のプロセスに対応する業務取引ペトリネット

８.３　業務取引ペトリネットの状態遷移

8.3.1 業務取引ペトリネットの状態遷移の例

図 8-5 の業務取引ペトリネット BTS-PN の発火のメカニズムはフローチャート図 8-6 で表わされる。実際の時間経過は図 8-5 の初期マーキングに対して表 8-1 の状態遷移表となる。つまり、業務取引ペトリネットのふるまいは、図 8-5 で表わされる計算法によって、状態遷移表を構成していくことで完全に記述できるのである。

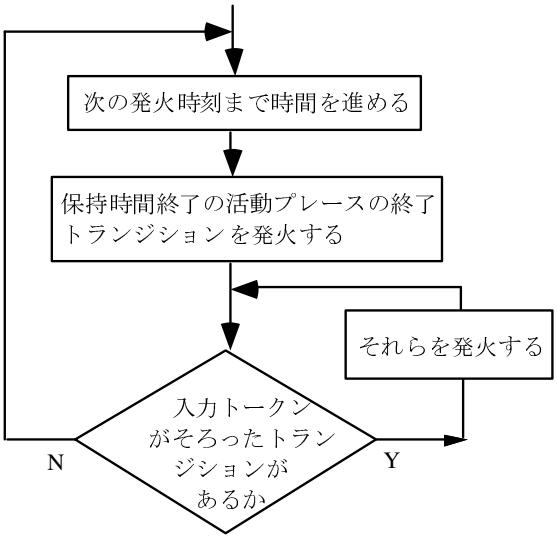


図 8-6　業務取引ペトリネットの動的ふるまい

表 8-1 において、time 欄の値はイベント時刻である。かならずどれかの活動が終了する時刻であるが、終了時に活動部分の出力トランジションが発火してトークンが移動するので、その時刻にある活動部分が開始可能になることがある。図 8-6 では右側のループで表現されるように、その場合、状態遷移表では同じイベント時刻をもった複数の行で記述されることになる。また、たとえば busy プレース *qS* の欄の l(25)は、現在1個のトークを持ち、あと 25単位時間たてばこのトークンを出力プレースに送る（つまり、保持時間が終了する）ことを表わす。

保持	0	5	0	25	0	0	4	30	0	0	0	27	0	
time	**As**	**qA**	**Aw**	**qS**	**pA**	**Bs**	**qB**	**q1**	**pB1**	**pB2**	**Bw**	**q2**	F	**発火済**
0	0	0	0	1(25)	0	0	1(4)	1(31)	0	0	0	1(27)	0	(t1t5t7t9)
4	0	0	0	1(21)	0	0	0	1(27)	0	1(0)	0	1(23)	0	t8
25	0	0	0	0	1(0)	0	0	1(6)	0	1(0)	0	1(2)	0	t2
27	0	0	0	0	1(0)	0	0	1(4)	0	1(0)	0	0	1	t10
27	0	0	0	0	1(0)	0	0	1(4)	0	0	1(0)	1(27)	1	t9
31	0	0	0	0	1(0)	0	0	0	1(0)	0	1(0)	1(23)	1	t6
31	0	0	0	0	1(0)	1(0)	1(4)	0	0	0	0	1(23)	1	t7
31	0	0	1(0)	0	0	0	1(4)	1(31)	0	0	0	1(23)	1	t5
31	0	1(5)	0	0	0	0	1(4)	1(31)	0	0	0	1(23)	1	t4
35	0	1(1)	0	0	0	0	0	1(27)	0	1(0)	0	1(19)	1	t8
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:

表 8-1　BTS-PN の状態遷移表

問 8-1

上の表 8-1 で時刻 36のときの行を書きなさい。

8.3.2 業務取引ペトリネットの状態遷移

業務取引ペトリネット(BTS-PN)のトークン配置やトークン保持時間が、トランジションの発火によってどのように変化していくかを定める。

BTS-PN の時間的な変化、すなわち状態遷移を正確に記述する。BTS-PN のペトリネット構造を任意にひとつとる。これを $J=\langle P,T,F,W\rangle$ とする。J のすべてのトランジションとプレースには予め番号がつけられている。活動部分の個数を n_A 個とするとき、まず busy プレースに番号 k $(k=1,2,3,...,n_A)$ がつけられており、それに対応して入力と出力トランジションが t_k^i, t_k^o というように入出力を区別して番号づけられている。つまり、$T=\{t_k^i, t_k^o \mid k \in \underline{n_A}\}$、ただし、$\underline{n_A}=\{1,2,3,...,n_A\}$ である。他のプレースとトランジションにも一意的に番号がつけられているものとする。

プレースの数を n とする。プレースの種類には結合プレースと busy プレースとが

ある。結合プレースの集合をP_Cとし、busyプレースの集合をP_Aとする。 $P = P_A \cup P_C$である。 P_Aの要素をq_aと書くことにする。

$N\ (=Z^+)$を0を含む自然数集合、N^nは自然数をn個並べたベクトルの集合とする。N^nの任意のベクトルμはn個のプレースの各々が含むトークンの個数を成分に持つひとつのマーキングとみなせる。たとえば$\mu(q_a)$はbusyプレースq_aの中のトークン数を表す。Rは実数全体からなる集合とし、$R^\infty = R \cup \{\infty\}$は$R$に無限大$\infty$を追加した集合とする。

R^{n_A}はn_A個の実数値を成分とするベクトルの集合を表わす。

関数$h : N^n \times P_A \to R^\infty$は、busyプレースがトークンを持ったときの保持時間(hold time)をあたえる。

$$h(\mu, q_a) = \begin{cases} h_a, \text{ if } \mu(q_a) = 1 \\ \infty, \textit{ otherwise.} \end{cases}$$

ここで、h_aはq_aのトークン保持時間を表す。業務取引ペトリネットは、保持時間がトークンに依存しないで各プレースごとに一定であるような、時間不変のtimed Petri netである。

初期状態はかならず次の初期条件を満たすものとする：

（初期条件） 任意の$q_a \in P_A$について$h(\mu, q_a) - e_a > 0$

ここで、e_aはbusyプレースq_aが現在保持しているトークンが入って来たときからの経過時間である。ベクトル$e = (e_1, e_2, ..., e_{n_A}) \in R^{n_A}$は各busyプレースが現在のトークンを持ったときからの経過時間である。トークンを持たないときはその経過時間はゼロとする。

BTS-PNの状態遷移関数

$\delta : N^n \times R^{n_A} \to N^n \times R^{n_A}$

を定める。P(T)を集合Tのべき集合とする。

$dueP : N^n \times R^{n_A} \to \mathrm{P}(T)$　という関数を次のように定める。

$dueP(\mu, e) = \{t^o_{i_1}, t^o_{i_2}, ..., t^o_{i_m}\}$

$\Leftrightarrow$ 各々の$k, 1 \le k \le m,$について次の(1)、(2)を満たすこと：

(1) $I(t^o_{i_k}) = \{q_{i_k}\}$、つまり$t^o_{i_k}$はあるbusyプレース$q_{i_k}$の出力トランジション；

(2) $h(\mu, q_{i_k}) - e_{i_k} = \min_{q_a \in P_A} \{0 < h(\mu, q_a) - e_a \ne \infty\}$.

つまり *dueP* はもっとも満期が近い将来の終了トランジションを示す関数である。

それらの終了トランジションの発火時刻まで満期時間を与える関数を

$ta(\mu,e)=\min_{q_a\in P_A}\{h(\mu,q_a)-e_a \mid 0<h(\mu,q_a)-e_a\neq\infty\}$ と定める。

これらの終了トランジションに対して busy プレースの番号の順にひとつずつトークンを busy プレースからその出力トランジションにつながるプレースへ移動し、その後、入力トランジションが発火可能かを調べて、可能なら入力側のプレースから busy プレースへトークンを移動する。つまり以下のように定義する；

. $\mu_0=\mu$ とし、各 $j, 1\le j\le m,$ について

. $\mu_j(p)=$

. $\mu_{j-1}(p)-w_{po}$, if $p=q_{i_j}$ (つまり $I(t^o_{i_j})=\{q_{i_j}\}$ のとき);

. $\mu_{j-1}(p)+w_{pi}$, if $p\in O(t^o_{i_j})$ （つまり p が $t^o_{i_j}$ の出力プレースのとき）；

. $\mu_{j-1}(p)$, otherwise. (不変)

これを使って、満期の活動の出力トランジションの発火による変化の結果を表わす関数 $\delta_1: N^n\times R^{n_A}\to N^n\times R^{n_A}$ を以下で定める。まず $\mu^1=\mu_m$ と定める。また、各 q_a に対して $e^1(q_a)=e_a+taP(\mu,e)$ とする。$\delta_1(\mu,e)=(\mu^1,e^1)$ と定める。この定義によって、満期となったためにトークンが出力された活動部分の idle プレースにトークンが入り、また、今回の時間進め分を経過時間に加える。

次に、開始可能な入力トランジションがあればそれらを発火する操作として δ_2 を定める。まず μ によって決まる発火可能な入力トランジションを集めて、それを $st(\mu)$ とする：

$st(\mu)=\{t^i_{k_1},t^i_{k_2},...,t^i_{k_r}\}\Leftrightarrow$ 任意の $m, 1\le m\le r,$ について

(1) $O(t^i_{k_m})=\{q_{k_m}\}$，つまり $t^i_{k_m}$ はある busy プレース q_{k_m} の入力トランジション；

(2) 任意の p について，$p\in I(t^i_{k_m})$ ならば $\mu(p)\ge w_{pi}$ であること。

上の条件(2)により、idle プレースにトークンを持つものだけが $st(\mu)$ の要素として開始可能になりうる。このとき、. $\mu'_0=\mu$ とし、各 $j, 1\le j\le r,$ について次とする。

$$\mu'_0(p)=\begin{cases}\mu'_{j-1}(p)-w_{po}, \text{if } p\in I(t^i_{k_j});\\ \mu'_{j-1}(p)+w_{pi}, \text{if } p\in O(t^i_{k_j});\\ \mu'_{j-1}(p), \text{otherwise.}\end{cases}$$

関数 g によって、μ から μ'_r を与えることを表現する。つまり $g(\mu)=\mu'_0$ である。
マーキングの最終的な更新結果を与える関数を k であらわすとき、

$$k(\mu)=\begin{cases}\mu, \text{ if } st(\mu)=\phi;\\ k(g(\mu)), \text{ if } st(\mu)\neq\phi.\end{cases}$$

と定める。ただし、ϕ は空集合を表す。付録の命題 1 に示すように関数 k は well-defined である。

$\delta_2 : N^n \times R^{n_A} \to N^n \times R^{n_A}$ を以下で定める。まず、

$$e^2(q_a)=\begin{cases}0, \text{if } O(t_a^i)=\{q_a\} \wedge t_a^i \in \bigcup_{p=0}^{\infty}\{st(g^p(\mu)\};\\ \text{unchanged, otherwise.}\end{cases}$$

とする。$\delta_2(\mu,e)=(h(\mu),e^2)$ と定める。

以上の準備をもとに、BTS-PN のマーキングが μ で、busy プレースの中のトークンの経過時間がベクトル e で与えられているとき、将来の最近の終了可能なトランジションがすべて発火し、かつその変化から引き続いて発火可能となるトランジションがすべて瞬時に発火した後の、マーキングと経過時間ベクトルとを与える状態遷移関数

$$\delta : N^n \times R^{n_A} \to N^n \times R^{n_A}$$

を $\delta(\mu,e)=\delta_2(\delta_1(\mu,e))$ と定める。

上で定めた業務取引ペトリネット BTS-PN の状態遷移関数の例を見てみよう。カンバンシステムは作業レベルの生産管理の方法のひとつである (門田安弘、『新トヨタシステム』、講談社、1991)。カンバン方式による生産ラインのコントロールでは、最終工程だけに生産指示が与えられる。図 8-7 の組立工程と納入業者の工程の組み合わせのように、引き続く 2 工程間で一定枚数のカンバンが使われ、各カンバンは部品収用箱とともに工程間を移動する。図 8-7 では納入業者をひとつの工程とみなしている。たとえば製品 F を組み立てる工程で部品 A が使われる時に A の外注カンバンがはずされる。A の外注カンバンがあらかじめ決められた枚数だけたまるか決められた時刻になると、外注先へ行って部品を引き取ってくることが運搬者に指示される。外注カンバンは組立の前工程である「納入業者」へ生産指示として空の部品収用箱とともに運ばれ、外注カンバンに対応する量だけの部品を製造する。後工程で部品が消費されない限り、カンバンははずされず、したがって必要のない生産は行われない。部品やカンバンの在庫状況をトークンで表している。トークンの配置状況がマーキング μ

で表される。

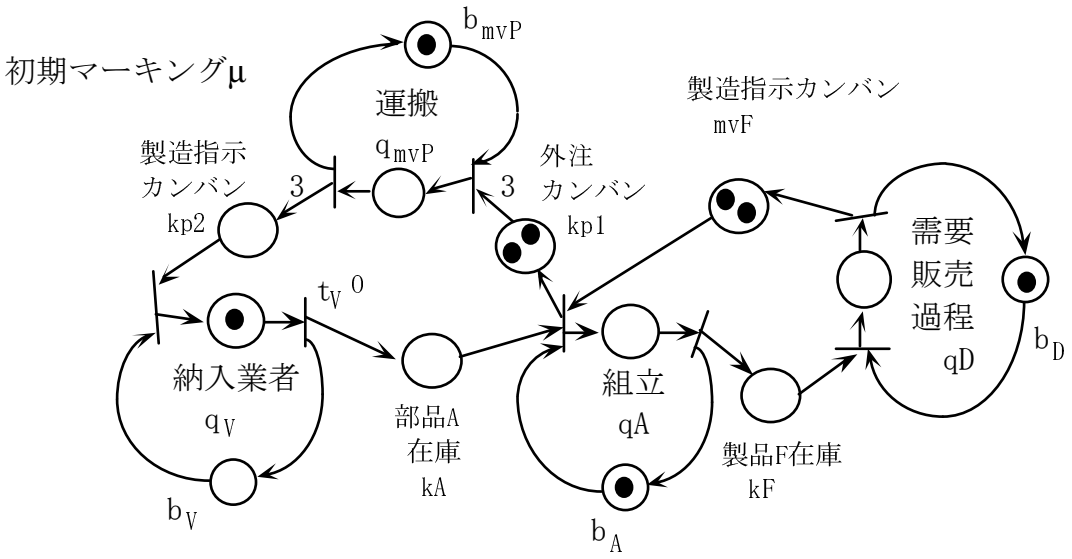


図 8-7　カンバンシステムの業務取引ペトリネットと初期マーキング

図 8-7 から始まって(PN1)から(PN5)の段階によってδによる状態遷移を計算する。

(PN1) $dueP(\mu,e)=\{t_V^o\}$である。なぜなら、他のどれも発火可能ではない。

(PN 2) q_Vが満期となってt_V^oが発火した後のシステムの状態は、δ_1の定義によって、$\delta_1(\mu,e)=(\mu^1,e^1)$とするとき、図 8-8 の$\mu^1$となる。

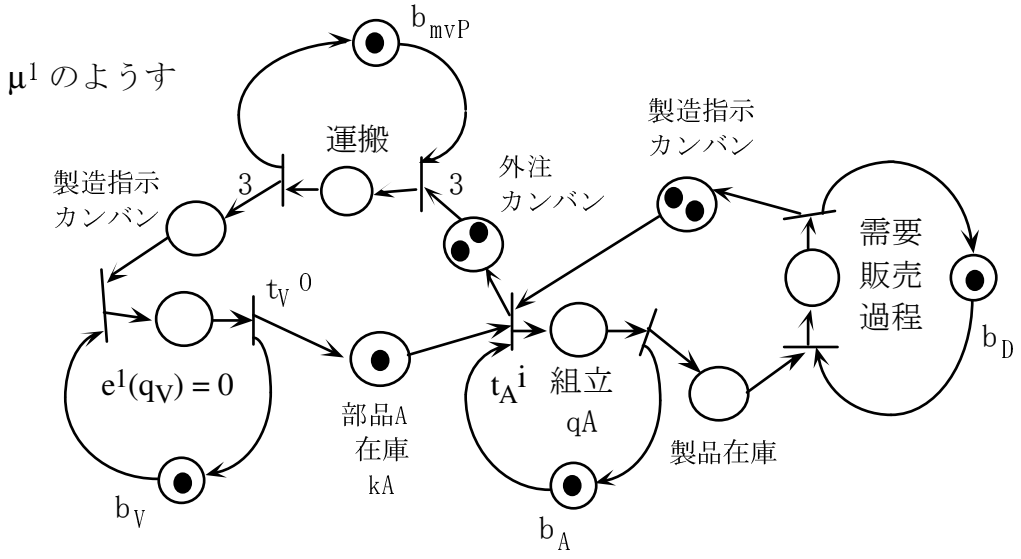


図 8-8　納入の終了

(PN 3) $st(\mu^1)=\{t_A^i\}$ である。つまり t_A^i が発火可能である。

(PN 4) t_A^i が発火すると最終的なシステムの状態が δ_2 によってわかる。

. $\delta_2(\mu^1,e^1)=(\mu^2,e^2)$ を求める。まず $g(\mu^1)$ は図 8-9 のようになる。

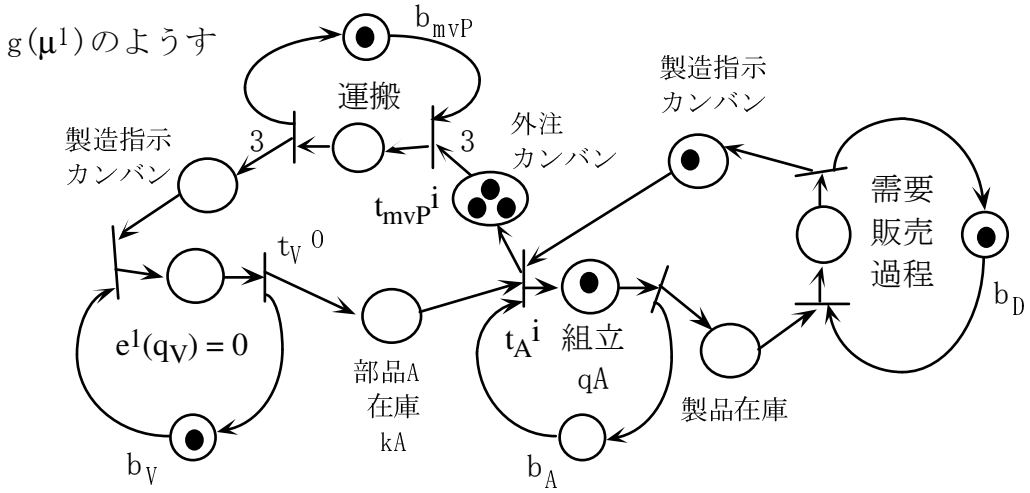


図 8-9　組み立ての開始

すると $st(g(\mu^1))=\{t_{mvP}^i\}$ である。だから $g(g(\mu^1))$ は次の図 8-10 のようになる。

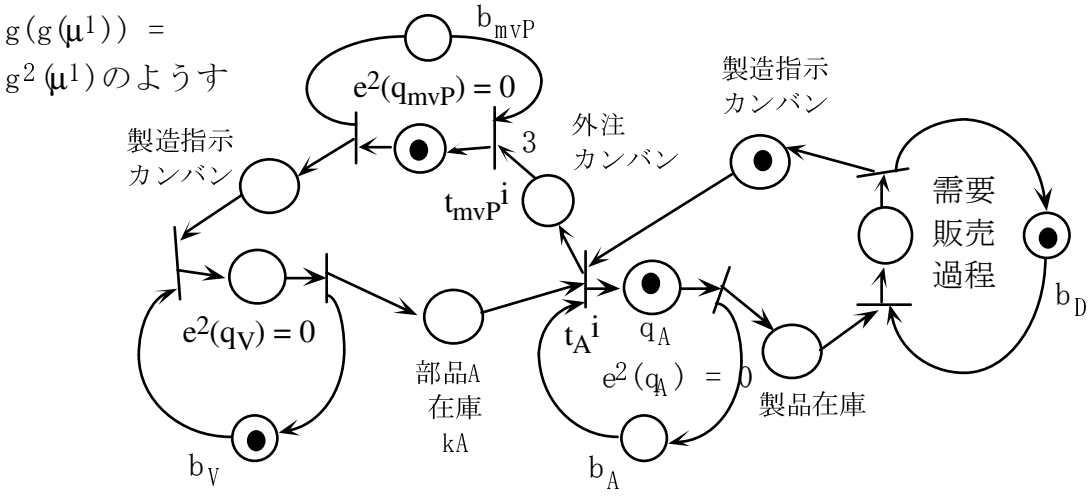


図 8-10　カンバンの運搬

すると、 $st(g(g(\mu^1)))=\phi$ だから $\mu^2=g(g(\mu^1))$ となる。

(PN５)　ここで $\overset{\infty}{\underset{p=0}{\cup}}\{st(g^p(\mu)\} = st(\mu^1) \cup st(g(\mu)) = \{t_A^i, t_{mvP}^i\}$ だから、

$e^1(q_V) = 0$ に加えて新たに $e^2(q_A) = e^2(q_{mvP}) = 0$ となる。
以上のようにして $\delta(\mu, e) = (\mu^2, e^2)$ が計算されることが示された。

８.４　業務取引ペトリネットをシミュレートする業務取引システム

業務取引ペトリネット(BTS-PN) の時間的振る舞いと同じ振る舞いをするシンプル業務取引システムを定める。

業務取引システムの特徴は次の２つである。

（１）ファイルシステムやファイルと業務活動の結合関係が、静的構造としてデータモデルと、アクティビティ・インタラクション・ダイアグラム（AID）やデータフローダイアグラム(DFD)で記述されること。

（２）離散事象システムの動的構造を持つこと。

第４章で述べた通り、AID の各活動は DEVS というひとつの離散事象システムである。それらがデータフローやデータストアで結合しているビジネスプロセス全体もまたひとつの DEVS である。

業務取引ペトリネットをシミュレートするシンプル業務取引システムを構成していく。ここでシミュレートとは、つぎの可換図を満たすことである。図の左側が業務取引ペトリネットの状態遷移関数であり、右側が業務取引システムのそれである。

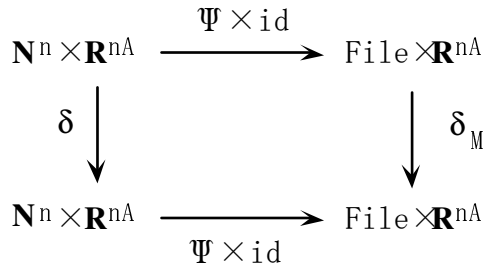


図 8-11　可換図

図 8-11 の関数 $\Psi : N^n \to File$ は、マーキングの状況をファイルシステムで自然に表現する対応であり後述する。

一般に、BTS を定めるにはその構成要素を定めればよい。それらは、(1) 活動、(2)

活動を開始するための条件、および(3) 活動開始による資源の使用状況と活動終了による資源の生産や解放を記録し共有する仕組みである。これらの構成要素は、次のような集合と関数を使って表現される。

(1) 活動の集合 $\{G_a = \langle S_a \times R^{\infty}, \delta_a, ta_a \rangle \mid a \in \underline{n_A}\}$

(2) ファイルによる活動の開始条件 $f_{FA} : File \to \mathrm{P}(\underline{n_A})$

(3) 活動によるファイル内容書き換え $f_{FileValue} : File \times \mathrm{P}(\underline{n_A}) \to File$

本節では BTS-PN に対応させてこれらの集合と関数について定義していく。

内部活動:各 busy プレースに対して内部活動を対応させる。つまり、$\underline{n_A}$ と $\{G_a \mid a \in \underline{n_A}\}$ は 1 対 1 に対応する。G_a はひとつの DEVS（離散事象システム）である。以下では、q_a という busy プレースに a という活動名が対応するものとする。

ファイルシステムを定める。$KS = P \setminus P_A$ とする。つまり活動プレース以外のプレースは各々一つのファイル（管理実体型）として表現される。シンプル業務務取引システムなので KS の要素である任意のファイル p についての属性は $\{\#FT(p), \text{個数}\}$ という２つだけである。$\#FT(p)$ はファイル p の主キーである。トークンが色付きでない場合(トークンが無差別な場合)には、この表にはいつも一行のデータしか存在せず、個数属性の値だけが変化することになる。これで $File$ が定まった。なお、記述を簡単にするために、マーキング μ に対するファイル内容関数を f_μ と表わし、個数属性に対する値を $f_\mu(p)$ と表わす。つまり $f_\mu(p) = \mu(p)$ である。

トークン配置をファイルシステムとして表わす対応である $\Psi : N^n \to File$ を $\Psi(\mu) = f_\mu$ と定める。Ψ は明らかに 1 対 1 対応、つまり単射かつ全射である。

f_{AK} も f_{KA} も、活動の集合 $\underline{n_A}$ から管理実体型のべき集合 $\mathrm{P}(KS)$ への関数であり、次のように定義する。たとえば、ある活動とプレースが図 8-12 のようであるとする。ただし q_a が busy プレース、b_a が idle プレースである。

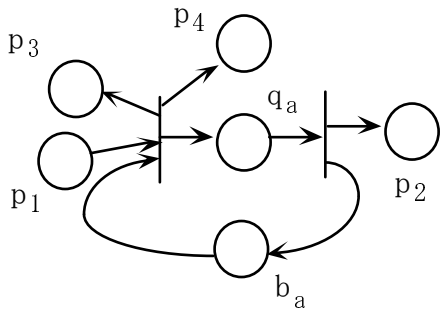


図 8-12.　プレースの例

このとき $b_a, p_1 \in f_{KA}(a)$、$b_a, p_2 \in f^F_{AK}(a)$、$p_3, p_4 \in f^S_{AK}(a)$ と定める。つまり p_3, p_4 だけが活動部分の入力トランジションからの出力プレースであるときには $f^S_{AK}(a) = \{p_3, p_4\}$ とかき、また、b_a, p_2 だけが活動部分の出力トランジションからの出力プレースであるときには $f^F_{AK}(a) = \{b_a, p_2\}$ とかく。$f_{AK}(a) = f^S_{AK}(a) \cup f^F_{AK}(a)$ と定める。また、活動部分 q_a の入力トランジションへの入力プレースを $f_{KA}(a)$ とかく。この絵の状況以外の場合も同様にして f_{KA} と f_{AK} を定義する。したがって、活動 a に関係する管理実体型（q_a の入出力のトランジションに直接接続するプレースで q_a 以外のもの）を $f_{KA}(a) \cup f_{AK}(a)$ と表すことができる。

シンプル業務取引システムの状態空間:

$S = \{(f_\mu, e_1, e_2, ..., e_{n_A}) \mid f_\mu \in File, (e_1, e_2, ..., e_{n_A}) \in R^{n_A}\}$ とする。

$S_a = File|_{f_{KA}(a) \cup f_{AK}(a)}$ とする。したがって、$f_\mu \in File$ を $f_{KA}(a) \cup f_{AK}(a)$ の管理実体型に制限したものを f^a_μ とかくとき、$S_a = \{f^a_\mu \mid f_\mu \in File\}$ である。

G_a の満期時間関数 $ta_a : S_a \to R^\infty$ を定める。

$ta_a(f^a_\mu) = h(f'_\mu, q_a)$; ただし f'_μ は $f'^a_\mu = f^a_\mu$ であるような任意のファイル内容関数である。h の定義からトークンがなければ満期時間は無限大となる。

ta_a の定義は well- defined である。

発火可能な活動を f_{FA} で定める。

$a \in f_{FA}(f_\mu) \Leftrightarrow$ 任意の p について、$p \in f_{KA}(a)$ ならば $f_\mu(p) \geq w_{pi}$ であること。

上の f_{FA} の条件は入力ファイルに、発火のために十分なトークンがあることを意味している。特に活動が busy でないことをも意味している。

当該の BTS の活動 a が f^a_μ で「開始可能」であるとは、

「あるトークン分布 f'_μ があって、$f^a_\mu = f'_\mu|_{f_{KA}(a)}$、かつ $a \in f_{FA}(f'_\mu)$」

であることと定める。

ひとつの DEVS である活動 a の状態遷移関数 δ_a を定めるために２つの関数 δ_{aO} と δ_{aI} を次のように定める。

関数 δ_{aO} は、活動終了によるトークンの移動に対応したファイルの値の変更を表わす。この処理を始めるには活動が busy 状態であることを示す $f^a_\mu(b_a) = 0$ が必要である。

$\delta_{aO} : File|_{f_{KA}(a) \cup f_{AK}(a)} \to File|_{f_{KA}(a) \cup f_{AK}(a)}$

$$\delta_{aO}(f^a_\mu)(p) = \begin{cases} f^a_\mu(p) + w_{po}, \text{if } p \in f^F_{AK}(a) \wedge f^a_\mu(b_a) = 0; \\ f^a_\mu(p), \text{otherwise.} \end{cases}$$

関数δ_{aI}は、現在のファイルシステムが表わす状況下で開始可能な活動を開始する。

$\delta_{aI} : File|_{f_{KA}(a)\cup f_{AK}(a)} \to File|_{f_{KA}(a)\cup f_{AK}(a)}$

. $\delta_{aI}(f_\mu^a)(p) =$

$f_\mu^a(p) - w_{pi}$, $p \in f_{KA}(a)$かつaがf_μ^aで開始可能のとき

$f_\mu^a(p) + w_{pi}$, $p \in f_{AK}^S(a)$かつaがf_μ^aで開始可能のとき

$f_\mu^a(p)$, その他のとき。

$\delta_a = \delta_{aI} \circ \delta_{aO}$と定める。つまり$\delta_a(f_\mu^a) = \delta_{aI}(\delta_{aO}(f_\mu^a))$である。まず終了の処理をしてから開始の処理をする。

ファイル更新関数$f_{FileValue} : File \times \mathrm{P}(\underline{n_A}) \to File$を定める。

$$f_{FileValue}(f_\mu,\{a\}) = \begin{cases} \delta_a(f_\mu^a)(p), \text{if } p \in f_{AK}(a) \cup f_{KA}(a); \\ f_\mu(p), \text{otherwise.} \end{cases}$$

任意の$\{a_{i_1}, a_{i_2}, ..., a_{i_r}\}$に対して、

$f_{FileValue}(f_\mu,\{a_{i_1}, a_{i_2}, ..., a_{i_r}\}) = f_{FileValue}(\cdots(f_{FileValue}(f_{FileValue}(f_\mu,\{a_{i_1}\}),\{a_{i_2}\}),\cdots),\{a_{i_r}\})$.

シンプル業務取引システムの状態遷移関数δ_Mは、上の諸関数を用いて、次のように構成された。まず満期時間関数taは以下の通りである。

$$ta : File \times R \to R^\infty, \quad ta(f_\mu, e_1, e_2, ..., e_{n_A}) = \min_{a \in \underline{n_A}}\{ta_a(f_\mu^a) - e_a\} .$$

関数dueが次で定義される：

$due(f_\mu, e_1, e_2, ..., e_{n_A}) = \{a_{i_1}, a_{i_2}, ..., a_{i_m}\} \Leftrightarrow$

任意の$k, 1 \le k \le m,$について $ta(f_\mu, e_1, e_2, ..., e_{n_A}) = ta_{a_{ik}}(f_\mu^{a_{ik}}) - e_{a_{ik}}$

と定める。次に、２つの関数 f_{FA} と $f_{FileValue}$から次のように定義され、f_μから開始可能な一連の活動を開始させる仕組みを記述している関数$f_F(f_\mu)$を定める：

$$f_F(f_\mu) = \begin{cases} f_\mu, \text{if } f_{FA}(f_\mu) = \phi \\ f_F(f_{FileValue}(f_\mu, f_{FA}(f_\mu))), \text{if } f_{FA}(f_\mu) \neq \phi \end{cases}$$

次に、f_μから開始した一連の活動のファイル内容への変更結果を示している関数である$f_{Dispatch}$は$f_{Dispatch}(f_\mu) = f_{FileValue}(f_\mu, f_{FA}(f_\mu))$ で定義される。

これから、各々の活動 i の経過時間を

$$e_i' = \begin{cases} 0, \text{if } i \in due(s) \cup \bigcup_{k=0}^{\infty} f_{FA}[{f_{Dispatch}}^k(f_{FileValue}(f_\mu, due(s)))] \\ e_i - ta(s), \text{otherwise.} \end{cases}$$

と定める。ただし、$(f_\mu, e_1, e_2, ..., e_{n_A})$を$s$と表した。また、${f_{Dispatch}}^k(f_\mu)$とは$f_\mu$に関数 $f_{Dispatch}$

を逐次的に k 回適用した結果を表している。

これを用いて、BTS の状態遷移関数 δ_M は次で定める。

$$\delta_M(f_\mu,e_1,e_2,...,e_{n_A}) = (f_F[f_{FileValue}(f_\mu,due(f_\mu,e_1,e_2,...,e_{n_A}))],e_1',e_2',...,e_{n_A}')$$

以上によって、BTS-PN に対応する BTS が定まった。

前節の図 4 のカンバンシステムのペトリネットモデル BTS-PN をシミュレートできるシンプル業務取引システムとその状態遷移を(BTS1)～(BTS5)で調べる。

(BTS1) まず、図 8-7 の初期マーキングに対応する初期状態 f_μ をもつ BTS は図 8-13 になる。図で、たとえば、$k_A(F)$ は k_A が $f_{AK}^F(V)$ の要素であることを示す。また、$kp1(S)$ は $kp1$ が $f_{AK}^S(A)$ の要素であることを示している。

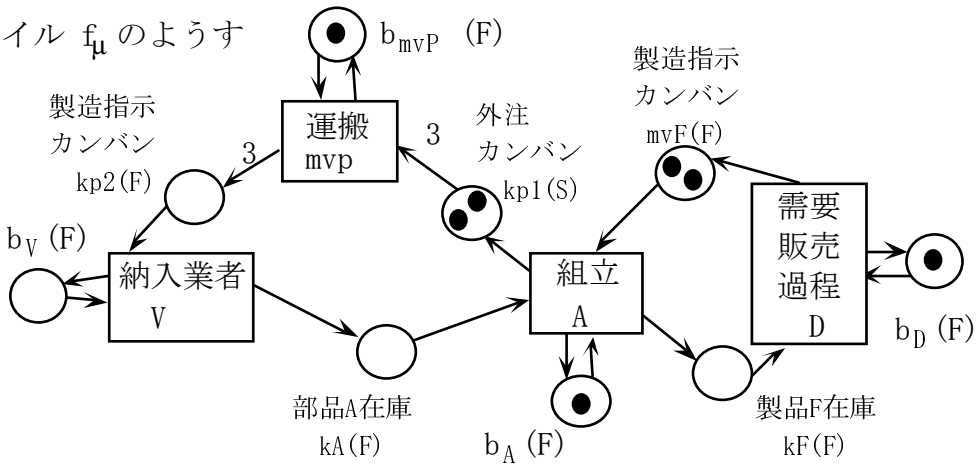


図 8-13　カンバンシステムの業務取引システム

(BTS 2) 図 8-7 の初期条件によって $due(f_\mu,e)=\{V\}$。また $f_{FA}(f_\mu)=\phi$(空)である。

(BTS 3) 図 8-8 の μ' に対して、ファイル内容関数 $f_{\mu'} \in File$ は $f_{\mu'}(kA)=1, f_{\mu'}(b_V)=1$, kA と b_V 以外の p に対しては $f_{\mu'}(p)=f_\mu(p)$ となる（図 8-14 参照）。

このとき、$f_{FileValue}(f_\mu,due(f_\mu,e))=f_{FileValue}(f_\mu,\{V\})=f_{\mu'}$ である。

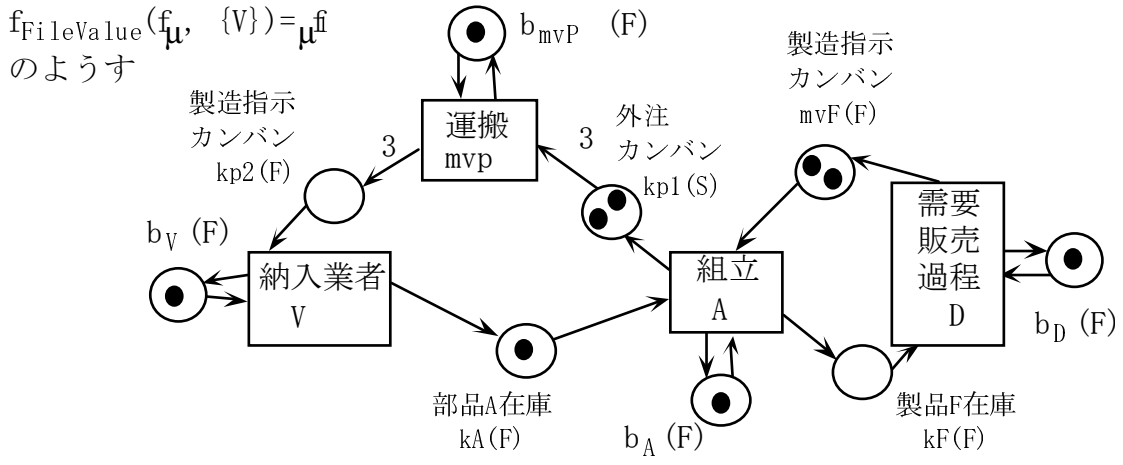


図 8-14　納入の終了

(BTS 4) 次に $f_F[f_{FileValue}(f_\mu, due(f_\mu, e))] = f_F(f_{\mu^1})$ を計算する。 $f_{FA}(f_{\mu^1}) = \{A\}$ だから、f_F の定義によって

$f_F(f_{\mu^1}) = f_F(f_{FileValue}(f_{\mu^1}, \{A\}))$ である。$f_F(f_{\mu^1})$ は図 8-15 のとおりとなる。明らかに、$f_F(f_{\mu^1})$ は図 8-9 に対応するファイル内容関数であるので、$f_F(f_{\mu^1}) = f_{g(\mu^1)}$ である。活動 A を開始した直後のようすが $f_{g(\mu^1)}$ である。

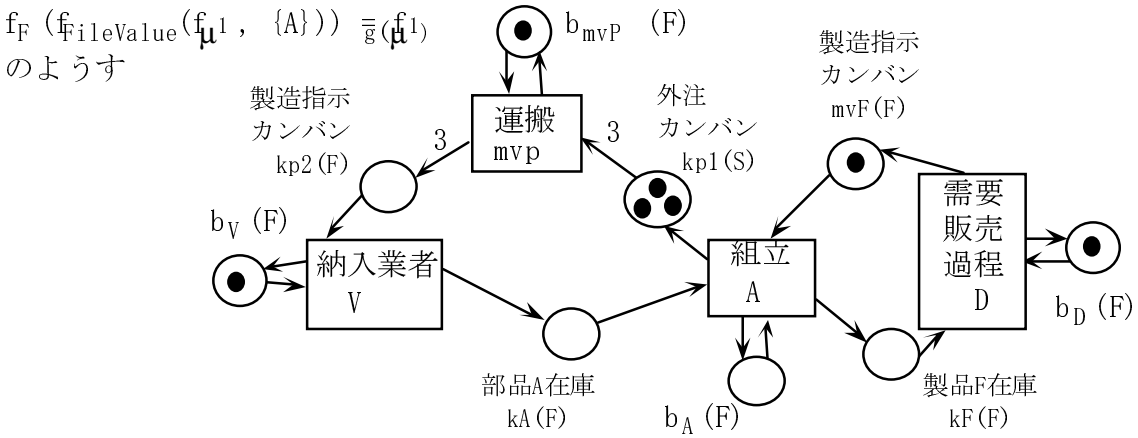


図 8-15 組み立ての開始

この段階において $f_{FA}(f_{g(\mu^1)}) = \{mvP\}$ だから活動 mvP が開始される。

$f_F(f_{g(\mu^1)}) = f_F(f_{g(\mu^1)}, \{mvP\})$ を求めると図 8-16 のようになる。図 8-10 との対応から明

らかに、$f_F(f_{g(\mu^1)}) = f_{g(g(\mu^1))} = f_{g^2(\mu^1)}$である。

$f_F(f_{g(\mu^1)}) = f(f_{g(\mu^1)}, \{mvP\})$
$= f_{g^2(\mu^1)}$ のようす

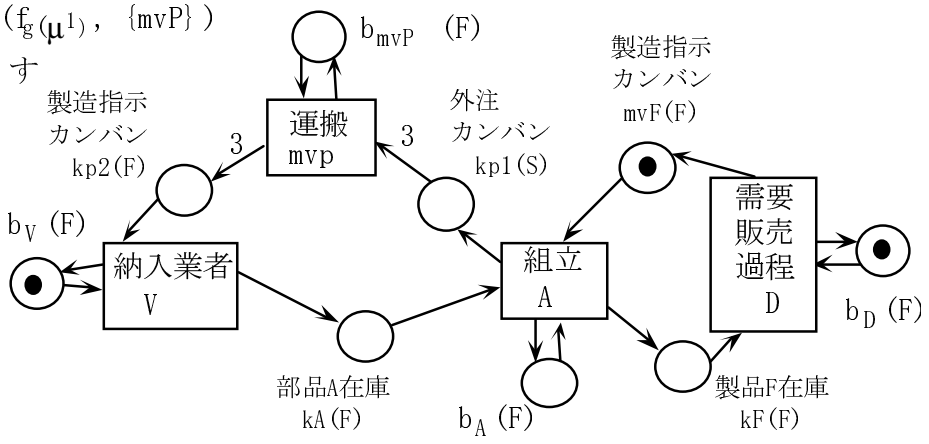


図 8-16 カンバンの運搬

もはや $f_{g^2(\mu^1)}$ の状況で開始できる活動はないので $f_{FA}(f_{g^2(\mu^1)}) = \phi$ である。したがって、$f_F[f_{FileValue}(f_\mu, due(f_\mu, e))] = f_{g^2(\mu^1)}$ となった。

(BTS 5) $(f_\mu, e) = s$ と書くとき、$\{A, mvP\} = due(s) \cup \bigcup_{k=0}^{\infty} f_{FA}[f_{Dispatch}{}^k(f_{FileValue}(f_\mu, due(s)))]$ なので、活動 V に加えて A と mvP の経過時間が 0 となる。

以上によって業務取引システムと業務取引ペトリネットの動作が同等であることを、図 8-7 のビジネスプロセスの BTS-PN を例として動的なふるまいを図示することによって示した。図 8-11 の可換図が成立することの一般的な可換性の証明は本章の付録２にある。業務取引システムの例は定型業務のプロセスであるから、生産プロセス以外にも「仕事はイベントで動いていく」(飯島・佐藤編『システム知の探究 2』, 日科技連出版社, 1997）で示されているようにいろいろなものがあるが、以下で述べる方法でどの定型業務も全く同じ方法で扱うことができる。

8.5 業務取引ペトリネットの動的性質

業務取引ペトリネットとシンプル業務取引システムの状態遷移関数の同型性

（図 8-11）が成立することによって、それらの動的性質は同様のものとなる。つまり、シンプル業務取引システムの動的性質の分析のために業務取引ペトリネットを用いることができる。

ペトリネットのサーキットとは、任意のトランジションから始めてアークの向きに順にいくつかのプレースやトランジションを経由して、元のトランジションに返るようなアークの列である。（これはシンプル業務取引システムのサーキットの定義と整合した定義である。）また、同じトランジションはただ一組しかないような、いわばサーキットの開始点と終了点のトランジション以外はすべて異なるトランジションから構成されているサーキットを基本サーキットと呼ぶ。当然、ひとつのペトリネットには有限の基本サーキットと無限のサーキットがあることになる。

業務取引ペトリネットはマークグラフと呼ばれる種類のペトリネットであり、次の「サーキット内のトークン不変」の性質が成立する。マークグラフとは、すべてのプレースが重み 1 のただひとつの入力アークと出力アークをもつペトリネットである。

命題 1（村田、ペトリネットの解析と応用、近代科学社、1992）

任意のマークグラフと任意の初期トークン配置において、任意のサーキット上のトークン数は、どのような発火が起きても不変である。

（証明）

マークグラフでは、すべてのプレースは、重み 1 のただひとつの入力アークと出力アークをもつ。（トランジションはいろいろな入出力アークを持っている。）トランジションの発火は、各入力アーク（トランジション）からひとつのトークンを取り去り、各出力アーク（プレース）へひとつのトークンを加えることになる。

任意のサーキット C を取る。任意のトランジションを取ると、C 上にあるかないかのどちらかである。トランジションが C 上にあれば、そのトランジションの入力アークと出力アークのうちでそれぞれただひとつだけが C に属する。だから、C 上でのトークン総数は不変。また、あるトランジションがそのループ上になければ、そのトランジションに接続するどんなアークもそのループに属さない。よってトークン数は、問題のループのトランジションの発火によっては（発火

しないので）変化しない。　　Q.E.D.

命題１より、業務取引ペトリネットにおいても任意のサーキット上のトークン数は、任意の発火に対して不変である。

離散事象システムの重要な性質であるデッドロックや活性の概念を、シンプル業務取引システムと同様に考えることができる。デッドロックとは、業務取引システムでいえば（資源や情報の不足や矛盾によって）どれかの活動が開始できなくなることである。また、業務取引ペトリネットではトークンが不足してどれかの活動への入力トランジションが発火できなくなることである。活性であるとは決してデッドロックが起こらないことであり、業務取引ペトリネットが初期トークン配置について活性であることは、任意の時点でその将来にどのトランジションも必ずいつか発火することである。したがって、活性であればイベントがいつまでも発生するので状態遷移表が無限に続くことになる。

補題１（村田，ペトリネットの解析と応用、近代科学社、1992）

強連結な業務取引ペトリネットと任意の初期トークン配置を考える。それが活性であることと、 その初期トークン配置において業務取引ペトリネット内のすべてのサーキット上に１個以上のトークンがあることは等価である。

（証明）

１個以上のトークンを持たないサーキットがあるとする。命題１より、そのサーキットの busy プレースにはトークンが入ることはない。よってその busy プレースの出力トランジションは決して発火しないので、活性でない。したがって、活性であれば、すべてのサーキット上に１個以上のトークンがある。

活性でないとすると、どれかの活動への入力トランジションが発火できない。ということは、その入力トランジションへの入力となっている結合プレースにトークンを送る可能性がある、ある busy プレースの出力トランジションが発火しないということである。こうしてトークンの入っていないプレースをたどると、強連結であることから、やがてトークンの入っていないサーキットが得られる。

Q.E.D.

業務取引ペトリネットに周期があるということは、状態遷移表の１周期分を隔

てた時刻における２つの行がまったく同じであるということである。シンプル業務取引システムについての周期的動作の存在によって、業務取引ペトリネットとの同型性から次の２つの事実（命題 2、3）が成立し、周期が存在する。

命題２

強連結で活性な業務取引ペトリネットの状態遷移表の行の中で取りうる busy プレースの残り時間の値は、有限通りの値しかない。

命題３

強連結で活性な業務取引ペトリネットはかならず周期を持つ。

命題２の結果は、各 busy プレースでのトークン保持時間が、トークンに依らずに定まった値を各々持っているということ（時間不変性）に関連している。ただ、時間不変性を仮定したからといって、命題２の事実は自明ではない。業務取引システムから作られる業務取引ペトリネットのトークン保持時間は、整数ではなく実数である。

以下の命題群は、ビジネスプロセスの周期的なふるまいについての分析と特徴づけである。

命題４

活性な業務取引ペトリネットが周期を持てば、１周期のあいだにすべてのトランジションが発火する。

（証明）　１周期のあいだに、どれかのプレースが満期を迎えないとする。すると、その出力トランジションは永遠に発火しない。これは活性であることに反する。

Q.E.D.

命題５

強連結で活性な業務取引ペトリネットのすべてのトランジションは、１周期のあいだに同じ回数だけ発火する。

（証明）

１周期のあいだに、どれかのトランジション t が n 回発火したとする。すると、

tの出力プレースにn個のトークンが送り出される。tが属するサーキットにおける全トークン数は不変だから、もし、サークル内のすべてのトランジションがn回だけ発火しないと、発火数の異なるトランジション間のプレースにトークンが澱んでしまい、周期的ではありえなくなる。したがって、tが属するサークル内のすべてのトランジションはn回だけ発火する。

このサーキットに含まれないトランジションt'は、強連結であることからtからのパスがあり、そのパスはいくつかのサーキットに属している。これらのサーキット上のどのトランジションも、１周期のあいだにtの場合と同じ理由でtと同回数だけ発火する。

以上から、すべてのトランジションは１周期のあいだに同じ回数だけ発火する。

Q.E.D.

命題６

強連結で活性な業務取引ペトリネットのすべての重みが１とする。このとき、

$$\lambda \le L$$

である。ただしLは周期であり、λは最大サイクル平均値である。業務取引ペトリネットの各サーキットごとの１トークンあたりの保持時間をサイクル平均値と呼ぶが、λは全サーキットの中で最大のサイクル平均値であり、以下で定義される量である：

基本サーキットが全部で$C_1,...,C_n$だけあるとする。サーキットC_iの上のトークン総数がk_iとする。busy プレースq_iの保持時間をh_iとし、サーキットC_k上にあるすべてのq_iの保持時間の和を$\sum_{q_i \in C_k} h_i$と表す。このとき

$$\lambda = \max\left\{ \frac{\sum_{q_i \in C_1} h_i}{k_1}, ..., \frac{\sum_{q_i \in C_n} h_i}{k_n} \right\}$$

（証明）

Mをトークン総数より十分に大きな整数とする。１周期のあいだにk回だけすべてのトランジションが発火するものとする。M回の周期を繰り返すと、各サーキットをMk個のトークンが移動したものと考えてよい。トークンがMk回サーキットをめぐるためにかかった時間LMは、トランジションがブロックされる可能性があるので、サーキットのプレースの保持時間の総和以上

になる。つまり任意のサーキットCについて、

$$kM\left(\sum_{i\in C} h_i\right) \le LM$$

が成立する。

したがって、$\sum_{i\in C} h_i \le L/k$より、$\lambda \le L/k$ を得る。kは１以上だから$\lambda \le L$。

Q.E.D.

この命題６は最大サイクル平均値λは周期の下限値を与えている。最大サイクル平均値を持つサーキットを、クリティカルサーキットと呼ぶ。なお、命題 6 の業務取引ペトリネットにおいてトークンの数が充分あって、「あるプレースについてはどうサーキットをとってもトークンを２つ以上含む」場合には状態遷移行列の固有値が業務ペトリネットの周期より厳密に小さい例が存在する。この意味で、命題６は不等式でしか成立しない。

業務取引システムから定義される業務取引ペトリネットについては、以下の命題の条件が成立すれば等号が成立する。

命題７

強連結で活性な業務取引ペトリネットのすべての重みが１とする。すべてのプレースについてトークンを一つだけ含むサーキットがとれるとする。このとき、$k\lambda = L$である。ただしLは周期であり、kは１周期のあいだにひとつの任意のトランジションが発火する回数である。また、λは最大サイクル平均値である。

（証明）　任意の２つのサーキットC, C'を考え、C'の方がサーキット内の活動プレースの保持時間の総和が小さいとする。このとき、C'の方が、１周期の間に発火をブロックされている時間の総和が長いことに注意する。なぜなら、どのサーキットにもトークンが１つだけ存在し、かつ１周期の間にどちらのサーキットのトランジションもすべて同じ回数だけ発火するである。

さて、１周期の状態遷移においてクリティカルサーキットC_1のどれかのトランジションtがブロックされたと仮定する。すると、tへの結合プレースへの入力トランジションを出力に持つ活動プレースがあり、それはC_1とは異なるサーキットC_2に属している。このとき、初めに述べた注意によってC_1の方が保持時間総和が大きいので、C_2の方がブロックされている時間が長い。つまり、C_2の

どれかのトランジションがC_2以外のサーキットC_3の活動プレースによってブロックされている。ここで、C_3はC_1ではない。もしそうなら、それ以降の状態遷移が進まず、活性であるという仮定に反するからである。C_1とC_3の保持時間総和を比較すると、C_1とC_2の場合と同様にして、C_3のどれかのトランジションがブロックされている。しかも、活性であることからC_1、C_2以外のサーキットによってブロックされている。こうして、クリティカルサーキットのトランジション発火がブロックさることは、無限個のサーキットがあることを意味する。これは矛盾である。

したがって、１周期の状態遷移においてクリティカルサーキットのどのトランジションもブロックされない。つまり、クリティカルサーキットの活動部分は満期が来ると順に発火していく。したがって、周期Lはクリティカルサーキットの保持時間総和と一致する。

Q.E.D.

図 8-5 の例は命題７の条件を満たし、$L=\lambda=31$となる。実際、表１の状態変化を続けていくと、周期は191時点以降で発生する。

（表１の続き）

time	As	qA	Aw	qS	pA	Bs	qB	q1	pB1	pB2	Bw	q2	F
191	1	0	0	0	0	0	0	1(26)	0	0	1	1(26)	7
191	0	0	0	1(25)	0	0	0	1(26)	0	0	1	1(26)	7
216	0	0	0	0	1	0	0	1(1)	0	0	1	1(1)	7
217	0	0	0	0	1	0	0	0	1	0	1	1(0)	7
217	0	0	0	0	1	0	0	0	1	0	1	0	8
217	0	0	0	0	1	1	1(4)	0	0	0	0	0	8
217	0	0	1	0	0	0	1(4)	1(31)	0	0	0	0	8
217	0	1(5)	0	0	0	0	1(4)	1(31)	0	0	0	0	8
221	0	1(1)	0	0	0	0	0	1(27)	0	1	0	0	8
221	0	1(1)	0	0	0	0	0	1(27)	0	0	1	1(27)	8
222	1	0	0	0	0	0	0	1(26)	0	0	1	1(26)	8

以上から、ビジネスプロセスの時間生産性を高めるための示唆が得られる。(1) もっとも遅いサイクル平均値を持つような定型業務の連携しているサーキットを見つけ、その時間量生産性を上げる（つまり、単位時間あたりの処理量を

上げる）。
(2) 問題となっているサーキットを組み直すことで分業の仕組みを変更することでサイクル平均値を下げて、クリティカルサーキットではないような分業体制に作り直す必要がある。
(3) 手あたりしだいにいろいろな作業を速くしても、コストが膨らむばかりで、たとえば短納期に結び付くとは限らない。
(4) 周期が異なるビジネスプロセスを統合しても、一番遅い周期の速さしか得られない可能性がある。

8.6 max-plus 方程式を用いる業務取引ペトリネットの分析

フィードバックによって制御システムを組み込んで強連結となったビジネスプロセスの最大サイクル平均値は、 Max-plus 代数と呼ばれる一種の「線形代数」の行列の固有値を計算することで求めることができる。

8.6.1 Max-Plus 代数

通常の実数を要素とするベクトルや行列の演算は加算と乗算を持つ。BTS-PN についての時間特性計算のためには、Baccelli, F.L., Cohen, G., Olsder, G.J., and Quadrat, JP,（ “Synchronization and Linearity -- An algebra for discrete event systems,” John Wiley，1992） に従い、加算を max にし乗算を＋にした、次のような行列計算を考える。

$$\begin{pmatrix}3 & 5\\ 2 & 8\end{pmatrix}\begin{pmatrix}7\\ 4\end{pmatrix}=\begin{pmatrix}\max(3+7,5+4)\\ \max(2+7,8+4)\end{pmatrix}=\begin{pmatrix}10\\ 12\end{pmatrix}$$

この乗算を⊗とかく。単位元は（実は＋だから）0である。これを e で表わす。すなわち $e=0$。加算を⊕とかくと単位元 ε は（実は⊕は max だから) $\varepsilon=-\infty$。ただし⊕の逆演算はない。行列形式の乗算⊗と加算⊕について、次のように、通常の乗算と加算と同じ形式の行列計算の規則が成立する。

$$\begin{pmatrix}3 & 5\\ 2 & 8\end{pmatrix}\otimes\begin{pmatrix}7\\ 4\end{pmatrix}=\begin{pmatrix}3\otimes 7\oplus 5\otimes 4\\ 2\otimes 7\oplus 8\otimes 4\end{pmatrix}=\begin{pmatrix}10\\ 12\end{pmatrix}$$

このように定まる一種の線形代数は max-plus 代数と呼ばれている（Baccelli et.al., 1992）。数学的には max-plus 代数は体ではなく半体(semi field）と呼ばれ、次の３条件を満たすものである。

(1) $\langle \otimes, e \rangle$は可換群。

(2) $\langle \oplus, \varepsilon \rangle$は可換モノイド。

(3) ⊗は⊕に対して分配的： $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

加算⊕が max 演算なので逆元がなく加算の逆演算もできない。しかし、通常の行列計算と同じ計算手順で計算できる。したがって、固有値や固有ベクトルも同じ形式で定義される。本書によって max-plus は、業務取引ペトリネットを通じてビジネスプロセスの特性計算に初めて用いられるのである。

8.6.2. 業務取引ペトリネットの Max-Plus 方程式

平衡状態にある時に図 8-5 の BTS-PN から Max-Plus 代数の乗算⊗と加算⊕を使って以下のように行列表現を見い出すことができる。ただしこれ以降は、乗算と加算の記号は、計算規則が同じなので通常の実数の場合と同じように、加算を＋で表わし、乗算記号は省略する。$X_i(t)$ を図 8-5 のトランジション t_i が t 回目に発火する時刻とする。つまり、$X_i(t)$ の t は時刻ではなく、トランジション t_i 発火の生起回数を数えるカウンター(生起回数を数える）変数である。たとえば、図 8-5 において t_1,t_2 各々のトランジションの $t+1$ 回目の生起時刻を次のように表わせる。

$$X_1(t+1) = 0X_2(t+1) + 0X_3(t+1)$$

$$X_2(t+1) = 25X_1(t)$$

X_1 の式の意味：t_1 が $t+1$ 回目に発火する時刻は、t_2 が $t+1$ 回目に発火する時刻に b_S の保持時間 0 を加えた値と、t_3 が $t+1$ 回目に発火する時刻に A_S の保持時間 0 を加えた値のうちの大きい方の時刻である。

X_2 の式の意味：t_2 が $t+1$ 回目に発火する時刻は t_1 が t 回目に発火する時刻に q_S の保持時間 25 を加えた時刻になるということである。

$X_i(t)$ を第 i 成分とするベクトルを $X(t)$ とし、$X_i(t+1)$ を第 i 成分とするベクトルを $X(t+1)$ と表すと次を得る。

$$X(t+1) = A_0X(t+1) + A_1X(t)$$

ただし、A_0とA_1は各々以下のとおりである。

$$A_0=\begin{pmatrix}\varepsilon&0&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&5&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&0&\varepsilon&\varepsilon&\varepsilon&0&0&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&0&0&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&0\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\end{pmatrix},A_1=\begin{pmatrix}\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ 25&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&31&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&4&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\ \varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&27&\varepsilon\end{pmatrix}$$

Eを Max-Plus 代数の単位行列とする。つまりEは対角要素がすべてeで、それ以外はεである行列とする。上式で$X(t+1)=A_0X(t+1)+A_1X(t)$の右辺の$X(t+1)$に$A_0X(t+1)+A_1X(t)$を逐次代入して行くと、n回の代入実行で次を得る。

$$X(t+1)=A_0(A_0X(t+1)+A_1X(t))+A_1X(t)$$
$$=A_0{}^2X(t+1)+(A_0+E)A_1X(t)$$
$$=\ldots=A_0{}^{n+1}X(t+1)+(A_0{}^n+A_0{}^{n-1}+\cdots+A_0+E)A_1X(t)$$

よってA_0がべき零で$A_0{}^k=0$（0はすべての要素がεであるゼロ行列）となるなら、

$$X(t+1)=(A_0{}^{k-1}+A_0{}^{k-2}+\cdots+A_0+E)A_1X(t)$$

となる。

現在の例ではA_0はべき零であり、$A_0{}^7$以降はゼロ行列となる。よって$A_0{}^*=A_0{}^6+A_0{}^5+\cdots+E, A=A_0{}^*A_1$とおくと、

$$X(t+1)=A_0{}^*A_1X(t)=AX(t)$$

と表すことができる。したがって、任意の生起回数tについて

$$X(t)=A^tX(0)$$

を得る。図 8-5 の場合には、Aは次のようになる。

$$A=\begin{pmatrix} 30 & \varepsilon & 5 & \varepsilon & 36 & \varepsilon & 9 & \varepsilon & 32 & \varepsilon \\ 25 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 30 & \varepsilon & 5 & \varepsilon & 36 & \varepsilon & 9 & \varepsilon & 32 & \varepsilon \\ 25 & \varepsilon & 0 & \varepsilon & 31 & \varepsilon & 4 & \varepsilon & 27 & \varepsilon \\ 25 & \varepsilon & \varepsilon & \varepsilon & 31 & \varepsilon & 4 & \varepsilon & 27 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 31 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 31 & \varepsilon & 4 & \varepsilon & 27 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 4 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 4 & \varepsilon & 27 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 27 & \varepsilon \end{pmatrix}$$

行列 Aを業務取引ペトリネットの状態遷移行列と呼ぶ。σをAの固有値とするとき次が成立する。

命題８（Baccelli et.al., 1992）

(1) 強連結な業務取引ペトリネットの状態遷移行列は常に存在する。つまり、A_0がべき零である。

(2) 連結な業務取引ペトリネットの状態遷移行列は、ただ一つの固有値をもつ。さらに、トランジションの総数をyとするとき、$\sigma=\max_{j=1,2,...,y}\left\{\frac{A^j{}_{11}}{j},\frac{A^j{}_{22}}{j},...,\frac{A^j{}_{yy}}{j}\right\}$である。ただし、$A^j{}_{kk}$は行列$A$を$j$乗した行列の$(k,k)$対角要素である。

例とした行列Aでは $\max\{31/1, 31/1, ..., 62/2, ..., 310/10\}=31$となる。

命題９（Baccelli et.al., 1992）

状態遷移行列の固有値は、もともとの業務取引ペトリネットの最大サイクル平均値と一致する。

状態遷移行列の固有値と周期の関係は次のようにまとめられる。

命題 10

強連結で活性であり、アークのすべての重みが１である業務取引ペトリネットをとる。周期をL，状態遷移行列の固有値をλとするとき、$\lambda\le L$である。

さらに、すべてのプレースについてトークンを一つだけ含むサーキットがとれるとき、$k\lambda = L$である。ただし、kは１周期のあいだにひとつの任意のトランジションが発火する回数である。

（証明）

命題６、７、８、９より明らかに成立する。

Q.E.D.

業務取引システムや、より広く、離散事象システムの動作を記述する方程式はいくつかの提案はあるものの、まだ知られていないと言ってよい。連続変化する現象に対しては微分方程式がその因果的な動作を記述することにおいて圧倒的なパワーを持つが、そのような方程式がないのである。連続変化する対象の微分方程式を基にして、機械工学や電気・電子工学、熱力学や化学反応方程式など、長年に渡って驚異的な展開を行って来たのである。ビジネスプロセスの工学的な設計においても、微分方程式に匹敵するビジネスプロセス方程式や離散事象方程式を発見できれば、その利用価値は相当に大きい。連続システムの場合には、重み関数や畳み込み積分で表現されるような微小入力に対する逐次的蓄積を入力全体についてまとめた総体（積分）としての出力を持つ。一方、離散事象的変化は、離散事象セグメントに対する直接的出力が並んでいることで表現される（システムコアにおける出力表現性と呼ばれる）。これは次章で詳しく示すように、離散事象システムと連続システムとの大きな違いである。つまり、離散事象システムと連続システムは質的に異なっているため、微分方程式で離散事象システムやビジネスプロセスを方程式表現することはできない。別の方法が必要なのである。次節でビジネスプロセス方程式の有望な方式として max-plus 方程式の使用方法を述べる。

8.7 ビジネスプロセスの方程式

本節で max-plus 方程式の使い道の一端を示す。

次の命題のようなシンプルな業務取引ペトリネットを考える。この場合には、状態遷移表の動作と、max-plus 行列方程式による動作履歴が完全に一致する。このときの max-plus 方程式を業務取引ペトリネットの状態方程式と呼ぶ。動的性質を記述する状態遷移表を逐次調べることなく、状態方程式を使って、たとえば、リトルの定理で表される、仕掛かり在庫と業務全体のリードタイムの分析

といったことを、max-plus 方程式を解析しその行列的性質を調べたり設計することにつながる。本書においてただちにそうした工学理論を展開できるわけではないが、将来の応用の扉を開くものと考えられる。max-plus 方程式が動的性質を調べるためのビジネスプロセス方程式として利用できるのである。

現在のところ、max-plus 方程式は限られたモデルに対してのみうまく状態遷移表を表現する。この意味でまだまだ限定的ではあるが、ビジネスプロセスの動的特性設計の工学的展開をはかる上での有望な考え方といえる。

命題 11

強連結で活性であり、平衡状態において任意の基本サーキットにただひとつのトークンが含まれるような業務取引ペトリネットをとる。このとき、1 周期の間にすべてのトランジションがただ 1 回だけ発火する。

（証明）

命題の条件を満たす任意の業務取引ペトリネットを考える。

周期を L、最大サイクル平均を λ とするとき、命題 7 により、ある整数 k があって $L = k\lambda$ である。このとき、図 8-17 のような状態遷移表となっている。

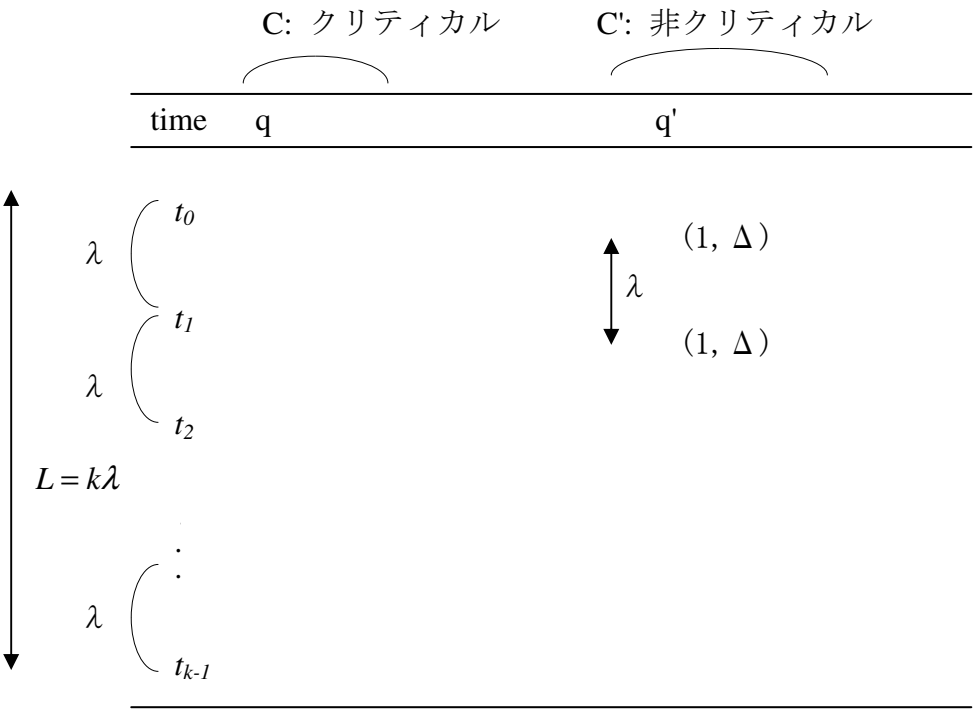


図 8-17 1 周期の状態遷移のようす

各々のトランジションは、１周期の間に同回数の発火をする。また、クリティカル・サイクルC上の全てのトランジションは、命題７の証明で示したようにブロックされることなく、つまり、C上にある入力プレースの保持時間が経過するとただちに発火する。$L=k\lambda$だから、C上の全てのトランジションは１周期の間にk回発火しており、他のすべてのトランジションもk回発火している。周期状態において、C上にない任意のトランジションtが時刻t_0に発火したとする。すると、幅がλの時間区間$[t_0,t_1)$内にtは発火する。なぜなら、もしtが$[t_0,t_1)$内に発火しないと仮定する。強連結なので、tからCのどこかのトランジションt'へ向かうパスがあって、いくつかの基本サーキットの部分をつなげて構成されている。各基本サーキットには１個しかトークンが無いため、tが$[t_0,t_1)$内に発火しないならば、そのパスを通って$[t_0,t_1)$内にはtからt'にトークンが決して渡らない。これは「C上の全てのトランジションがブロックされない」ということに反する。こうして、時間λの間にすべてのトランジションが発火する。

さて、どのトランジションも時間λごとに発火する。というのは、もし、どれかのトランジションが時刻t''で発火し、時間幅λの区間$[t'',t''+\lambda)$で発火しないとすると、業務取引ペトリネット自体はt''を「初期時刻」とみなしたときも周期的にふるまっているので、「時間λの間にすべてのトランジションが発火する」ということに反するからである。

以上から、１周期の間にすべてのトランジションがただ１回だけ発火する。

Q.E.D.

次に、上の命題の条件を満たす業務取引ペトリネットでは、状態遷移表と状態方程式の計算が一致することを示す。

命題 12

強連結で活性であり、平衡状態において任意の基本サーキットにただひとつのトークンが含まれるような業務取引ペトリネットをとる。周期状態の任意の状態について、それを表すカウンタ変数の max-plus ベクトルが状態遷移行列の固有ベクトルとなる。また、その固有ベクトルを初期値とする行列方程式は、状態遷移表と一致する。つまり、行列方程式が状態遷移を正しく表現する。

（証明）

命題 11 から、周期状態においてすべてのトランジションは、最大サイクル平

均値λごとに発火している。つまり、任意の時刻tに発火すると次に発火するのは$t+\lambda$である。よって、周期状態において、全トランジションを任意に番号付けておいて、k回目に発火した時刻をならべて発火のカウンタ変数を$X(k)$とすると、

$$X(k+1)=k\otimes X(k)$$

である。一方、この強連結で活性な業務取引ペトリネットの状態遷移行列Aは常に存在し、$X(k+1)=A\otimes X(k)$である。したがって、乗算記号を省略表記した形だと$AX(k)=kX(k)$であり、固有ベクトルとなる。したがって、任意の生起回数tについて

$$X(t)=A^{t}X(0)$$

によって、状態遷移（のトランジション発火時刻）を計算できる。（証明終）

例を示す。図 8-18(a)は２章の図 2-1 組立プロセスを強連結構造にしたものである。同図(b)はそれと同型な業務取引ペトリネットである。

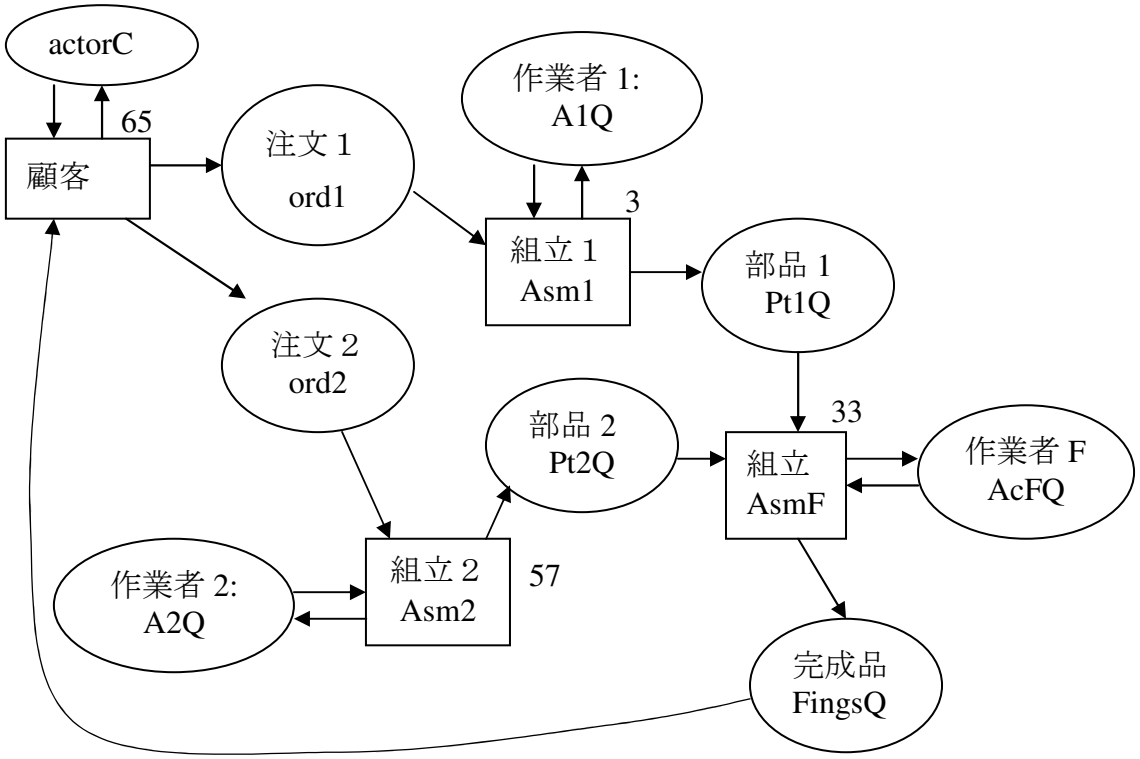


図 8-18(a)　強連結な組立プロセス

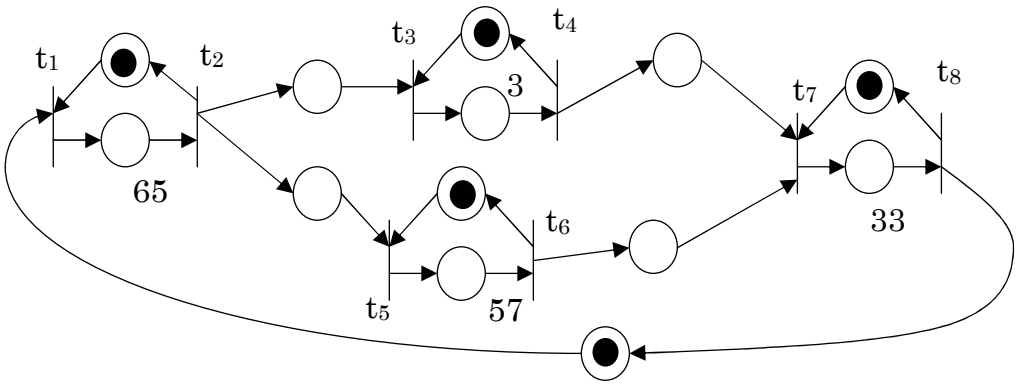


図 8-18(b)　同型な強連結業務取引ペトリネット

問 8-2

図 8-18(b)の業務取引ペトリネットの６つの基本サーキットをすべて示せ。

図のペトリネットは強連結で任意の基本サーキットにただひとつのトークンを含み、活性である。図 8-18(a)の状態遷移表を表 8-2 に示す。時刻 0 時点から周期が 155 の平衡的動作をしていることが分かる。各活動は 1 周期の間にすべて 1 回だけ発火（開始）している。業務取引ペトリネットはこれと同型な状態遷移をする。たとえば、時刻 155 において業務取引システムの Cust 活動が次回の注文を 65 分後に出すために(1,65)になり、時刻 220 に Ord1 と Ord2 にそれぞれ出力したということは、業務取引ペトリネットにおいてはトランジション t_1 が時刻 155 に発火し、155+65=220 時点で t_2 が発火することである。このような対応によって、業務取引ペトリネットの各トランジションの発火時刻を表にすると表 8-3 を得る。（業務取引ペトリネットの状態遷移表を直接作ることももちろんできる。）

時刻	Cust	Ord1	Ord2	A1Q	Asm1	Pt1Q	A2Q	asm2	Pt2Q	AcFQ	AsmF	FingsQ
0	-cus-	0	0	1	-asm1-	0	1	-asm2-	0	1	-asmF-	1
0	(1,65)	0	0	1	-asm1-	0	1	-asm2-	0	1	-asmF-	0
65	-cus-	1	1	1	-asm1-	0	1	-asm2-	0	1	-asmF-	0
65	-cus-	0	1	0	(1,3)	0	1	-asm2-	0	1	-asmF-	0
65	-cus-	0	0	0	(1,3)	0	0	(1,57)	0	1	-asmF-	0
68	-cus-	0	0	1	-asm1-	1	0	(1,54)	0	1	-asmF-	0
122	-cus-	0	0	1	-asm1-	1	1	-asm2-	1	1	-asmF-	0
122	-cus-	0	0	1	-asm1-	0	1	-asm2-	0	0	(1,33)	0
155	-cus-	0	0	1	-asm1-	0	1	-asm2-	0	1	-asmF-	1
155	(1,65)	0	0	1	-asm1-	0	1	-asm2-	0	1	-asmF-	0
220	-cus-	1	1	1	-asm1-	0	1	-asm2-	0	1	-asmF-	0
220	-cus-	0	1	0	(1,3)	0	1	-asm2-	0	1	-asmF-	0
220	-cus-	0	0	0	(1,3)	0	0	(1,57)	0	1	-asmF-	0
223	-cus-	0	0	1	-asm1-	1	0	(1,54)	0	1	-asmF-	0
277	-cus-	0	0	1	-asm1-	1	1	-asm2-	1	1	-asmF-	0
277	-cus-	0	0	1	-asm1-	0	1	-asm2-	0	0	(1,33)	0

表 8-2　組立プロセスの業務取引システムの状態遷移表

発火回数	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
1	0	65	65	68	65	122	122	155
2	155	220	220	223	220	277	277	310
3	310	375	375	378	375	432	432	465
4	465	530	530	533	530	587	587	620
5	620	685	685	688	685	742	742	775
6	775	840	840	843	840	897	897	930

表 8-3　業務取引ペトリネットのトランジション発火時刻表

対象の業務取引ペトリネットの任意のサーキットのトークンの最大数が 1 なので、状態遷移方程式は $X(t+1) = A_0X(t+1) + A_1X(t)$ である。ただし、

$$A_0=\begin{pmatrix}\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\65&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&3&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&57&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&0&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&33&\varepsilon\end{pmatrix},A_1=\begin{pmatrix}\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&0\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\end{pmatrix}$$

このとき、状態遷移行列$A=A_0^{*}A_1$を計算すると次となる。

$$A_0^{*}=\begin{pmatrix}0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\65&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\65&0&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\68&3&3&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon\\65&0&\varepsilon&\varepsilon&0&\varepsilon&\varepsilon&\varepsilon\\122&57&\varepsilon&\varepsilon&57&0&\varepsilon&\varepsilon\\122&57&3&0&57&0&0&\varepsilon\\155&90&36&33&90&33&33&0\end{pmatrix},A=\begin{pmatrix}\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&0\\\varepsilon&65&\varepsilon&\varepsilon&\varepsilon&\varepsilon&\varepsilon&65\\\varepsilon&65&\varepsilon&0&\varepsilon&\varepsilon&\varepsilon&65\\\varepsilon&68&\varepsilon&3&\varepsilon&\varepsilon&\varepsilon&68\\\varepsilon&65&\varepsilon&\varepsilon&\varepsilon&0&\varepsilon&65\\\varepsilon&122&\varepsilon&0&\varepsilon&57&\varepsilon&122\\\varepsilon&122&\varepsilon&3&\varepsilon&59&\varepsilon&122\\0&155&0&36&0&90&0&155\end{pmatrix}$$

計算するとAの固有値は155であり、周期と一致する。したがって、周期状態に入っている1回目の発火$X(1)$を

$$X(0)=\begin{pmatrix}0\\\varepsilon\\\varepsilon\\\varepsilon\\\varepsilon\\\varepsilon\\\varepsilon\\\varepsilon\end{pmatrix},X(1)=\begin{pmatrix}0\\65\\65\\68\\65\\122\\122\\155\end{pmatrix}$$

として任意の発火回数kについて$X(k)=A^kX(0)=A^{k-1}X(1)$によっていくつか計算すると、次を得る。簡単なプログラムを作成して計算する。

$$X(2)=\begin{pmatrix}155\\220\\220\\223\\220\\277\\277\\310\end{pmatrix}, X(3)=\begin{pmatrix}310\\375\\375\\378\\375\\432\\432\\465\end{pmatrix}, X(4)=\begin{pmatrix}465\\530\\530\\533\\530\\587\\587\\620\end{pmatrix}, X(5)=\begin{pmatrix}620\\685\\685\\688\\685\\742\\742\\775\end{pmatrix}$$

当然、定理が示す通り表 8-3 の発火時刻表と一致する。定理が示す通り周期と固有値も一致する。

このように業務取引ペトリネット（シンプル業務取引システム）が特別な条件を満たすときには、精確で詳細な時間変化を表す状態遷移表を max-plus 代数を使った状態遷移方程式によって表現できる。

ビジネスプロセス全般に適用可能なように一般的な業務取引システムについての状態遷移方程式を展開するために今後発見すべきことは多いが、ひとつの有望な方法と考えられる。微分方程式や制御理論の状態表現の発展を見本としたような、max-plus 代数を使った今後の状態遷移方程式研究と、企業プロセスシミュレータやネットワークを介した生産販売計画パラメータの自動化システムへの応用と開発など、大きな可能性がある。

問 8-3　　図 8-19 は強連結な決裁プロセスである。表 8-4 はその周期動作での状態遷移表である。

（１）同型な業務取引ペトリネットに変換しなさい。活動が４つなのでトランジションは８つある。

（２）８つのトランジションに適当に番号付けをし、周期状態の状態遷移表のひとつの行に注目し状態遷移方程式 $X(t+1)=A_0X(t+1)+A_1X(t)$ の２つの行列を求めなさい。

（３）$A=A_0^*A_1$ を計算で求めなさい。さらに、周期状態の状態遷移表から発火時刻表に変換して、任意の発火回数 k について $X(k)=A^kX(0)=A^{k-1}X(1)$ による max-plus 計算が状態遷移表と一致することを確かめなさい。

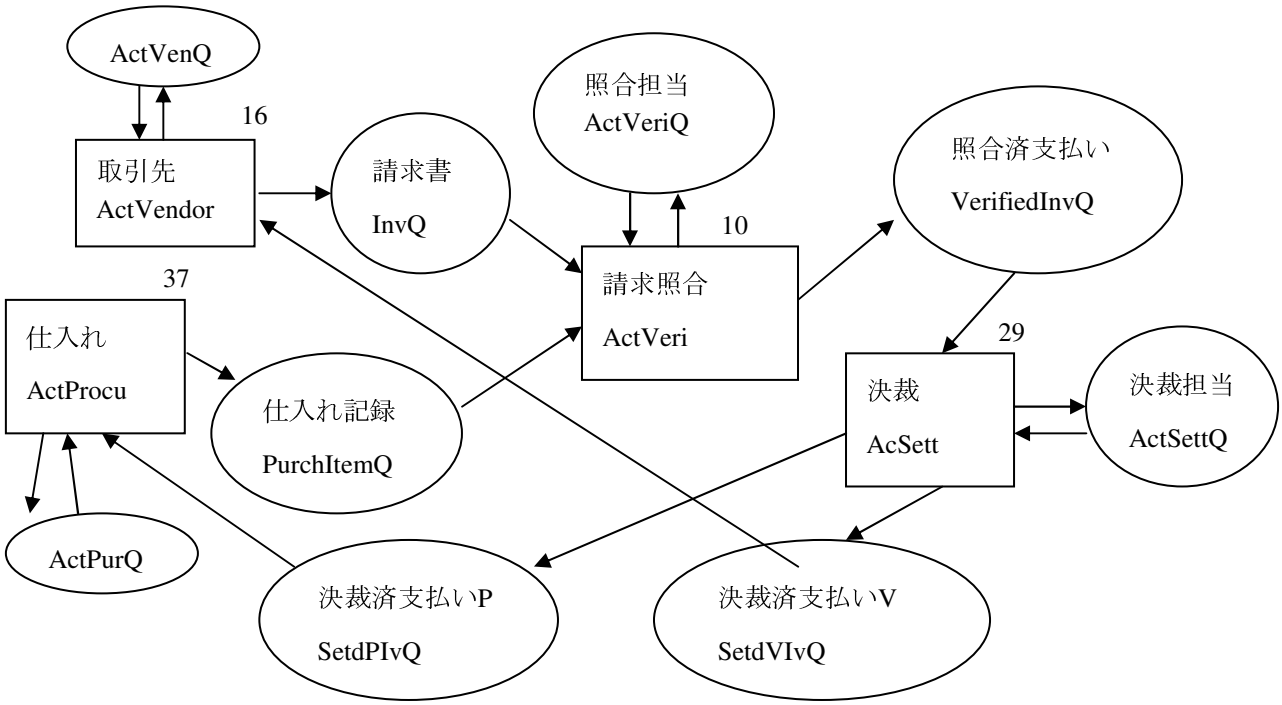


図 8-19　強連結な決裁プロセス

time	AcVnd	InvQ	AcPur	PurItQ	AcVeri	VrdIvcQ	AcSett	SetdPIvQ	SetdVIvQ
0	-aVen-	0	-aPrc-	0	-aVri-	1	-aSet-	0	0
0	-aVen-	0	-aPrc-	0	-aVri-	0	'(1,29)	0	0
29	-aVen-	0	-aPrc-	0	-aVri-	0	-aSet-	1	1
29	'(1,16)	0	-aPrc-	0	-aVri-	0	-aSet-	0	1
29	'(1,16)	0	'(1,37)	0	-aVri-	0	-aSet-	0	0
45	-aVen-	1	'(1,21)	0	-aVri-	0	-aSet-	0	0
66	-aVen-	1	-aPrc-	1	-aVri-	0	-aSet-	0	0
66	-aVen-	0	-aPrc-	0	'(1,10)	0	-aSet-	0	0
76	-aVen-	0	-aPrc-	0	-aVri-	1	-aSet-	0	0
76	-aVen-	0	-aPrc-	0	-aVri-	0	'(1,29)	0	0
105	-aVen-	0	-aPrc-	0	-aVri-	0	-aSet-	1	1
105	'(1,16)	0	-aPrc-	0	-aVri-	0	-aSet-	0	1
105	'(1,16)	0	'(1,37)	0	-aVri-	0	-aSet-	0	0
121	-aVen-	1	'(1,21)	0	-aVri-	0	-aSet-	0	0
142	-aVen-	1	-aPrc-	1	-aVri-	0	-aSet-	0	0
142	-aVen-	0	-aPrc-	0	'(1,10)	0	-aSet-	0	0
152	-aVen-	0	-aPrc-	0	-aVri-	1	-aSet-	0	0
152	-aVen-	0	-aPrc-	0	-aVri-	0	'(1,29)	0	0

（それぞれの作業者を表す ActVenQ、ActPurQ、ActVeriQ、ActSettQ は初期値はいずれも 1 であり表では省略。他の初期値は請求書= 0, 照合後支払い=1, 仕入れ記録= 0, 決裁済み支払い P= 0, 決裁済み支払い V= 0 である。）

表 8-4 強連結な決裁プロセスの周期的状態遷移表

8.8 ビジネスプロセスの定性的性質

本章では、ビジネスプロセスのモデルとして、アクティビティ・インタラクション・ダイアグラム(AID)やデータフローダイアグラム（DFD）とペトリネットを比較したり使用する方法を示した。離散事象システムのモデル化において、DEVS モデルとペトリネットの理論的関係は明らかにされていなかったが、業務取引システムの構造を持つビジネスプロセスという対象に対して、シンプル業務取引システムを考えるとそれら２種類のモデルが等価なモデル化能力を持つことが分かった。

ビジネスプロセスの制御機構として、現在もっとも発展しているのは、産業発展の歴史的理由により、生産管理の制御機構であろう。情報システム関係の発展は、コンピュータプログラムによって仕事や計算論理・機構の定義を記述することに忙しく、情報システムの適用対象であるビジネスシステムの制御や最適性能の考えと設計までを視野に入れにくかったのである。本章の結果はペトリネットで扱われることが多い生産管理の制御機構を、DFD による購買や販売のサービス活動が結合したビジネスプロセスについての管理に適用するための基礎的な理解を与えている。

情報システム方法論で広く用いられている DFD のモデルに接続したペトリネットのモデル化の方法が示された。これは、たとえば max-plus 代数というような、時間ペトリネットの時間特性の計算方法をビジネスプロセス設計に応用する際の基礎である。

本章で調べたことはまだ原理的段階であるためにすぐに特定のビジネスシステムの改善に利用できる訳ではない。一方で、メーカ各社や流通業で進められている顧客リードタイムの短縮であるとか、電子取り引きによる業態の変化を考える場合に、ビジネスプロセス全般について成立する事実であり一般的なヒントとなる。

ビジネスプロセスの全体が強連結に閉じていればかならず周期を持つ。ビジ

ネスプロセスの個々の活動をつなぐ物やサービス要求から成る「在庫」の量と活動の処理スピードから定まるそれら「在庫」の平均滞留時間という個別活動についての量が、業務取引システムという分業の仕組みを通じて、全体としての最大平均滞留時間（最大サイクル平均値）に影響する仕組みの一端を明らかにした。そして、ビジネスプロセスの状態遷移表の周期を max-plus 代数による状態遷移行列の固有値の計算で得る方法を示した。7 章では強連結プロセスが結合してでき上がる全体プロセスを分析した。時間的変動が有界な結合情報や結合在庫によって、部分プロセスが連結されたものである。購買・製造・営業・会計・マーケティング・R&D などの部門を結合したいわば 1 社だけのビジネスプロセスに止まらず、サプライチェーンやそれらの結合系としての経済社会プロセスもモデル化の対象に含まれている。ただし、現在ではそうした有界結合な業務取引システムに対する max-plus 方程式の適用方法がよく発展している訳ではない。今後理解が進むだろう。

納期や在庫期間という時間生産性の特性については、個別活動の特性値と全体プロセスの特性値との関係は直接的には明らかにされていないが、最大サイクル平均値の場合と同じように、最大の納期のサーキット、または最大の在庫期間を持つサーキットにおいて処置を講じなければ、全体プロセスとしての効果がないことがわかる。

時間生産性の高いビジネスプロセスを構築するには、全体としてクローズなフィードバックの流れを作り、作業や工程をつないでいるモノやサービスの在庫を全体として小さくすること、そしてそれぞれの作業や工程の処理スピードを上げることである。情報システムの基本的目的である。工程の一部やほとんどがアウトソーシングされていても、 購買から顧客へのビジネスプロセス全体がクローズしている必要がある。クローズしていないビジネスプロセスは、どこかに在庫が無制限に蓄積する可能性をはらんでいる。全体をクローズさせて強連結にするのは難しい場合があることが予想される。その場合でも、ある一定範囲の需要のパターンに対しては強連結な業務単位を 7 章でのべたような有界結合な方法で定常運転できるような業務構造にしておかないと、自社のビジネスプロセスの動的な性能をとらえることはできない。そうでなければ、やり方だけはしっかり決めてあるが、どこがボトルネック業務なのか、どこを改善すればよいのかがわからず、いきおい在庫の目立つ部署やまとはずれの全社コスト削減活動を始めたりしかねない。業務の処理スピードはやみくもにすべて

のものを向上させる必要はない。e-コマースやサプライチェーンにおいて、周期が異なるビジネスプロセスを統合しても、一番遅い周期の速さしか得られない可能性があるのである。モノやサービスやデータによってプロセス内にたくさん形成されているフィードバックループのうちの、最大サイクル平均値を持つループの処理時間をまず小さくするような方策がとられなければならない。

工学的設計法としては残された課題は多い。いろいろな経営の制御システムについての分業の仕組や時間生産性の分類や性能比較などの課題が数多く残っている。

付録 1　本章の記号一覧表

記号	意味
BTS-PN	業務取引ペトリネット
DEVS	discrete-event specification およびそれが表す離散事象システム
$dueP$	最も満期が近い将来の終了トランジションを示す関数
e	すべての busy プレースのトークンの経過時間を並べたベクトル
e^1	δ_1 による発火の結果定まるすべての busy プレースのトークンの経過時間を並べたベクトル
e^2	δ_2 による発火の結果定まるすべての busy プレースのトークンの経過時間を並べたベクトル
e_a	busy プレース q_a 内のトークンの経過時間、および q_a に対応する BTS の活動が持つ経過時間
$e^1(q_a)$	ベクトル e^1 の第 q_a 成分
F	アーク（矢印）の集合
$File$	BTS のファイルシステム
f_μ	マーキング μ から $\Psi(\mu)$ として定まるファイル内容関数
$f_{FA}(f_\mu)$	ファイル f_μ から決まる発火可能な活動の集合
$f_{FileValue}$	ファイルへの活動による更新結果を与える関数
$f_{AK}^F(a)$	活動部分 q_a の出力トランジションからの出力プレースに対応する管理実体型の集合
$f_{AK}^S(a)$	活動部分 q_a の入力トランジションからの出力プレースに対応する管理実体型の集合
$f_{AK}(a)$	活動部分 q_a からの出力プレースに対応する管理実体の集合
$f_{KA}(a)$	活動部分 q_a への入力プレースに対応する管理実体の集合
$f_F(f_\mu)$	f_μ から開始可能な一連の活動を開始させる仕組みを記述する関数
$\mathrm{f}_{\mathrm{Dispatch}}(f_\square)$	f_μ から開始した一連の活動によってファイルシステムを更新する関数
f_μ^a	f_μ を $f_{KA}(a) \cup f_{AK}(a)$ という管理実体型に制限したファイル内容関数
G_a	BTS の活動を表す DEVS
$g(\mu)$	μ から $st(\mu)$ を番号順に発火させ得られるマーキング
h	busy プレースがトークンを持ったときの保持時間を定めた関数
h_a	q_a 内のトークンの保持時間

$I(t)$	トランジション t への入力プレースの集合
J	ペトリネット構造。 $J=\langle P,T,F,W\rangle$
$k(\mu)$	μからの最終的な発火結果として得られるマーキング
KS	管理実体型（ファイル名の集合）
N	0 を含む自然数集合
N^n	自然数を n 個並べたベクトルの集合
n_A	活動部分の個数
$\underline{n_A}$	BTS の活動の名前の集合
$O(t)$	トランジション t からの出力プレースの集合
P	プレースの有限集合
PN	ペトリネット
P(T)	集合 T のべき集合
R	実数全体からなる集合
R^{n_A}	n_A 個の実数値を成分とするベクトルの集合
R^{∞}	R に無限大 ∞ を付け加えた集合
S	BTS の状態空間
S_a	f_μ^a の集合（G_a のファイルシステム）
$S_a \times R^{\infty}$	G_a の状態空間
$st(\mu)$	μで発火可能な入力トランジションの集合
T	トランジションの有限集合
ta	BTS の満期時間関数
$taP(\mu,e)$	終了トランジションの発火時刻まで満期時間
t_k^i	busy プレース k の入力トランジション
t_k^o	busy プレース k の出力トランジション
W	アークの重みを表す関数
μ, μ_0	マーキング
w_{pi}	BTS-PN のプレース p へ入るアークの重み
w_{po}	BTS-PN のプレース p から出るアークの重み
$w(p,t)$	p から t へのアークの重み
$w(t,p)$	t から p へのアークの重み
δ	BTS-PN の状態遷移関数

δ_1	満期の活動の出力トランジション発火による結果を表わす関数
δ_2	発火可能な入力トランジション発火による結果を表わす関数
δ_a	G_aのファイルシステムの更新関数
δ_{aI}	G_aの活動開始によるG_aのファイルシステムの更新関数
δ_{aO}	G_aの活動終了によるG_aのファイルシステムの更新関数
δ_M	BTS の状態遷移関数
Ψ	BTS-PN のマーキングを BTS のファイル内容関数で表す関数

付録 2　可換性の証明（記号は本文中の説明で用いたものを使う）

命題 1

関数　$k(\mu)=\begin{cases}\mu, \text{ if } st(\mu)=\phi; \\ k(g(\mu)), \text{ if } st(\mu)\neq\phi.\end{cases}$

は well-defined である。

（証明）　$st(\mu)\neq\phi$とすると、$st(\mu)$の入力トランジションに対応する活動部分は、トークンを取り組むために、もはやその時刻には開始することはできない。活動部分の数は有限個だから、かならず有限回の、たとえばp回の、gの適用によって$st(g^p(\mu))=\phi$となる。　Q.E.D.

以下で、本文で示したような BTS-PN とそれから作られる業務取引システムの状態遷移の可換性が一般的に成立することを証明する。

BTS-PN を所与とする。Ψと id でマップして、対応する BTS を作ったとする。 このとき以下の順序で可換性が示される。

命題 2

$st(\mu)=\{t^i_{k_1},t^i_{k_2},...,t^i_{k_r}\}\Leftrightarrow f_{FA}(f_\mu)=\{a_{k_1},a_{k_2},...,a_{k_r}\}$である。ただし、各$a_{k_j}$は$t^i_{k_j}$を入力トランジションとする活動プレースと 1 対 1 に対応する BTS の活動である。

（証明）定義より明らかである。　Q.E.D.

命題 3

$taP(\mu,e)=ta(f_\mu,e)$

（証明）任意の $q_a \in P_A$ について $h(\mu, q_a) = ta_a(f_\mu^a)$ であることから成立する。

Q.E.D.

命題４

$dueP(\mu,e) = \{t_{i_1}^o, t_{i_2}^o, ..., t_{i_m}^o\} \Leftrightarrow due(f_\mu,e) = \{a_{i_1}, a_{i_2}, ..., a_{i_m}\}$ である。ただし、各 a_{i_j} は $t_{i_j}^o$ を入力トランジションとする活動プレースと１対１に対応する BTS の活動である。

（証明）

dueP と *due* の定義と、命題２から成立する。 Q.E.D.

命題５

$\delta_1(\mu,e) = (\mu^1, e^1)$ とするとき、

$$\Psi(\mu^1) = f_{FileValue}(f_\mu, due(f_\mu,e))$$

である。

（証明）

命題４により、μに適用する出力トランジション群と、f_μ に適用する活動群は１対１に対応する。さらに、 $due(f_\mu,e)$ の各活動はすでに活動中でアイドルファイルの個数属性の値は0となっているので、$f_{FileValue}(f_\mu, due(f_\mu,e))$ は各活動の開始活動は行われず、出力処理だけを順に行う。したがって、δ_1によって順に出力トランジションによって μの変更を行った結果と、Ψによって対応する。 Q.E.D.

命題６

$\delta_2(\mu,e) = (\mu^2, e^2)$ とするとき、

$$\Psi(\mu^2) = f_F(\Psi(\mu))$$

(証明)

命題２によって $st(\mu) = \{t_{k_1}^i, t_{k_2}^i, ..., t_{k_r}^i\} \Leftrightarrow f_{FA}(f_\mu) = \{a_{k_1}, a_{k_2}, ..., a_{k_r}\}$ が成立している。

$st(\mu) = \phi$ のとき $f_{FA}(f_\mu) = \phi$ だから、$\mu^2 = \mu$ かつ $\Psi(\mu^2) = f_\mu = f_F(f_\mu)$ である。

$st(\mu) \neq \phi$ とする。δ_2 は $st(\mu)$ の各入力トランジションについて、順に発火の操作を μ に対して行い、結果としてと μ' となる。同じように、$f_{FileValue}(f_\mu, f_{FA}(f_\mu))$ は、$st(\mu)$ に１対１に対応する $f_{FA}(f_\mu)$ の活動について、順に終了と開始の操作の組を行いと f'_μ となる。このとき、$f_{FA}(f_\mu)$ の要素である活動は busy でないので、終了の操作は行われ

ず開始の操作だけが行われるので、$\Psi(\mu') = f'_{\mu} = f_{\mu'}$である。これを繰り返すが、命題2によって$st(\mu')$と$f_{FA}(f_{\mu'})$はいつも1対1に対応していることから題意が成立する。

Q.E.D.

命題7

初期条件を満たす任意のBTS-PNについて、図8-11の可換図は可換となる。

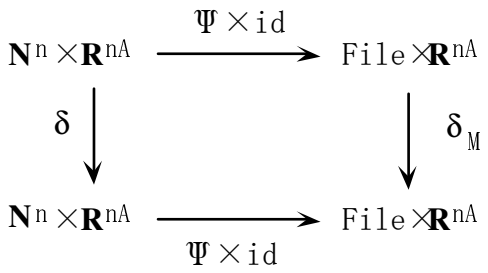


図8-11(再掲)　可換図

（証明）

1) 初期状態条件を満たすような任意の(μ,e)を$N^n \times R^{n_A}$からとる。

2) $\delta(\mu,e) = \delta_2(\delta_1(\mu,e))$である。 $\delta_1(\mu,e) = (\mu^1,e^1)$ とすると、命題5より

$$\Psi(\mu^1) = f_{FileValue}(f_{\mu}, due(f_{\mu},e))$$

である。 $\delta_2(\mu^1,e^1) = (\mu^2,e^2)$とすると命題6から、

$$\Psi(\mu^2) = f_F(\Psi(\mu^1)) = f_F(f_{FileValue}(f_{\mu}, due(f_{\mu},e)))。$$

3) 経過時間e_iが0になるのは終了と開始の時だが、(2)のμについての可換性と命題4および命題2によって、経過時間が変化するBTS-PNの活動部分とBTSの活動は1対1に対応する。また、終了も開始もしないものは、時間が進むだけだが、命題2より同じ経過時間となる。　Q.E.D.

第９章　離散事象システムの標準状態表現と DEVS モデルのユニーク性

9.1　離散事象システムいう方法論

9.1.1 ビジネスプロセスモデルの標準性

本章はこれまでの章と視点を変え、離散事象システムというモデルの特徴を一般的にとらえること、さらに、業務取引システムに使われている DEVS という方法論によって離散事象システムの標準的なモデル化を行えることを示す。これによって、DEVS というモデル化方法論をつかって離散事象システムをモデル化するなら、離散事象的な仕組みをとらえ損ねることはない。また、DEVS に同型なモデルも同等の資格を持つことになる。

本書ではビジネスプロセスの構造を表現するために、離散事象システムを使ってきた。言い換えれば、ビジネスプロセスの背後に厳然として横たわっているものが離散事象システムという仕組みであり、したがって、情報システムやビジネスプロセスを扱う場合には、意図しない場合でも必然的に離散事象システムを扱っていることになる。また、会社での仕事だけではなく、個人生活も個人の仕事のすすめ方、つまりビジネスプロセスを持つので、社会生活をする中で仕事を進めていくというレベルでは、離散事象の仕組みから逃れることはできない。そのレベルというのは、仕事をすすめるときの開始や終了だけを観るというレベルであって、個人の心理的な葛藤・成長とか脳の中での物理化学的変化を気にしないで捨象したレベルである。

9.1.2　問題の所在

離散事象システムは非常に多くの場面で見られる。生産プロセスや販売プロセスなどの組織の機構，データベースソフトウエアで使われる排他制御機構，あるいはまた，病院の診断や治療のプロセスもそのような特徴を持っている。物理的な具体的実体の場合であれば微分方程式が基本的な変数間関係を表現するし，論理的な動作とか離散時間上の変化には差分方程式が用いられてきた。ところが離散事象システムでは離散事象方程式といったものは利用可能ではなく，いろいろな問題に応じてたくさんのモデルの流儀がある。離散事象システムは、微分方程式や差分方程式によるモデルと異なる部分もあるが、どれも物事の動的な仕組みをとらえるということだから、共通する部分もある。時間的に変化する対象を人間がとらえるための一般的論理構造は、状態遷移メカニズムとして一般システム理論という分野で深く考察されてきたのであるが、本章の説明はその知見を使って離散事象という構造の解明を追加する。

ジーグラー博士は文献 (Zeigler,1976) において離散事象システムに対する DEVS 方法論を提唱した。DEVS はシステム理論に基礎をおいており、それ以来、多くの方向への応用がなされてきた。情報システム分析ではこれまで本書で示したような応用がされた。また、ペトリネットとの同型性によって異なるモデル化方法論との関連も分かってきた。

直観的に描かれた図9-1の簡単な製造ラインを考えよう。この製造プロセスを分析し管理・制御システムを設計するときにどういうフォーマリズムを採用す

るかは自明ではない。製造業での組立ラインにおいては，需要の多様性への対応と原価削減を同時に進める方法としてリーン生産システムを構成することがすすめられている（ウオマック/ジョーンズ, 1997）。カンバン方式はその代表例である。

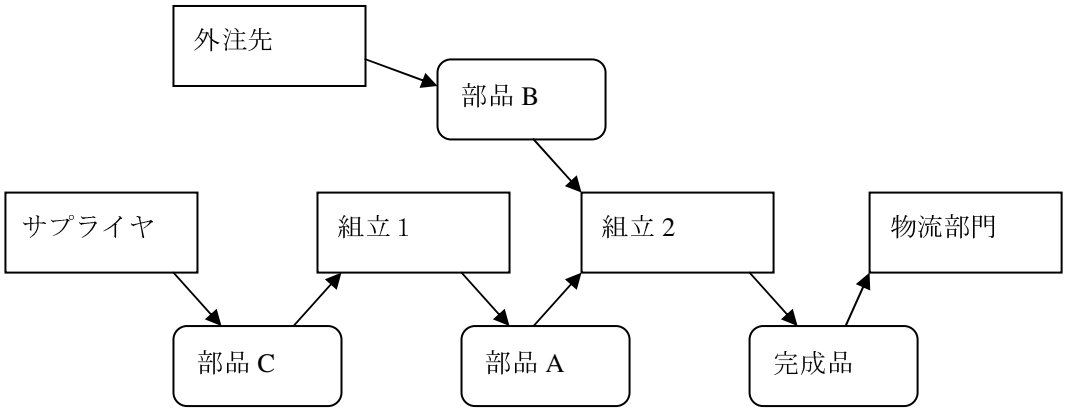


図 9-1　製造ライン

下請け業者から加工品が届けられそれをトランスミッションのような中間部品に組み立てて，さらにエンジンに取付けるラインを想定する。 このようなプロセスをコントロールするのに，カンバンシステムが用いられる（門田, 1991, 2006)。

カンバン方式の制御機構は，図 9-2 のようにペトリネットでも表現されるし，また，図 9-3 のような DEVS 結合システムでも表現できる。現実に離散事象的なことがらの多くの分野で重要であるため、生産プロセスに限らず、オペレーティングシステムや通信などの分散システムに対して、離散事象的な並行動作の記述や制御方式のモデルとして、communicating sequential process (CSP)や並行オートマトンの生成言語制御アプローチ等々、様々なモデルが使わされている。

ここでのポイントは，同じ対象に対して，異なるモデルフォーマリズムによって異なるモデルが作られることである。

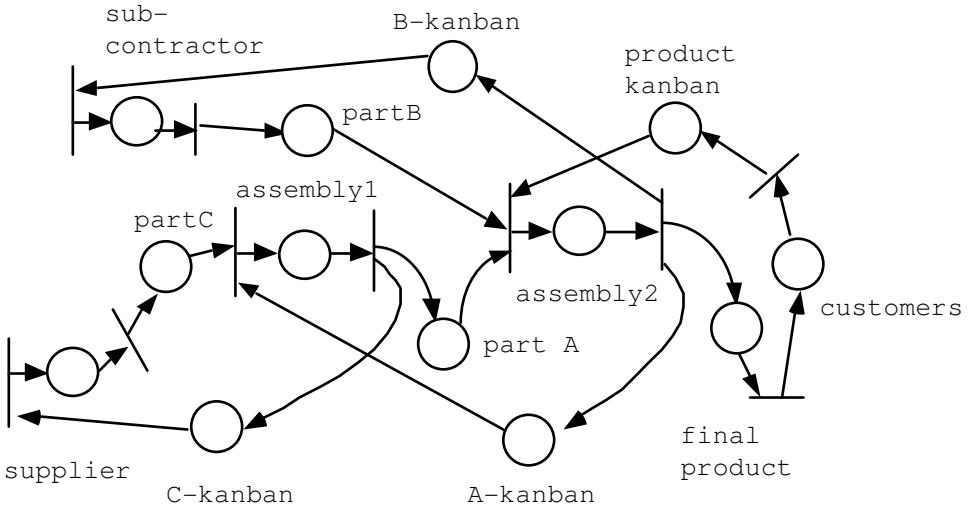


図 9-2 ペトリネットによるカンバン方式

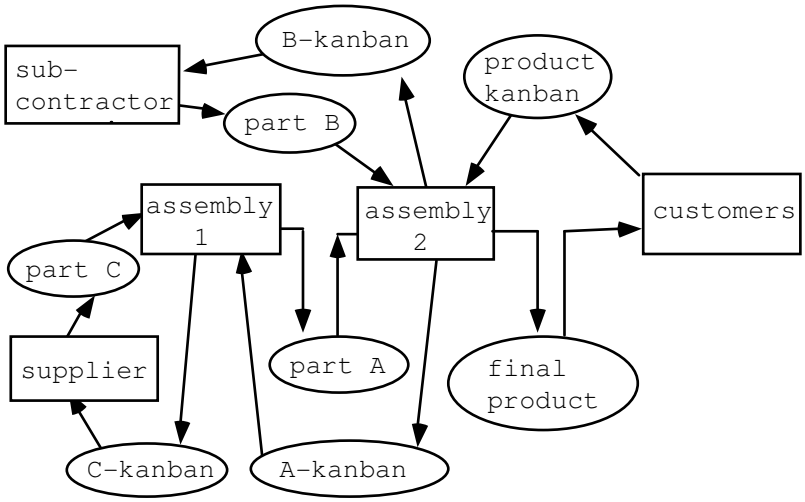


図 9-3 DEVS(BTS)によるカンバン方式

ビジネスプロセス（再）構築を命じられたとしよう。顧客から見た全待ち時間である顧客リードタイムの短縮、平均在庫の圧縮、適度な稼働率の維持、全体的なスループットの向上は、ビジネスプロセス設計の普遍的な目標である。実践する際には目標間の調整をとって、たとえば、在庫圧縮とリードタイムをそれぞれほどほどに満たすようにしなければならない。適当と思われる何らかのモデルを用いていろいろな情報システム利用や管理方策の設計を行う訳である。

ある現象をみてモデルを選ぶ際には、現象がどんな種類のものかを認識しているはずである。内部の仕組みが見える場合もあれば、注文に応じて生産されたり、宅配サービスで商品が届けられたりといった内部が見えない場合もある。本章では、考察対象のシステムは内部メカニズムが不明の場合でも扱えるように、システム入力パターンと出力パターンの集合と見なすところから議論を始める。入出力関数の集合として離散事象のシステムととらえるということである。そして、入力を出力に変換するような内部的なメカニズムの特徴は何かということに焦点を当てる。このような入出力パターンからそれを産み出す内部構造を見つける問題を、実現性理論という。

内部的メカニズムを状態遷移関数とか状態遷移メカニズムと呼んで、入出力のパターンとして認識された離散事象システムについての、本章の実現性理論によって得られる状態遷移メカニズムの表現方法の優劣について明らかにする。これによって、ひとつの問題状況についてさまざまなモデルがあり得るときに、モデルを選択するための概念的な基礎が確立するのである。図 9-4 と図 9-5 はこうした状況を図示している。

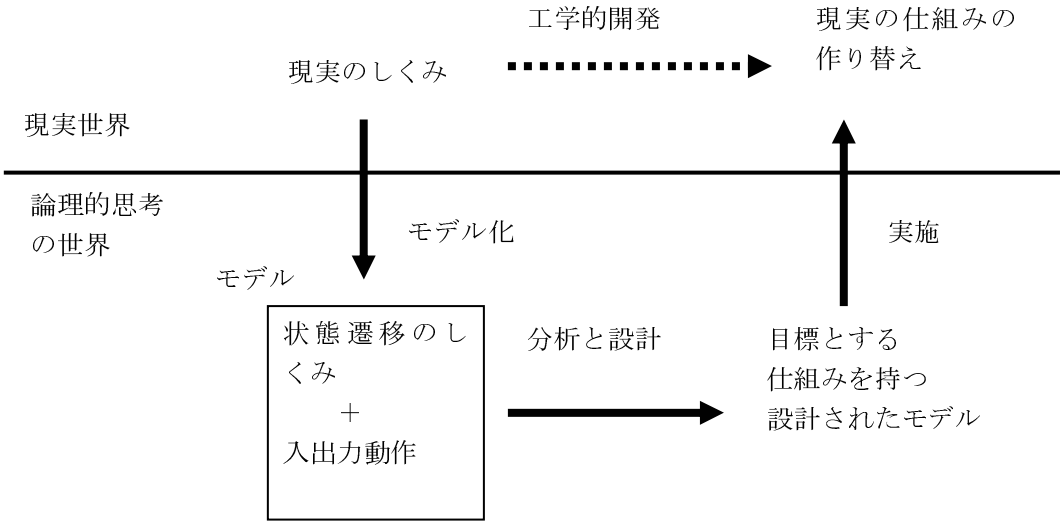


図 9-4 モデル化と工学アプローチ

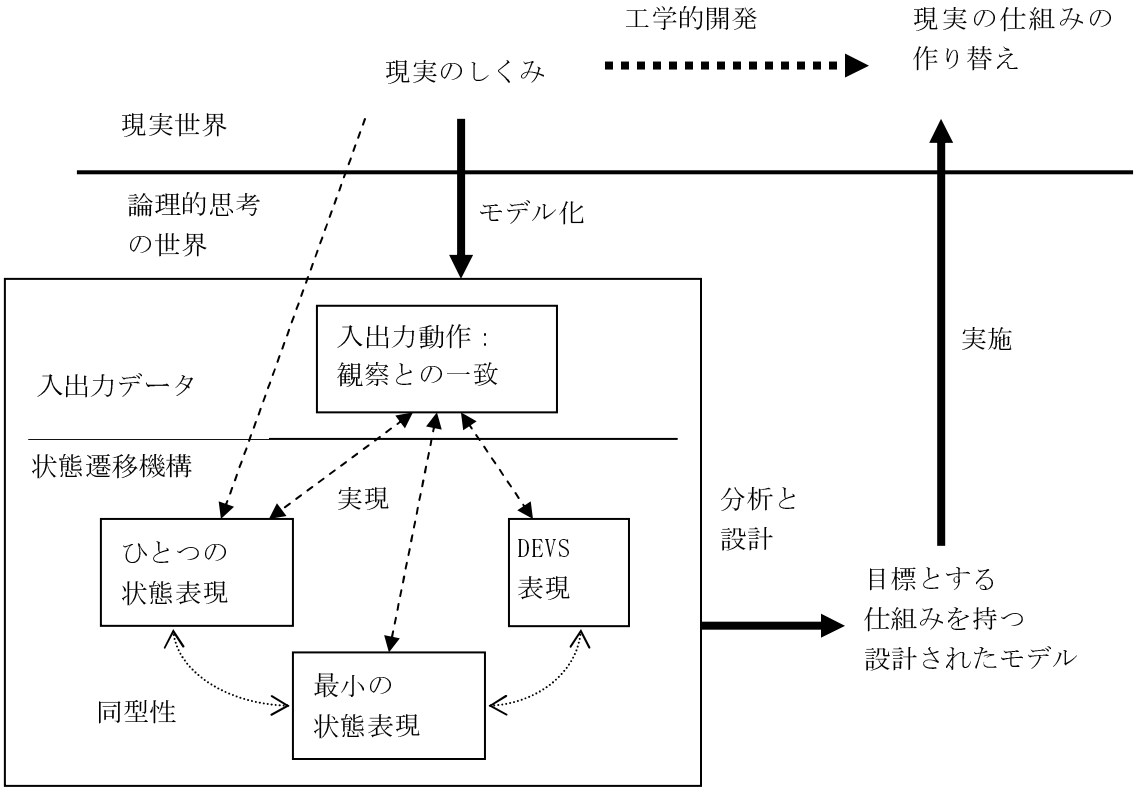


図 9-5　モデル化の内容：入出力データと状態表現（実現）

本章で設定する問を明確化する。離散事象データや離散事象現象がシステム S として観察されているとき、その状態遷移メカニズムを構築したい。この場合、S をどのようにとらえているかによって、当然、それを産み出す状態遷移関数が変わってくる。この段階で 2 つのことを考慮する必要がある。ひとつは、メサロビッチ・高原(1989) により提唱された状態表現のユニーク性の問題である。また、他のひとつは、8 章で部分的に解答したような DEVS やペトリネットや、あるいは他の離散事象モデルとの相互の位置関係の捉え方である。本章では、前者のユーク性問題への解答を示し、それを適用して DEVS 方法論のユニーク性条件を導く。

メサロビッチ・高原(1989）によれば、入出力システム S の状態表現のユニーク性問題とは一般の時間システムに対して次のように言い表わせる：

「時間システム S について、S の状態表現が（同型の範囲で）一意的になるために S が満たすべき条件は何か。」

この問題は、たとえば離散事象システムに対して、ぜひとも不可能性ではなく可能であるような条件を示す形で用意される必要がある。ユニーク性問題が解かれなければ、離散事象的現象を扱う場合に、それまで提唱されている可能な状態表現のモデルを同時的にいくつも考えて、大事な特性の考慮が抜け落ちていないか、理論的にびくつきながら分析をすすめる必要があることになる。それは、実際にはほとんど不可能である。

本章では離散事象システムの定義とそれの最小な標準状態表現として S(□)実現を定めることによって、ユニーク性問題に対する肯定的な解答を与える。S(□)実現の定義と意味は後に説明される。さらに、DEVS は 4 章で述べたような状態遷移関数を定めるが、その状態遷移関数が標準状態表現と同型であることが示される。この同型性があるため、DEVS 方法論は離散事象モデルとして必要な情報を全てかつ無駄なく表す表現力を持つことが分かるのである。

9.2 記法と基本的概念

2 章において用いたのと同じ記号を用いる。時間を表す集合を T とし、無限大を付加した集合を T^{∞} と表す。

定義 9.1　システム（メサロビッチ・高原, 1989)

システムは入力集合と出力集合との関係である。入出力集合が時間関数の集合の場合には時間システムも呼ぶ。

したがって、入力集合をXとし出力集合をYとするときシステムSは$S \subseteq X \times Y$と書かれる。TからAへの関数すべての集合をA^Tと表す。シフト演算子σ^{-t}を用いて左シフト演算子λ^tを定める。任意の時間関数xと$t,t' \in T$に対して$(\lambda^t(x))(t') = x(t+t')$と定義する。時間システム$S$に対して、$\sigma^t(S) = \{(\sigma^t(x), \sigma^t(y)) \mid (x,y) \in S\}$および$S_t = S|_{[t,\infty)} = \{(x_t, y_t) \mid (x,y) \in S\}$と定められるが、$S$が定常であるとは$\lambda^t(S) \subseteq S$であることである。また、強定常であるとは$\lambda^t(S) = S$が成立することである。定常なシステムの入力区間に対しては常に$\lambda^t(X) = X, t \in T$が成立するものとする（そのような$X$を考える）。

定義 9.2 過去決定性（メサロビッチ・高原, 1989)

$S \subseteq X \times Y$を時間システムとするとき、Sが時刻$k \in T$から過去決定的であるとは次の２条件を満たすことである：

(1) 任意の$(x,y),(x',y') \in S$について、もし$x|_{[k,t]} = x'|_{[k,t]}$ならば$y|_{[k,t]} = y'|_{[k,t]}$であること。

(2) 任意の(x^k, y^k)とx_kについて、もし$(x^k, y^k) \in S^k$ならば、あるy'_kがあって$(x^k \cdot x_k, y^k \cdot y'_k) \in S$が成立すること。

システムが過去決定的であることの意図は、定義の(1)の条件である。過去決定性の直感的意味は、システムの時間的な変動の多様性が時刻kまでのシステムの時間的変動によって、実質的に決まっているということを表している。線形微分方程式によって表現される連続システムでは、任意の$k > 0$から過去決定的である。つまり実質的に初期値の取りうる多様性がその後の入出力の時間的軌道の多様性を支配する。また、線形差分方程式の行列の次元をnとすると、その離散時間システムは$k > n$から過去決定的となる。

次の命題が自明に成立する。

補題１

時間システム$S \subseteq X \times Y$が時刻$k \in T$から過去決定的であるとする。任意の$(x,y) \in X \times Y$について、$(x,y) \in S$であることと、任意の$t \geq k$なるtについて$(x^t, y^t) \in S^t$であることは等価である。

過去決定性を取り扱う上で次の性質を利用することがある。

定義 9.3 有限観測性（メサロビッチ・高原, 1989)

時間システム$S \subseteq X \times Y$が時刻$k \in T$から有限観測的であるとは、任意の$x \in X$と$y, y' \in S(x)$について$y^k = y'^k \rightarrow y = y'$であること。

命題１（メサロビッチ・高原, 1989)

時間システム$S \subseteq X \times Y$が時刻$k \in T$から過去決定的であることは、次の２条件が成立することと同値である：

(1) Sが前因果的であること：つまり、次が成立すること。

任意の $x,x' \in X$ と $\tau, \tau \geq k,$ について、もし $x|_{[0,\tau]} = x'|_{[0,\tau]}$ ならば $S(x)|_{[0,\tau]} = S(x')|_{[0,\tau]}$ である。
(2) S が $k \in T$ から有限観測的である。

第２章８節では離散事象システムの DEVS 表現から状態遷移関数を構成する方法を述べた。記号 Λ は空イベントを表し、また、入力値集合を A とするとき $\Omega_G = \Omega_A \cup \Omega_\Lambda$ によって入力セグメント集合を定めた。

任意の集合 B に何らかの位相が定義されているものとする（たとえば通常のユークリッド距離を考える）。このとき、関数 $\Gamma : B^T \to \mathrm{P}(T)$ を定める：任意の時間関数 $y : T \to B$ に対して $\Gamma(y) \equiv \{t \mid \lim_{h>0, h\to 0} y(t-h) \neq y(t)\}$ とする。$\Gamma(y)$ は y が左から不連続になるような時刻の集合を表す。ここで $\mathrm{P}(T)$ は時間集合 T のべき集合である。

さて、離散事象システムの状態表現について異なる方法論を比較しようとするのだから、DEVS とかペトリネットとか CSP とかの特定の表現方法を表せるような一般的な基盤で議論を進める必要がある。したがって、まず、離散事象システムを入出力システムのレベルで（内部の構造や表現をまったく用いない形で）定義する。

定義 9.4 離散事象システム

記号 Λ で空イベントを表し、空以外の入力値集合を A とする。出力値集合を B とし何らかの位相が定義されているものとする。入力空間を A^T、出力空間を B^T とするシステム $S \subseteq X \times Y$ が離散事象システムであるとは次の６つの条件をすべて満たすことである。

(1) S は強定常。
(2) S は任意の正の時刻から過去決定的。
(3) X は離散事象入力空間である。つまり次を満たす。
(3.1) 空イベント値をとる定値関数 $\hat{\Lambda} : T \to \{\Lambda\}$ を含む。ここで、すべての $t \in T$ に対して $\hat{\Lambda}(t) = \Lambda$ である。
(3.2) 任意の $x \in X$ と $t \in T$ に対して、$event(x) \equiv \{t \mid x(t) \neq \Lambda\}$ とするときに $event(x) \cap [0,t)$ は有限。
(4) Y は次を満たすような離散事象入力空間である：任意の $y \in Y$ と $t,t' \in T$ に対して $\Gamma(y) \cap [t,t')$ が有限。
(5) S はシステムコアにおける出力表現性を満たす。つまり、
任意の $(x,y) \in X \times Y$ と正の $t \in T$ についてある $z \in S(\hat{\Lambda})$ があって、
（$x^t \in G_A \to z^t = y^t$）である。
(6) S はシステムコア決定性を満たす。つまり
$S[y^k] \equiv \{(x',y') \in S \mid (\hat{\Lambda}^k \cdot \sigma^k(x'), y^k \cdot \sigma^k(y') \in S\}$ と定めるとき、次を満たす（図 9-6 参照）：

$$(\forall t, t>0, \forall r, r>0, \forall y^t \in S(\hat{\Lambda})|_{[0,t)}, \forall \hat{y}^r \in S(\hat{\Lambda})|_{[0,r)}, \forall c \in S(\hat{\Lambda}))$$
$$((\hat{\Lambda}, y^t \cdot \sigma^t(c)) \in S \text{ and } (\hat{\Lambda}, \hat{y}^r \cdot \sigma^r(c)) \in S \to S[y^t] = S[\hat{y}^r])$$

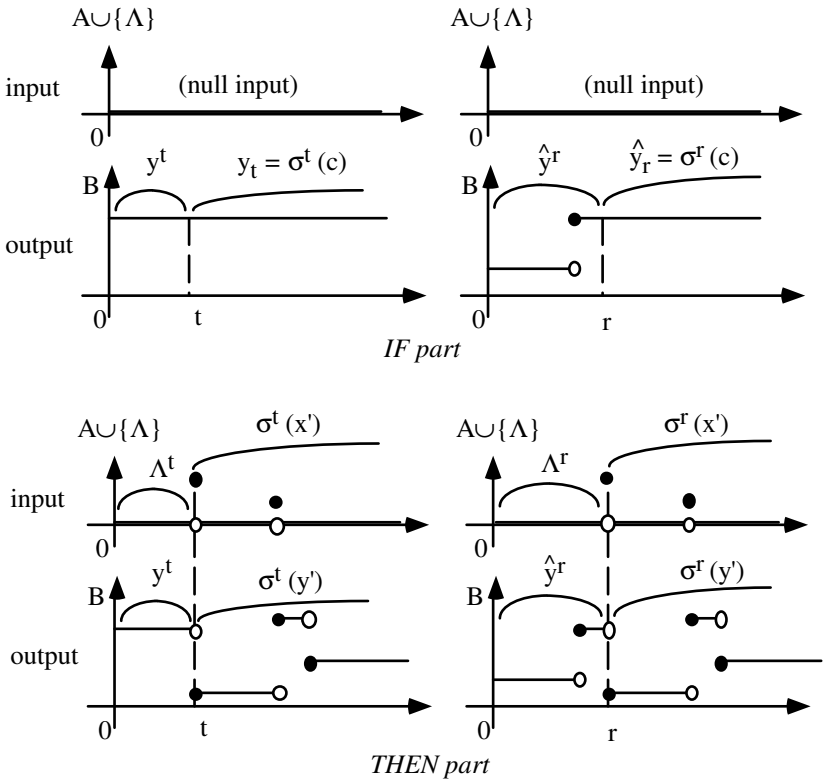


図 9-6　システムコア決定性

入出力集合としての離散事象システムの定義の意味を説明する。条件(1)は強定常性を要求し、条件(2)はシステムが強因果的であることを意味する。条件(3)は、入力空間 X が空イベントを値とする一定値関数 $\hat{\Lambda}$ を含むこと、また、任意の入力 $x \in X$ において、非空である時点は、とびとびにしかないことを意味する。以下では、特に混乱が生じない限り関数 $\hat{\Lambda}$ を Λ と書く。したがって、たとえば $S(\hat{\Lambda}) = \{y \mid (\hat{\Lambda}, y) \in S\}$ は $S(\Lambda)$ と表現される。

条件(4)は、すべての出力が、左から不連続な時刻はとびとびにしかないことを意味する。

条件(5)は入力の効果が、瞬時にシステムコア $S(\Lambda)$ に現れることを意味する。たとえば製品の入荷終了は、瞬時にその製品の在庫増加を意味する。倉庫の入荷業務では、1 個の洗濯機製品のダンボールが、チューブから絞り出すように零個からじわじわと連続的に入荷してある時間をかけて 1 個のダンボール姿の洗濯機になるようなことは決してない。未入荷か入荷済みかのどちらかしかない。なお、微分方程式で表されるような連続システムでは、連続入力の出力生成効果は、各入力値に対応した重み関数（インパルス応答）の畳み込み積分で表現されるが、離散事象システムではそのような連続的影響蓄積効果はない。

条件(6)のシステムコア決定性は、システムコアの過去の履歴がシステムの未来のふるまいの多様性をある程度決定してしまうということの表現であり、オートマトンの実現性理論であるネロード実現の離散事象システム版であると

解釈できる。ネロード実現は以下のように構成される。

一般に集合 A に対して A^* を A の要素で作られる文字列の集合とする（free monoid と呼ばれる）。文字列 x の長さを文字列を構成する文字の個数とし、$len(x)$ と表す。関数 $f:A^* \to B^*$ が順序関数であるとは次の 2 条件を満たすことである。

(1) f は長さを保つ：任意の x に対して $len(x) = len(f(x))$

(2) f は非予測的である：$(\forall x, x' \in A^*)(\exists y)(f(x \cdot x') = f(x) \cdot y)$

順序関数 $f:A^* \to B^*$ を実現する状態遷移メカニズムであるオートマトンとしてネロード実現が知られている。まず、A^* の上に同値関係 E を定める。A^* の任意の２つの ω, ω' がネロード同値であることを

$(\forall x \in A^*)(\exists y \in B^*)(f(\omega \cdot x) = f(\omega) \cdot y \wedge f(\omega \cdot x) = f(\omega') \cdot y)$

と定め、$(\omega, \omega') \in E$ とかく。オートマトンは状態空間を A^*/E として状態遷移関数 $\delta:(A^*/E) \times A \to A^*/E$ と出力関数 $\mu:(A^*/E) \times A \to B$ によって定めることができる。$\delta([x], a) = [xa]$、および $\mu([x], a) = b \Leftrightarrow f(x \cdot a) = f(x) \cdot a$ と定める。これらの関数の定義は well-defined である。初期値を $[\Lambda]$ ととると、任意の文字列 x に対して $f(x) = \mu([\Lambda], x)$ となる。このオートマトンをネロード・オートマトンとか、ネロード実現と呼ぶ。

さて、ネロード実現において、

$f[x] = \{(x'', y) \mid (f(x \cdot x'') = f(x'') \cdot y\}$ と定める。つまり $f[x]$ は、入力 x の後ろにつながる入力 x'' に対して関数 f によって出力可能な y のペア (x'', y) の集合である。入力セグメント ω, ω' がネロード同値のときには $f[\omega] = f[\omega']$ が成立するので、入力セグメント ω, ω' はシステムを同じ状態に運んだと解釈できる。なぜなら同じ入力に対して同じ入出力応答をもたらすからである。

システムがシステムコア決定的である場合にも同様の仕組みが観察される。任意の正の時刻 $k, r \in T$ と y^k, y''^r について、共通の $c \in S(\Lambda)$ によって $y^k \cdot \sigma^k(c) \in S(\Lambda)$ および $y''^r \cdot \sigma^k(c) \in S(\Lambda)$ となるときに、システムコア決定性により $S[y^k] = \{(x'', y'') \in S \mid (\hat{\Lambda}^k \cdot \sigma^k(x''), y^k \cdot \sigma^k(y'') \in S\} = S[y''^r]$ が保証される。つまり、それぞれの時刻以降の入出力の対応は同じであることになる。いいかえれば、将来のシステム行動が，過去のシステムコアを有限時間観察することによって決まるのである。これが，システムコア決定性と呼ぶ理由である。

したがって，システムコア決定性は，Nerode 実現における入力セグメントのかわりにシステムコアのセグメントに注目し，ネロード実現における入力列の同値類のかわりに，システムコアに関連するオブジェクトを状態空間として利用できることを示唆する性質である。実際，あとで示す実現においては，システムコア $S(\Lambda)$ を状態として状態遷移メカニズムを構成できることが示される。

さて、S’を離散事象システムとし、P は y の値集合から z の値集合への写像とするとき，

S = {(x, y) | ある(x, z) εS'があって，y = P(z)}

も離散事象システムである（図 9-7）。

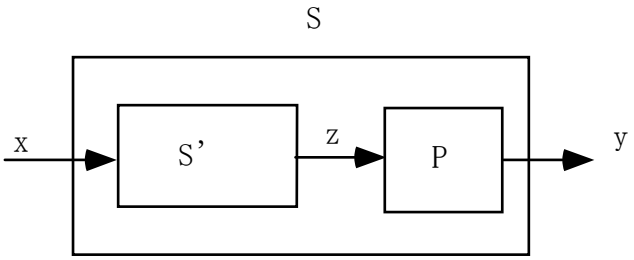


図 9-7　離散事象システム

9.3　離散事象状態表現

本節では離散事象入出力システムの状態表現を構成する。線形システムの場合には、線形入出力システムが基本線形システムとして定められて、状態応答空間 $S(0) \equiv \{y \mid (0,y) \in S\}$ を状態として標準状態表現が構成できる。線形システムの場合には $S(0)$ をシステムコアと呼ぶ（メサロビッチ&高原、1989）。入力に対してシステムが発生できる出力の多様性、すなわちシステムが持つ複雑生成能力が端的に $S(0)$ に現れているため、システムの本質的・中心的部分という意味でコアと呼んでいる。行列表現された線形システムでは、行列の固有値がコアを表現していると考えてよく、実際、安定性や振動の具合を表現する。

本節で示すように、線形システムの場合のシステムコアは離散事象システムのシステムコア $S(\Lambda)$ に対応する。

まず、時間システムのグルーバルな応答関数といくつかの性質を定義する。

定義　9.4　初期状態応答（メサロビッチ・高原、1989）

$S \subseteq X \times Y$ を時間システムとする。ある集合 C と関数 $\rho_0 : C \times X \to Y$ があって次を満たすとき、ρ_0 をシステムの初期状態応答、C を状態と呼ぶ。

$(x,y) \in S \Leftrightarrow (\exists c \in C)(\rho_0(c,x) = y)$

初期状態応答関数が因果的であるとは、任意の $x,x' \in X$、$c \in C$、$t \in T$ に対して $x|_{[0,t]} = x'|_{[0,t]}$ ならば $\rho_0(c,x)|_{[0,t]} = \rho_0(c,x')|_{[0,t]}$ なることである。因果的初期状態応答を持つシステムを因果的システムと呼ぶ。また、任意の $c,c' \in C$ に対して $(\forall x)(\rho_0(c,x) = \rho_0(c',x)) \to c = c'$ が成立するとき、初期状態応答関数が規約であるという。

時間システム対しては、第２章で述べたように、状態遷移関数と出力関数を用いて時間的な変化を記述することができる。メサロビッチ・高原(Abstract SystemsTheory, 1989)にあるように、もし時間システムが状態表現 $<\Phi,\mu>$ を持つなら、任意の $x \in X$、$c \in C$、$t \in T$ に対して $\rho_0(c,x)(t) = \mu(\phi_{0t}(c,x_{0t}),x(t))$ によって ρ_0 を定めると、ρ_0 は因果的となる。つまり、時間システムが状態表現を持つならばそのシステムは因果的である。

離散事象システム S に対する状態表現に対しては，離散事象システムにとって独自な性質を持つ物だけを考える。

定義 9.5　全射的空イベント遷移関数（Zeigler, 1996)

離散事象システムの状態表現$<\Phi,\mu>$が強い意味の全射的空イベント遷移関数であるとは、任意の$c \in C$に対して、ある$c' \in C$から$t \in T$時間で到達できることである。つまり$c = \phi_{0t}(c,\Lambda_{0t})$が成立することである。

また、$c \sim c'$であることの定義を、任意の$x \in X$に対して$\rho_0(c,x) = \rho_0(c',x)$とする。$<\Phi,\mu>$が弱い意味の全射的空イベント遷移関数であるとは、任意の$c \in C$に対して、ある$c' \in C$と$t \in T$があって$c \sim \phi_{0t}(c',\Lambda_{0t})$が成立することである。

当然、空イベント遷移関数が強い意味で全射的であれば、弱い意味で全射的である。本章では離散事象システムに対しては、必ず時間不変で弱い意味の全射的空イベント遷移関数を持つ状態遷移関数が存在することを示す。よって、本章では、離散事象システムの状態遷移関数は常に時間不変で弱い意味の全射的空イベント遷移関数であるものとする。

次の命題は、入出力ペア$(x,y) \in S$に対する状態を決定するために用いる事実である。

命題 1

$S \subseteq X \times Y$を離散事象システムとし、$(x,y) \in S$を任意にとる。このとき正の時刻rとシステムコアの要素$c \in S(\Lambda)$があって、$(\Lambda^r \cdot \sigma^r(x), c^r \cdot \sigma^r(y)) \in S$が成立する。

（証明）
任意の$(x,y) \in S$と$k > 0$をとる。Sが強定常だからある$(\hat{x},\hat{y}) \in S$があって$\lambda^k(\hat{x},\hat{y}) = (x,y)$である。したがって、$(\hat{x}^k \cdot \sigma^k(x), \hat{y}^k \cdot \sigma^k(x)) \in S$が成立する。ここで、$event(\hat{x}) \cap [0,k)$が有限個であることから、$\hat{x}_{uk} = \Lambda_{uk}$となるようなある$u\,(0 < u < k)$がある。$r = k - u$とおくと$(\lambda^u(\hat{x},\hat{y}))_{0r} = (\Lambda_{0r}, \lambda^u(\hat{y}))_{0r}$となる。$S$が$r$から過去決定的であるので、ある要素$y'_r \in Y$があって$(\Lambda, \lambda^u(\hat{y}))_{0r} \cdot y'_r) \in S$となる。つまり、$\lambda^u(\hat{y}))_{0r} \cdot y'_r \in S(\Lambda)$である。$c = \lambda^u(\hat{y}))_{0r} \cdot y'_r$とおくと、$\lambda^u(\hat{x},\hat{y}) = (\Lambda^r \cdot \sigma^r(x), c^r \cdot \sigma^r(y)) \in S$を得る。（証明終）

離散事象システム$S \subseteq X \times Y$に対する状態表現を、状態空間を$S(\Lambda)$にとって構成していく。まず関数$\rho_0 : S(\Lambda) \times X \to Y$を定める。任意の$(c,x) \in S(\Lambda) \times X$に対して、$\rho_0(c,x) = y \Leftrightarrow$
$(\exists k > 0)(\exists z \in S(\Lambda))((\Lambda, z^k \cdot \sigma^k(c)) \in S \wedge (\Lambda^k \cdot \sigma^k(x), z^k \cdot \sigma^k(y)) \in S$

この定義は well-defined である。実際 $(\exists p>0)(\exists \hat{z}\in S(\Lambda))((\Lambda,\hat{z}^p\cdot\sigma^p(c))\in S)$ と仮定するとシステムコア決定性によって、$(\Lambda^p\cdot\sigma^p(x),\hat{z}^p\cdot\sigma^p(y))\in S$ が成立し、したがって、$\rho_0(c,x)=y$ を得る。

命題２

上記のように定義された関数 $\rho_0:S(\Lambda)\times X\to Y$ は $S\subseteq X\times Y$ の初期状態応答である。

（証明）

任意の $c\in S(\Lambda)$ と $x\in X$ をとって $(x,\rho_0(c,x))\in S$ が成立することを示す。命題１より $(\Lambda,c'^r\cdot\sigma^r(c))\in S$ となるような $r>0$ と $c'\in S(\Lambda)$ がある。S が r から過去決定的なので、ある y があって $(\Lambda^r\cdot\sigma^r(x),c'^r\cdot\sigma^r(y))\in S$ が成立する。強定常性から $(x,y)\in\lambda^r(S)=S$ を得る。すると ρ_0 の定義によって $\rho_0(c,x)=y$ を得るので、$(x,\rho_0(c,x))\in S$ となる。

次に、$(x,y)\in S$ を任意に取る。命題１によって正の時刻 r とシステムコアの要素 $c\in S(\Lambda)$ があって $(\Lambda^r\cdot\sigma^r(x),c^r\cdot\sigma^r(y))\in S$ である。したがって ρ_0 の定義によって $\rho_0(\lambda^r(c),x)=y$ を得る。　（証明終）

この命題２の $\rho_0:S(\Lambda)\times X\to Y$ はシステム S に対して唯一定まるので、S の標準初期状態応答と呼ぶ。

命題３

S の標準初期状態応答は次の性質を満たす。

(1) 任意の $c\in S(\Lambda)$ について $\rho_0(c,\Lambda)=c$

(2) ρ_0 は規約である。

(3) ρ_0 は因果的である。

(4) 任意の $c\in S(\Lambda),x\in X,t\in T$ について $\rho_0(c,x)|_{[0,t)}=\rho_0(c,x^t\cdot\Lambda_t)|_{[0,t)}$

（証明）

(1) 任意の $c\in S(\Lambda)$ を取る。$y=\rho_0(c,\Lambda)$ とおく。ρ_0 の定義によって $k>0$ と $z\in S(\Lambda)$ があって、$(\Lambda,z^k\cdot\sigma^k(c))\in S\wedge(\Lambda^k\cdot\sigma^k(\Lambda),z^k\cdot\sigma^k(y))\in S$ である。過去決定性により $c=y$ を得る。

(2) $\rho_0(c',\Lambda)=\rho_0(c,\Lambda)$ と仮定すると、(1)により $c=c'$ となる。

(3) 任意の $c\in S(\Lambda)$ と $x,x'\in X$ をとって $y=\rho_0(c,x)$ とおく。ρ_0 の定義によって $k>0$ と $z\in S(\Lambda)$ があって、$(\Lambda,z^k\cdot\sigma^k(c))\in S\wedge(\Lambda^k\cdot\sigma^k(x),z^k\cdot\sigma^k(y))\in S$ である。$y'=\rho_0(c,x')$ とおくと、同様にして、ある $r>0$ と $z'\in S(\Lambda)$ があって、$(\Lambda,z'^r\cdot\sigma^r(c))\in S\wedge(\Lambda^r\cdot\sigma^r(x'),z'^r\cdot\sigma^r(y'))\in S$ である。すると、システムコア決定性によって $(\Lambda^r\cdot\sigma^r(x),z'^r\cdot\sigma^r(y))\in S$ となる。ここで任意の $t\in T$ について $x|_{[0,t]}=x'|_{[0,t]}$ と仮定すると S が過去決定的であることから $y|_{[0,t]}=y'|_{[0,t]}$ である。つまり、$\rho_0(c,x)|_{[0,t]}=\rho_0(c,x')|_{[0,t]}$ が成立するので因果的であることが分かる。

(4) 任意の $c\in S(\Lambda),x\in X,t\in T$ をとる。いま $\rho_0(c,x)|_{[0,t)}\neq\rho_0(c,x^t\cdot\Lambda_t)|_{[0,t)}$ と仮定

する。したがって、ある時刻 $p\,(0 \le p < t)$において $\rho_0(c,x)(p) \neq \rho_0(c,x^t \cdot \Lambda_t)(p)$ である。しかし、 $x|_{[0,p]} = (x^t \cdot \Lambda_t)|_{[0,p]}$ であるから(3)によってこれは不可能である。つまり、 $\rho_0(c,x)|_{[0,t)} = \rho_0(c,x^t \cdot \Lambda_t)|_{[0,t)}$である。　　（証明終）

次の命題は、離散事象システムの出力に特殊性があって、どんな出力も必ずシステムコアの出力の連接として表現できることを示している。命題 3 とあわせて、状態表現を構成していくときに使われる。

命題４

$S \subseteq X \times Y$を離散事象システムとする。

(1)任意の $c \in S(\Lambda), x \in X$に対して、出力 $\rho_0(c,x) = y$は次のように分割される。

(i) $event(x) = \{t_0, t_1, t_2, ... \mid t_0 < t_1 < t_2 < ...\}$のとき、 $y^{(n)} \equiv \lambda^{t_{n-1}} \rho_0(c, x^{t_n} \cdot \Lambda_{t_n})$, $n \ge 1$として $\rho_0(c,x) = c_{0t_0} \cdot (\sigma^{t_0} y^{(1)})_{t_0 t_1} \cdot (\sigma^{t_1} y^{(2)})_{t_1 t_2} \cdot ...$である。

(ii) $event(x) = \{t_0, t_1, ..., t_{n-1} \mid t_0 < t_1 < ... < t_{n-1}\}$（有限）のとき、 $y^{(p)} \equiv \lambda^{t_{p-1}} \rho_0(c, x^{t_p} \cdot \Lambda_{t_p})$, $n > p \ge 1$, および$y^{(n)} \equiv \lambda^{t_{n-1}} \rho_0(c,x)$として $\rho_0(c,x) = c_{0t_0} \cdot (\sigma^{t_0} y^{(1)})_{t_0 t_1} \cdot ... \cdot (\sigma^{t_{n-1}} y^{(n)})$である。

(2) $event(x)$が有限でも無限でも、それぞれの $p, 1 \le p,$について次が成立する。

$y^{(p)} = \rho_0(\lambda^{t_{p-1} - t_{p-2}} y^{(p-1)}, \lambda^{t_{p-1}}(x^{t_p} \cdot \Lambda_{t_p}))$、ただし、 $t_{0-1} = 0, y^{(0)} = c$と定める。

（証明）

(1) これを示すには(1.1) $\rho_0(c,x)_{[0t_0)} = c^{t_0}$の成立と、(1.2) 各 $n\,(\ge 1)$について $\rho_0(c,x)_{t_{n-1}t_n} = (\sigma^{t_{n-1}} y^{(n)})_{t_{n-1}t_n}$が成立することを示せばよい。

(1.1) 命題３の(4)より $\rho_0(c,x)|_{[0,t_0)} = \rho_0(c, x^{t_0} \cdot \Lambda_{t_0})|_{[0,t_0)} = \rho_0(c,\Lambda)|_{[0,t_0)} = c^{t_0}$となる。

(1.2) 定義から $\sigma^{t_{n-1}} y^{(n)} \equiv \sigma^{t_{n-1}} \lambda^{t_{n-1}} \rho_0(c, x^{t_n} \cdot \Lambda_{t_n})$なので、

$(\sigma^{t_{n-1}} y^{(n)})_{t_{n-1}t_n} \equiv \rho_0(c, x^{t_n} \cdot \Lambda_{t_n})_{t_{n-1}t_n}$である。 ρ_0が因果的であるから、

$\rho_0(c,x)_{0t_n} = \rho_0(c, x^{t_n} \cdot \Lambda_{t_n})_{0t_n}$および、 $\rho_0(c,x)_{t_{n-1}t_n} = \rho_0(c, x^{t_n} \cdot \Lambda_{t_n})_{t_{n-1}t_n}$が成立する。

(2) nについての数学的帰納法によって示す。

任意の $c \in S(\Lambda), x \in X$をとる。

基本ステップ：$n = 1$とする。この場合、$(x^{t_1} \cdot \Lambda_{t_1})_{0t_0} = \Lambda_{0t_0}$となるので、命題３の(4)より、 $c^{t_0} = \rho_0(c,\Lambda)_{0t_0} = \rho_0(c, x^{t_1} \cdot \Lambda_{t_1})_{0t_0}$が成立する。したがって、

$(x^{t_1} \cdot \Lambda_{t_1}, \rho_0(c, x^{t_1} \cdot \Lambda_{t_1})) = (\Lambda^{t_0} \cdot \sigma^{t_0}(\lambda^{t_0}(x^{t_1} \cdot \Lambda_{t_1})), c^{t_0} \cdot [\rho_0(c, x^{t_1} \cdot \Lambda_{t_1})|_{[t_1,\infty)}]$を得る。さらに、$(\Lambda, c) = (\Lambda, c^{t_0} \cdot \sigma^{t_0}(\lambda^{t_0} c)) \in S$であるから、 ρ_0の定義によって、

$\rho_0(\lambda^{t_0} c, \lambda^{t_0}(x^{t_1} \cdot \Lambda_{t_1})) = \sigma^{-t_0}(\rho_0(c, x^{t_1} \cdot \Lambda_{t_1})|_{[t_1,\infty)}) = \lambda^{t_0} \rho_0(c, x^{t_1} \cdot \Lambda_{t_1}) = y^{(1)}$を得る。

帰納ステップ：

任意に $k \ge 1$を固定して、 $y^{(k)} = \rho_0(\lambda^{t_{k-1} - t_{k-2}} y^{(k-1)}, \lambda^{t_{k-1}}(x^{t_k} \cdot \Lambda_{t_k}))$が成立すると仮定する。これが $k+1$においても成立することを示す。

任意の $r\,(t_{k-1} < r < t_k)$をとる。(1) から $y^{(k)} = \lambda^{t_{k-1}} \rho_0(c, x^{t_k} \cdot \Lambda_{t_k})$である。 Sはシステ

ムコアにおける出力表現性を満たすので$y^{(k)} \in S(\Lambda)$である。ここで
$\lambda^{r-t_{k-1}}(\Lambda, y^{(k)}) = (\Lambda, \lambda^{r-t_{k-1}} y^{(k)}) = (\Lambda, [\lambda^{r-t_{k-1}} y^{(k)}]_{0,t_k-r} \cdot \sigma^{t_k-r}(\lambda^{t_k-t_{k-1}} y^{(k)}))$であるので、
(*)　$(\Lambda, [\lambda^{r-t_{k-1}} y^{(k)}]_{0,t_k-r} \cdot \sigma^{t_k-r}(\lambda^{t_k-t_{k-1}} y^{(k)})) \in S$
を得る。

さて、$(x^{t_{k+1}} \cdot \Lambda_{t_{k+1}}, \rho_0(c, x^{t_{k+1}} \cdot \Lambda_{t_{k+1}})) \in S$であるから、定常性によって
$(\lambda^r(x^{t_{k+1}} \cdot \Lambda_{t_{k+1}}), \lambda^r(\rho_0(c, x^{t_{k+1}} \cdot \Lambda_{t_{k+1}}))) \in S$を得る。入力と出力をそれぞれ計算すると、
$\lambda^r(x^{t_{k+1}} \cdot \Lambda_{t_{k+1}}) = \Lambda^{t_k-r} \cdot \sigma^{t_k-r}[\lambda^{t_k}(x^{t_{k+1}} \cdot \Lambda_{t_{k+1}})]$と、
$\lambda^r(\rho_0(c, x^{t_{k+1}} \cdot \Lambda_{t_{k+1}})) = (\lambda^r \sigma^{t_{k-1}} y^{(k)})_{0,t_k-r} \cdot \sigma^{t_k-r} y^{(k+1)} =$
$(\lambda^{r-t_{k-1}} y^{(k)})_{0,t_k-r} \cdot \sigma^{t_k-r} y^{(k+1)}$
となる。したがって、これと(*)式をρ_0の定義に当てはめることによって、
$\rho_0(\lambda^{t_k-t_{k-1}} y^{(k)}, \lambda^{t_k}(x^{t_{k+1}} \cdot \Lambda_{t_{k+1}})) = y^{(k+1)}$
となる。（証明終）

システムコアにおける出力表現性を満たすので、上の命題で$y^{(p)}$はすべてシステムコアの要素である。

離散事象システムの状態表現を構成しよう。
各々の$t,t' \in T, c \in S(\Lambda), x \in X$に対して、関数$\phi_{0t} : S(\Lambda) \times X^t \to S(\Lambda)$を次のように定める。$\phi_{0t}(c, x^t) \equiv \lambda^t \rho_0(c, x^t \cdot \Lambda_t)$，$\phi_{tt'}(c, x_{tt'}) \equiv \phi_{0\tau}(c, \sigma^{-t}(x_{tt'}))$,ただし$\tau = t' - t$，および$\phi_{00}(c, x_{00}) \equiv c$とする。また、関数$\mu : S(\Lambda) \times A \to B$を$\mu(c, a) \equiv \rho_0(c, x)(0)$と定める。ただし、$x$は$x(0) = a$を満たす任意の入力であり、この定義は標準状態応答$\rho_0$の因果性によって well-defined である。
. $\Phi = \{\phi_{tt'} \mid \phi_{tt'} : S(\Lambda) \times X_{tt'} \to S(\Lambda); \forall t, t' \in T, t \le t'\}$とおく。

命題５
$S \subseteq X \times Y$を離散事象システムとする。上記のように定めた関数の集合$<\Phi, \mu>$は、Sの状態表現である。

（証明）
まず、$S = \{(x, y) \mid \exists c \in S(\Lambda), \forall t, y(t) = \mu(\phi_{0t}(c, x^t), x(t))\}$であることを示す。
任意の$c \in S(\Lambda)$、$x \in X$、$t \in T$をとる。$\rho_0(c, x)(t) = \mu(\phi_{0t}(c, x^t), x(t))$が成立することを示せばよい。
いま$event(x) = \{t_0, t_1, t_2, ... \mid 0 \le t_0 < t_1 < t_2 < ...\}$とする。$\rho_0(c, x)$は命題４のようにシステムコアの要素の連接に分割できる。つまり、
$\rho_0(c, x) = c_{0t_0} \cdot (\sigma^{t_0} y^{(1)})_{t_0 t_1} \cdot (\sigma^{t_1} y^{(2)})_{t_1 t_2} \cdot ...$とかける。$t < t_0$の場合には
$\rho_0(c, x^t \cdot \Lambda_t) = \rho_0(c, \Lambda) = c$なので成立する。$t_0 < t$とする。するとある$n \ge 1$があって、$t_{n-1} \le t < t_n$である。
場合１：$t_{n-1} < t < t_n$のとき。
この場合は$x(t) = \Lambda$である。したがって$\mu(\phi_{0t}(c, x^t), x(t)) = \mu(\lambda^t \rho_0(c, x^t \cdot \Lambda_t), \Lambda) =$
$\rho_0(\lambda^t \rho_0(c, x^t \cdot \Lambda_t), \Lambda)(0) = \lambda^t \rho_0(c, x^t \cdot \Lambda_t)(0) = \rho_0(c, x^t \cdot \Lambda_t)(t)$となる。$x(t) = \Lambda$なので$x^t \cdot \Lambda_t \mid_{[0,t]} = x \mid_{[0,t]}$となり、すると、因果性によって$\rho_0(c, x)(t) = \rho_0(c, x^t \cdot \Lambda_t)(t)$と

なる。

場合２：$t_{n-1}=t$のとき。

$x^{t_n}\cdot\Lambda_{t_n}|_{[0,t]}=x|_{[0,t]}$なので

$\rho_0(c,x)(t_{n-1})=\rho_0(c,x^{t_n}\cdot\Lambda_{t_n})(t_{n-1})=\lambda^{t_{n-1}}\rho_0(c,x^{t_n}\cdot\Lambda_{t_n})(0)=y^{(n)}(0)$である。さて、標準状態表現の定義によって$\mu(\phi_{0t}(c,x^t),x(t))=\rho_0(\lambda^{t_{n-1}}\rho_0(c,x^{t_{n-1}}\cdot\Lambda_{t_{n-1}}),x')(0)$、ただし、$x'\in X$は$x'(0)=x(t_{n-1})$であるような任意の入力である。いま$x'=\lambda^{t_{n-1}}(x)$とおく。すると

$\rho_0(\lambda^{t_{n-1}}\rho_0(c,x^{t_{n-1}}\cdot\Lambda_{t_{n-1}}),\lambda^{t_{n-1}}x)=\rho_0(\lambda^{t_{n-1}-t_{n-2}}\lambda^{t_{n-2}}\rho_0(c,x^{t_{n-1}}\cdot\Lambda_{t_{n-1}}),\lambda^{t_{n-1}}x)$

$=\rho_0(\lambda^{t_{n-1}-t_{n-2}}y^{(n-1)},\lambda^{t_{n-1}}x)$となる。命題４の証明で示したように、

$\rho_0(\lambda^{t_{n-1}-t_{n-2}}y^{(n-1)},\lambda^{t_{n-1}}x)=y^{(n)}$である。したがって、

$\mu(\phi_{0t}(c,x^t),x(t))=y^{(n)}(0)=\rho_0(c,x)(t)$を得る。

次に、状態遷移関数族$\Phi=\{\phi_{tt'}\mid\phi_{tt'}:S(\Lambda)\times X_{tt'}\to S(\Lambda);\forall t,t'\in T,t\le t'\}$の半群性を示す。

任意の$c\in S(\Lambda)$、$x\in X$、$t,t'\in T(0<t'<t)$をとる。$p=t-t'$とおく。

$\phi_{0t}(c,x^t)=\phi_{0p}(\phi_{0t'}(c,x^{t'}),\sigma^{-t'}(x_{t't}))$が成立することを示せば充分である。次のように$\hat{c},c',c''\in S(\Lambda)$を定める。

$\hat{c}=\phi_{0t}(c,x^t)=\lambda^t\rho_0(c,x^t\cdot\Lambda_t),$

$c'=\phi_{0t'}(c,x^{t'})=\lambda^{t'}\rho_0(c,x^{t'}\cdot\Lambda_{t'}),$

$c''=\phi_{0p}(c,\sigma^{-t'}(x_{t't}))=\lambda^p\rho_0(c',\sigma^{-t'}(x_{t't})\cdot\Lambda_p).$

ここで、$\sigma^{-t'}(x_{t't})\cdot\Lambda_p=\lambda^{t'}(x^t\cdot\Lambda_t)$に注意して、$c''=\lambda^p\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))$を得る。

いま、あるmがあって$event(x^t\cdot\Lambda_t)=\{t_0,t_1,...,t_{m-1}\}$とかける。命題４によって$\rho_0(c,x^t\cdot\Lambda_t)$をシステムコア分割できるので

$\rho_0(c,x^t\cdot\Lambda_t)=c_{0t_0}\cdot(\sigma^{t_0}y^{(1)})_{t_0t_1}\cdot...\cdot(\sigma^{t_{m-1}}y^{(m)})$

のようにおくことができる。このとき、ある$k\ge0$があって$t_{k-1}<t'\le t_k$となる。ただし$t_{-1}=0$としている。$x^{t'}\cdot\Lambda_{t'}=x^{t_k}\cdot\Lambda_{t_k}$が成立するので、

$c'=\lambda^{t'}\rho_0(c,x^{t'}\cdot\Lambda_{t'})=\lambda^{t'}\rho_0(c,x^t\cdot\Lambda_t)=\lambda^{t'-t_{k-1}}\lambda^{t_{k-1}}\rho_0(c,x^{t_k}\cdot\Lambda_{t_k})=\lambda^{t'-t_{k-1}}y^{(k)}$である。

次に$\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))$をシステムコア分割することができる。入力について$event(\lambda^{t'}(x^t\cdot\Lambda_t))=\{t'_0,t'_1,...,t'_r\}$とすると、

$\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))=c'_{0t'_0}\cdot(\sigma^{t'_0}y'^{(1)})_{t'_0t'_1}\cdot...\cdot(\sigma^{t'_r}y'^{(r+1)})$とかける。ここで、

$event(x^t\cdot\Lambda_t)=\{t_0,t_1,...,t_{m-1}\}$であったので$event(\lambda^{t'}(x^t\cdot\Lambda_t))=\{t_k,t_{k+1},...,t_{m-1}\}$であるから、$t'_0=t_k-t',t'_1=t_{k+1}-t',...,t'_r=t_{m-1}-t'$である。したがって、

$\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))|_{[0,t_k-t')}=c'|_{[0,t_k-t')}=\lambda^{t'-t_{k-1}}y^{(k)}|_{[0,t_k-t')}$となる。よって、

(*)　$(\lambda^{t'}(x^t\cdot\Lambda_t),[\lambda^{t'-t_{k-1}}y^{(k)}]_{0t'_0}\cdot(\sigma^{t'_0}y'^{(1)})_{t'_0t'_1}\cdot...\cdot\sigma^{t'_r}y'^{(r+1)})\in S$

が成立する。また、$\rho_0(c,x^t\cdot\Lambda_t)|_{[t',t_k)}=(\sigma^{t_{k-1}}y^{(k)})|_{[t',t_k)}=(\sigma^{t_{k-1}}y^{(k)})|_{[t',t'+t_0)}$より、

(**)
$$\lambda^{t'}(x^t\cdot\Lambda_t,\rho_0(c,x^t\cdot\Lambda_t))=$$
$$(\lambda^{t'}(x^t\cdot\Lambda_t),[\lambda^{t'-t_{k-1}}y^{(k)}]_{0t'_0}\cdot(\sigma^{t'_0}y^{(k+1)})_{t'_0t'_1}\cdot...\cdot\sigma^{t'_r}y^{(m)})\in S$$

となる。Sが過去決定的であるので、(*)と(**)によって

$y'^{(1)}=y^{(k)},y'^{(2)}=y^{(k+1)},...,y'^{(r+1)}=y^{(m)}$を得る。すなわち、

$\rho_0(c,x^t\cdot\Lambda_t)|_{[t',\infty)}=\sigma^{t'}\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))$が成り立つ。書き換えると
$\lambda^{t'}\rho_0(c,x^t\cdot\Lambda_t)=\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))$となる。以上から、
$c''=\lambda^p\rho_0(c',\lambda^{t'}(x^t\cdot\Lambda_t))=\lambda^p(\lambda^{t'}\rho_0(c,x^t\cdot\Lambda_t))=\lambda^{p+t'}\rho_0(c,x^t\cdot\Lambda_t))=$
$\lambda^t\rho_0(c,x^t\cdot\Lambda_t))=\hat{c}$
となる。すなわち、半群性が成立することが示された。

最後に、$<\Phi,\mu>$が弱い意味の全射的空イベント遷移関数であることを示す。任意の$c\in S(\Lambda)$をとる。命題１からある時刻$r>0$と$z\in S(\Lambda)$があって、$(\Lambda,z^r\cdot\sigma^r(c))\in S$である。よって初期状態応答によって、ある$c'\in S(\Lambda)$があって$\rho_0(c',\Lambda)=z^r\cdot\sigma^r(c)$である。これより$c=\lambda^r\rho_0(c',\Lambda)=\phi_{0t}(c',\Lambda^r)$となり、証明された。

Q.E.D.

命題５の状態表現$<\Phi,\mu>$はシステムSに対して唯一定まるので、Sの標準状態表現と呼ぶ。ここまでのことから、離散事象入出力システムは、つねに標準状態表現を構成することができる。

標準状態表現のユニーク性を示す。そのためには、ひとつの離散事象システムに対して可能な複数の状態表現を比較する方法が必要である。モルフィズムという概念を導入する。

定義８　モルフィズム（Mesarovic and Takahara(Abstract SystemsTheory, 1989)）
$S\subseteq X\times Y$を時間システムとする。Sの２つの状態表現を$<\Phi,\mu>,<\Phi',\mu'>$とし、それぞれの状態空間をC,C'とする。関数$h:C\to C'$が$<\Phi,\mu>$から$<\Phi',\mu'>$へのモルフィズムであるとは、図 9-8 が可換であることである。可換であるとは、任意の $c\in C$，$x\in X$，$t\in T$，$a\in A$について$h(\phi_{0t}(c,x^t))=\phi'_{0t}(h(c),x^t)$および$\mu(c,a)=\mu'(h(c),a)$が成立することである。

さらに、hが全単射のときは同型なモルフィズムといい、また$<\Phi,\mu>$と$<\Phi',\mu'>$は同型であるという。

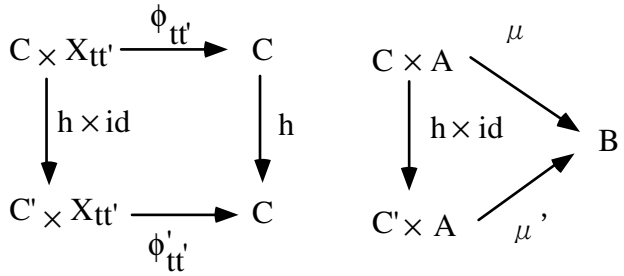


図 9-8. 可換図

関数$h:C\to C'$が$<\Phi,\mu>$から$<\Phi',\mu'>$へのモルフィズムであるとき、$h:<\Phi,\mu>\to<\Phi',\mu'>$とかく。

離散事象システムの標準状態表現を他の状態表現と区別するために、本章では標準状態表現を$<\Phi^*,\mu^*>$とかく。次の定理は標準状態表現が可能な状態表現の中で「最小」なものであることを示す。その意味で無駄のない表現である。

定理 1

$S \subseteq X \times Y$を離散事象システム、$<\Phi,\mu>$をSの状態表現、Cをその状態空間とする。このとき次の(1)と(2)は同値である。

(1) Φは弱い意味の全射的空イベント遷移関数である。

(2) 関数$h : C \to S(\Lambda)$を$h(c) = \rho_0(c,\Lambda)$によって定めると、hは全射の$h :<\Phi,\mu>\to<\Phi^*,\mu^*>$というモルフィズムである。

（証明）

$(1) \to (2)$を示す。

任意の$c \in C, x \in X$をとる。$<\Phi,\mu>$が弱い意味の全射的空イベント遷移関数をもつので、ある時刻$r>0$と$\hat{c} \in S(\Lambda)$があって$c \sim \phi_{0t}(\hat{c},\Lambda^r)$である。

いま、$z = \rho_0(\hat{c},\Lambda)$とおく。任意に$k \in T$をとる。

$$z(r+k) = \mu(\phi_{0,r+k}(\hat{c},\Lambda^{r+k}),\Lambda) =$$

$$\mu(\phi_{0k}(\phi_{0r}(\hat{c},\Lambda^r),\Lambda^k),\Lambda) = \mu(\phi_{0k}(c,\Lambda^k),\Lambda) = \rho_0(c,\Lambda)(k)$$

つまり、$(\Lambda,\ z^r \cdot \sigma^r(\rho_0(c,\Lambda))) \in S$が成立する。

また、$v = \rho_0(\hat{c},\Lambda^r \cdot \sigma^r(x))$とおく。すると、

$$v(r+k) = \mu(\phi_{0,r+k}(\hat{c},(\Lambda^r \cdot \sigma^r(x))^{r+k}),\Lambda^r \cdot \sigma^r(x)(r+k))$$

$$= \mu(\phi_{r,r+k}(\phi_{0r}(\hat{c},\Lambda^r),\sigma^r(x)_{r,r+k}),\ \sigma^r(x)(r+k)) = \mu(\phi_{0k}(\phi_{0r}(\hat{c},\Lambda^r),\ x^k),x(k))$$

$$= \mu(\phi_{0k}(c,x^k),x(k) = \rho_0(c,x)(k)$$

したがって、$z^r = v^r$であることを考慮すると$(\Lambda^r \cdot \sigma^r(x),\ z^r \cdot \sigma^r(\rho_0(c,x))) \in S$が成立する。

以上より、標準初期状態応答ρ_0^*の定義によって$\rho_0^*(\rho_0(c,\Lambda),x) = \rho_0(c,x)$を得る。

$\rho_0^*(\rho_0(c,\Lambda),x) = \rho_0^*(h(c),x)$であるから、$\rho_0^*(\rho_0(c,\Lambda),x^t \cdot \Lambda_t) = \rho_0^*(h(c),x^t \cdot \Lambda_t)$である。したがって、

$$\phi_{0t}^*(h(c),x^t) = \lambda^t \rho_0^*(\rho_0(c,\Lambda),x^t \cdot \Lambda_t)$$

$$= \lambda^t \rho_0^*(h(c),x^t \cdot \Lambda_t) = \lambda^t \rho_0(c,x^t \cdot \Lambda_t)$$

$$= \rho_0(\phi_{0t}(c,x^t),\Lambda) = h(\phi_{0t}(c,x^t))$$

となり、可換図を得る。

次に$\mu(c,a) = \mu^*(h(c),a)$が任意の$c \in C, a \in A$に対して成立することを示す。$x \in X$として$x(0) = a$であるものを任意にとる。$\rho_0^*(\rho_0(c,\Lambda),x) = \rho_0(c,x)$であるから、

$\rho_0^*(\rho_0(c,\Lambda),x)(0)=\rho_0(c,x)(0)$であるが、これは$\mu(c,x(0))=\mu^*(h(c),x(0))$にほかならない。つまり可換図が成立する。

$(2)\rightarrow(1)$を示す。

任意の$c\in C$をとる。図が可換であることから$\rho_0^*(\rho_0(c,\Lambda),\Lambda)=\rho_0(c,\Lambda)$である。$\rho_0^*$の定義によって、ある時刻$r>0$と$z\in S(\Lambda)$があって次が成立する。

(*)　$(\Lambda, z^r\cdot\sigma^r(\rho_0(c,\Lambda)))\in S$

よって、ある$c'\in C$があって$\rho_0(c',\Lambda)=z^r\cdot\sigma^r(\rho_0(c,\Lambda))$とかける。任意に$x\in X,k\in T$について、

$\rho_0(c',\Lambda^r\cdot\sigma^r(x))(r+k)=\mu(\phi_{0,r+k}(c',(\Lambda^r\cdot\sigma^r(x))^{r+k}),(\Lambda^r\cdot\sigma^r(x))(r+k))$

$=\mu(\phi_{0k}(\phi_{0r}(c',\Lambda^r),x^k),x(k))=\rho_0(\phi_{0r}(c',\Lambda^r),x)(k)=\sigma^r(\rho_0(\phi_{0r}(c',\Lambda^r),x))(r+k)$

である。ここでρ_0が因果的であることから、$\rho_0(c',\Lambda^r\cdot\sigma^r(x))_{0r}=z^r$なので次を得る。

(**)　$(\Lambda^r\cdot\sigma^r(x), z^r\cdot\sigma^r(\rho_0(\phi_{0r}(c',\Lambda^r),x)))\in S$

したがって(*)と(**)により$\rho_0^*(\rho_0(c,\Lambda),x)=\rho_0(\phi_{0r}(c',\Lambda^r),x)$。また、可換図より$\rho_0(c,x)=\rho_0^*(\rho_0(c,\Lambda),x)$であるから、$\rho_0(c,x)=\rho_0(\phi_{0r}(c',\Lambda^r),x)$を得る。つまり、$c\sim\phi_{0r}(c',\Lambda^r)$が示された。　Q.E.D.

任意の状態表現が標準状態表現と同型であるためには、状態空間の規約性と関連がある。

系

$S\subseteq X\times Y$を離散事象システム、$<\Phi,\mu>$をSの状態表現であって、弱い意味の全射的空イベント遷移関数を持ち規約であるとする。このとき、$<\Phi,\mu>$は標準状態表現と同型である。

（証明）

関数$h:<\Phi,\mu>\rightarrow<\Phi^*,\mu^*>$が単射であることを示せばよい。$h(c)=h(c')$と仮定する。すると任意の$x\in X$について$\rho_0^*(h(c),x)=\rho_0^*(h(c'),x)$である。図の可換性を用いて、$\rho_0(c,x)=\rho_0(c',x)$となるが、規約であることから$c=c'$を得る。　Q.E.D.

この定理と系から、標準状態表現は最小であり、また、その状態遷移関数は全射的空イベント遷移関数である。

9.4　DEVS のユニーク性と普遍性

第 2 章において、DEVS モデルが有限時間消費条件を満たすとき状態表現を構成し、DEVS の状態システム $S_D = \mathrm{Res}(<\Phi,\lambda>)$ を得ることができた（図 9-9）。ここで DEVS から定められる状態表現 $<\Phi,\lambda>$ を、DEVS 状態表現と呼ぶ。

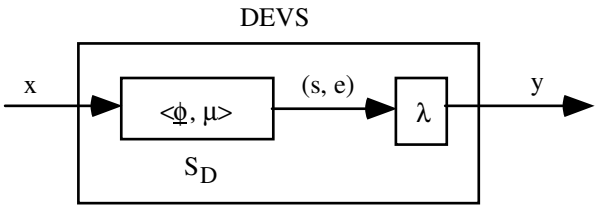


図 9-9. DEVS 離散事象システムの中にある状態システム

以下のいくつかの命題に示すように、DEVS の状態システムが定常であること、任意の正の時刻から過去決定的であること、離散事象入力空間を持つこと、システムコアにおける出力表現性を満たすこと、システムコア決定的であることが示される。状態システムが強定常であるためには、DEVS の内部状態遷移関数 δ_Λ が全射的であることが十分条件となる。

命題６

有限時間消費の DEVS の状態システム $S_D = \mathrm{Res}(<\Phi,\lambda>)$ は、定常性、任意の正の時刻からの過去決定性、離散事象入力空間であること、システムコアにおける出力表現性、システムコア決定性をすべて満たす。

（証明）

まず、S_D が任意の正の時刻から過去決定的であることを示す。任意の正時刻 k をとる。

状態システム S_D は k から有限観測である。実際、$x \in X$ を任意にとり、$y, y' \in S_D(x)$ とする。すると適当な $(s,e),(s',e') \in Q_M$ があって、すべての $t \in T$ について

$$y(t) = \mu(\phi_{0t}(s,e,x^t),x(t)),\ y'(t) = \mu(\phi_{0t}(s',e',x^t),x(t))$$

である。いま $y^k = y'^k$ と仮定すると、当然 $y(0) = y'(0)$ である。したがって

$$y(t) = \mu(\phi_{0t}(s,e,x^t),x(t)) = \mu(\phi_{0t}(y(0),x^t),x(t))$$

$$= \mu(\phi_{0t}(y'(0),x^t),x(t)) = \mu(\phi_{0t}(s',e',x^t),x(t)) = y'(t)$$

となるので、有限観測である。

つぎに S_D が任意の正時刻 k から前因果的であることを示す。$t \geq k$ を任意にとる。$x, x' \in X$ を $x|_{[0,t]} = x'|_{[0,t]}$ を満たす任意の入力とする。任意に $y \in S_D(x)$ をとる。適当な $(s,e) \in Q_M$ があって $y(t) = \mu(\phi_{0t}(s,e,x^t),x(t))$ とかける。このとき、任意の時刻

$r \in T$について$y'(r) = \mu(\phi_{0r}(s,e,x'^r), x'(r))$と定める。$y' \in S_D(x')$である。すると、$x|_{[0,t]} = x'|_{[0,t]}$であることから$y|_{[0,t]} = y'|_{[0,t]}$である。つまり、$S_D(x)|_{[0,t]} \subseteq S_D(x')|_{[0,t]}$が成立する。逆向きの包含関係も同様に成立するので、$S_D(x)|_{[0,t]} = S_D(x')|_{[0,t]}$を得る。つまり$k$から前因果的である。命題１によって、有限観測であり、かつ前因果的であるので、S_Dはkから過去決定的である。

状態システムS_Dが定常であること示す。つまり、任意の時刻$t \in T$について$\lambda^t(S_D) \subseteq S_D$が成立することを示す。任意の$t \in T$と$(x,y) \in S_D$をとる。適当な$(s,e) \in Q_M$によって$y(t) = \mu(\phi_{0t}(s,e,x^t), x(t))$とかける。$(s',e') = \phi_{0t}(s,e,x^t)$とおく。また、各$p \in T$に対して$y'(p) = \mu(\phi_{0p}(s',e',\lambda^t(x)_{0p}), \lambda^t(x)(p))$と定める。状態システムの定義から$(\lambda^t(x), y') \in S_D$である。

$\phi_{0p}(s',e',\lambda^t(x)_{0p}) = \phi_{0p}(\phi_{0t}(s,e,x^t), \lambda^t(x)_{0p})$

$= \phi_{t,t+p}(\phi_{0t}(s,e,x^t), x_{t,t+p})$　（時間不変性より）

$= \phi_{0,t+p}(s,e,x^{t+p})$　（半群性より）

によって、$y'(p) = \mu(\phi_{0,t+p}(s,e,x^{t+p}), x(t+p)) = y(t+p) = \lambda^t y(p)$、つまり$\lambda^t(y) = y'$である。よって、$(\lambda^t(x), \lambda^t(y)) \in S_D$を得るので定常である。

システムコア決定を満たすことを示す。２つの正の時刻$k,r \in T$を任意にとり固定する。任意に$(\Lambda^r, y^r) \in S_D^r, (\Lambda^k, \hat{y}^k) \in S_D^k, c \in S_D(\Lambda), x' \in X, y' \in Y$をとる。さらに、$(\Lambda, y^r \cdot \sigma^r(c)) \in S_D, (\Lambda, \hat{y}^k \cdot \sigma^k(c)) \in S_D$と$(\Lambda^k \cdot \sigma^k(x'), \hat{y}^k \cdot \sigma^k(y')) \in S_D$を仮定する。示すべきことは$(\Lambda^r \cdot \sigma^r(x'), y^r \cdot \sigma^r(y')) \in S_D$である。

状態システムS_Dの初期状態応答$\rho_0 : Q_M \times X \to Y$は$S_D = \mathrm{Res}(<\Phi, \lambda>)$であるから、各$t \in T, (s,e) \in Q_M, x \in X$について$\rho_0(s,e,x)(t) = \mu(\phi_{0t}(s,e,x^t), x(t))$が成り立つ。$(\Lambda^k \cdot \sigma^k(x'), \hat{y}^k \cdot \sigma^k(y')) \in S_D$であるから、ある$(\hat{s},\hat{e}) \in Q_M$があって、

$\rho_0(\hat{s},\hat{e},\Lambda^k \cdot \sigma^k(x'))(t) = \mu(\phi_{0t}(\hat{s},\hat{e},[\Lambda^k \cdot \sigma^k(x')]_{0t}), [\Lambda^k \cdot \sigma^k(x')](t)) = [\hat{y}^k \cdot \sigma^k(y')](t)$

である。いま$c' = \rho_0(\hat{s},\hat{e},\Lambda)$とおくと、初期状態応答の因果性によって$c'^k = \hat{y}^k$を得る。$c'' = \lambda^k(c')$とおくと、$(\Lambda, \hat{y}^k \cdot \sigma^k(c'')) \in S_D$となる。$S_D$の過去決定性によって$c'' = c$であり、したがって$c(0) = c''(0) = \phi_{0k}(\hat{s},\hat{e},\Lambda^k)$である。

仮定により$(\Lambda, y^r \cdot \sigma^r(c)) \in S_D$であるから、ある$(s',e') \in Q_M$があって、

$y^r \cdot \sigma^r(c)(t) = \mu(\phi_{0t}(s',e',\Lambda^t), \Lambda)$であり、$c(0) = \phi_{0r}(s',e',\Lambda^r)$となる。したがって、

$\phi_{0r}(s',e',\Lambda^r) = \phi_{0k}(\hat{s},\hat{e},\Lambda^k)$が成立する。

$\rho_0(s',e',\Lambda^r \cdot \sigma^r(x')) = y^r \cdot \sigma^r(y')$であることを示す。任意の$t \in T$について

$\rho_0(s',e',\Lambda^r \cdot \sigma^r(x'))(r+t)$

$= \mu(\phi_{0,r+t}(s',e',[\Lambda^r \cdot \sigma^r(x')]_{0,r+t}), [\Lambda^r \cdot \sigma^r(x')](r+t))$

$= \mu(\phi_{0t}(\phi_{0r}(s',e',\Lambda^r), x'^t), x'(t))$

$= \mu(\phi_{0t}(\phi_{0k}(\hat{s},\hat{e},\Lambda^k), x'^t), x'(t))$

$=\mu(\phi_{0,k+t}(\hat{s},\hat{e},[\Lambda^k\cdot\sigma^k(x')]_{0,k+t}),[\Lambda^k\cdot\sigma^k(x')](k+t))$

$=[\hat{y}^k\cdot\sigma^k(y')](k+t)=y'(t)$

$=\sigma^r(y')(r+t)$

また、$t<r$である任意のtに対して

$\rho_0(s',e',\Lambda^r\cdot\sigma^r(x'))(t)=\mu(\phi_{0t}(s',e',[\Lambda^r\cdot\sigma^r(x')]_{0t}),[\Lambda^r\cdot\sigma^r(x')](t))$

$=\mu(\phi_{0t}(s',e',\Lambda^t),\Lambda)$

$=y^r\cdot\sigma^r(c)(t)=y(t)$

したがって、$\rho_0(s',e',\Lambda^r\cdot\sigma^r(x'))=y^r\cdot\sigma^r(y')$である。したがってシステムコア決定性のための目標であった$(\Lambda^r\cdot\sigma^r(x'),y^r\cdot\sigma^r(y'))\in S_D$が示された。

最後に、S_Dがシステムコアにおける出力表現性を満たすことを示す。任意の$(x,y)\in S_D$をとる。xの mls 分割を$x^{(0)}x^{(1)}\cdots$とする。もし$x^{(0)}\in G_\Lambda$なら、証明は終了する。$x^{(0)}\in G_{A_M}$と仮定する。するとある$t\in T$があって、$x^{(0)}=x^t$である。すると、$S_D=\mathrm{Res}(<\Phi,\lambda>)$という定義によって、任意の$k(<t)$について、$y(k)=\mu(\phi_{0k}(s,e,x^k),\Lambda)=\phi_{0k}(s,e,x^k)=\delta_G(\mu(s,e,x(0)),\Lambda^k)$である。ここで$z\in S_D(\Lambda)$を任意の$t'\in T$に対して

$z(t')=\phi_{0t'}(\mu(s,e,x(0),\Lambda^{t'})$

と定める。当然、$z^t=y^t$である。以上によってシステムコアにおける出力表現性を満たすことが示された。 Q.E.D.

命題 7 （Zeigler, 1976)

有限時間消費の DEVS の状態システムS_Dの状態遷移関数が強い意味の全射的空イベント遷移関数であるとする。その時、S_Dは強定常である。

（証明）

任意に$t\in T$をとる。$S_D\subseteq\lambda^t(S_D)$を示せば十分である。任意の$(x,y)\in S_D$をとる。ある$c\in Q_M$があって$y=\rho_0(c,x)$である。

$<\Phi,\lambda>$が強い意味の全射的空イベント遷移関数をもつので、ある$c'\in Q_M$があって$c=\phi_{0t}(c',\Lambda^t)$である。$z=\rho_0(c',\Lambda)$とおく。すると$\rho_0(c',\Lambda^t\cdot\sigma^t(x))=z^t\cdot\sigma^t(y)$であることを以下のように示せる。$\rho_0(c',\Lambda^t\cdot\sigma^t(x))_{0t}=z^t$であることは、$\rho_0$が因果的であることからわかる。また、任意の$k\in T$に対して、

$\rho_0(c',\Lambda^t\cdot\sigma^t(x))(t+k)$

$=\mu(\phi_{0,t+k}(c',[\Lambda^t\cdot\sigma^t(x)]_{0,t+k}),[\Lambda^t\cdot\sigma^t(x)](t+k))$

$=\mu(\phi_{t,t+k}(\phi_{0t}(c',\Lambda^t),\sigma^t(x)_{t,t+k}),x(k))$

$=\mu(\phi_{0k}(c,x_{0k}),x(k))$

$=\rho_0(c,x)(k)=y(k)=(\sigma^t y)(t+k)$

したがって、$\rho_0(c',\Lambda^t\cdot\sigma^t(x))=z^t\cdot\sigma^t(y)$を得たので、

$(\Lambda^t \cdot \sigma^t(x), z^t \cdot \sigma^t(y)) \in S_D$である。 $\lambda^t(\Lambda^t \cdot \sigma^t(x), z^t \cdot \sigma^t(y)) = (x,y) \in S_D$であるから、強定常である。 Q.E.D.

命題８

有限時間消費の DEVS の状態システム $S_D = \mathrm{Res}(<\Phi, \lambda>)$において、状態遷移関数が強い意味の全射的空イベント遷移関数であることと、DEVS の内部状態遷移関数δ_Λが全射的であることは等価である。

（証明）

まず、証明にテクニカルに必要な次の補題２を示す。

補題２

Mを有限時間消費の DEVS とする。δ_Λが全射的であること、つぎの命題が成立することは同値である：

$(\forall t \in T)(\forall s \in S_M)(\exists s' \in S_M)(\exists n > 0)$

$(s = \delta(s', n) \wedge \sum(s, n-1) \le t \wedge \sum(s, n) > t).$

（証明）

内部遷移関数δ_Λが全射的であることを仮定して、命題が成立することを示す。任意に$t \in T, s \in S_M$をとる。δ_Λが全射的であるから、ある$s_1 \in S_M$があって

$s = \delta_\Lambda(s_1) = \delta(s_1, 1)$である。

もし、$\sum(s_1, 1) = ta(s_1) > t$ならば証明は完了する。$\sum(s_1, 1) \le t$とすると、ある$s_2 \in S_M$があって$s_1 = \delta_\Lambda(s_2)$である。$s = \delta(s_2, 2)$となることに注意。ここでもし、$\sum(s_2, 2) = ta(s_1) + ta(s_2) > t$ならば証明は完了する。$\sum(s_2, 2) \le t$とすると、ある$s_3 \in S_M$があって$s_2 = \delta_\Lambda(s_3)$および$s = \delta(s_3, 3)$である。この手続きを続けていくことによって、$M$の有限時間消費条件から、必ずある$n$と$s_n \in S_M$があって、

$s = \delta(s_n, n), \sum(s_n, n-1) \le t, \sum(s_n, n) > t$が成立する。（補題 2 の証明終）

（命題 8 の証明）

内部遷移関数δ_Λが全射的であると仮定する。$(s,e) \in Q_M, t \in T$を任意にとる。

もし$e \ge t$の場合には$c' = (s, e-t)$とおく。すると$\phi_{0t}(c', \Lambda^t) = \delta_G(s, e-t, \Lambda^t) = (s,e) = c$より成立する。

もし$e < t$の場合には補題２を用いて、あるnと$s' \in S_M$があって、

$s = \delta(s', n), \sum(s', n-1) \le t-e, \sum(s', n) > t-e$が成立する。ここで、

$r = \sum(s', n), c' = (s', r-(t-e))$

とおく。このとき、

$$\phi_{0t}(c',\Lambda^t)=\delta_G(\mu(s',r-(t-e),\Lambda),\Lambda^t)$$
$$=\delta_G(\delta(s',1),0,\Lambda^{t+r-(t-e)-ta(s')})$$
$$=\ldots=\delta_G(\delta(s',n),0,\Lambda^{\sum(s',n)+e-\sum(s',n)})$$
$$=\delta_G(s,0,\Lambda^e)=(s,e)=c$$

が成立する。つまり、Mの有限時間消費であるとき$S_D=\mathrm{Res}(<\Phi,\lambda>)$は強い意味の全射的空イベント遷移関数を持つ。

つぎに、有限時間消費のMの状態システムの状態遷移関数$<\Phi,\lambda>$が強い意味の全射的空イベント遷移関数を持つ場合には、内部遷移関数δ_Λが全射的であることを示す。対偶により示すため、δ_Λが全射的でないと仮定する。すると補題２によって、ある$(s,e)\in Q_M$と時刻$t\in T$があって、どんなnと$s'\in S_M$をとっても必ず$s\neq\delta(s',n)$かまたは$\sum(s',n)\le t$となる。有限時間消費なのですべてのnについて$\sum(s',n)\le t$ということはないから、$s\neq\delta(s',n)$ということである。今、$(s,0)=c$とおく。任意の$s'\in S_M$と$e'(0\le e'\le ta(s'))$について、$s\neq\delta(s',n)$であることから、

$$\phi_{0t}(s',e',\Lambda^t)=\delta_G(\mu(s',e',\Lambda),\Lambda^t)\neq(s,0)=c$$

である。つまり、強い意味の全射的空イベント遷移関数ではない。　　Q.E.D.

さて、入力を持たないDEVSをオートノマスであるという。オートノマスな場合には、内部状態遷移関数δ_Λが全射的であることはかなり一般的に成り立つと想像されている。その理由は次の通りである。

離散事象システムをDEVSによってモデル化する場合、業務取引システムのように、外部入力はジェネレータ DEVS としてモデル化することがほとんどである。したがって結合DEVSシステムとして得られた離散事象システムでは、状態遷移表によって状態遷移が表現できる。初期値が変わることでさまざまな状態遷移表が得られる。それらをすべてまとめたものが一つの状態遷移関数である。結合DEVSシステムの状態遷移表では、ひとつの行はひとつの状態であり、次の行への変更が、内部状態遷移関数δ_Λである。したがって、δ_Λが全射的であるということは、任意の行が与えられた時にその行に遷移する前の行が状態遷移表にあるかどうか、あるいは、初期値として与えられるかどうかである。これは状態遷移表の各活動の記述が「理にかなったもの」であれば、当然成り立っている。理にかなっていることの表現が、状態遷移表において各活動の状態(s,e)において、逐次状態と経過時間とのペアが満たすべき条件$0\le e\le ta(s)$である。したがって、結合DEVSシステムの場合には、初期条件すべてに対して、この条件を課すことによって、内部状態遷移関数δ_Λは全射的となる。

DEVSモデルのユニーク性を考えよう。DEVSが定める状態システムS_Dの状態表現は、S_Dに対して可能な状態表現の中で最小であるという意味でユニークである。つまり次の定理が成り立つ。

定理２

有限時間消費の DEVS の状態システムをS_Dとする。もし DEVS 状態表現$<\Phi,\lambda>$が強い意味の全射的空イベント状態遷移で規約ならば、$<\Phi,\lambda>$はS_Dの標準状態表現と同型である。

この定理は、まず DEVS モデルがあるときにそれが定める離散事象入出力システムが定まり、そのシステムの状態表現クラスの中での DEVS 状態表現の位置付けをしたものである

今度はこの状況を逆から考える。任意の離散事象入出力システム$S \subseteq X \times Y$が与えられたとき、その DEVS モデルを構成することである。これは常に可能である。この意味で、DEVS は普遍的である、または、普遍的なモデル化方法論であるといえるのである。

実際$S \subseteq X \times Y$に対する DEVS を以下のように定めることができる。入力値集合をA、出力値集合をBとする。標準状態表現は$<\Phi^*,\mu^*>$である。このとき、Sを生成する DEVS　$M =< A_M, C_M, B_M, \delta_M, \lambda_M, ta >$　を定義していく。

$A_M = A, C_M = S(\Lambda), B_M = B$と定める。時間進め関数$ta : C_M \to T^\infty$として、任意の$y \in S(\Lambda)$について次で定める。

$$ta(y) = \begin{cases} min\ \Gamma(y), \text{if } \Gamma(y) \neq \phi \\ \infty, \text{if } \Gamma(y) = \phi \end{cases}$$

内部状態遷移関数$\delta_\Lambda : C_M \to C_M$として、任意の$y \in S(\Lambda)$について次で定める。

$$\delta_\Lambda(y) = \begin{cases} \lambda^r(y), \text{if } r = ta(y) < \infty \\ y, \text{if } ta(y) = \infty \end{cases}$$

$Q_M = \{(c,e) \mid c \in C_M, 0 \le e \le ta(c)\}$と定める。

疑似状態遷移関数$\delta_M : Q_M \times (A_M \cup \{\Lambda\}) \to C_M$として、任意の$(c,e) \in Q_M$について$\delta_M(c,e,\Lambda) = \delta_\Lambda(\lambda^e c)$とし、また、任意の$a \in A_M$について

$\delta_M(c,e,a) = \rho_0^*(\lambda^e c, a_\infty)$と定める。

出力関数$\lambda_M : Q_M \to B_M$として、任意の$(c,e) \in Q_M$について$\lambda_M(c,e) = c(e)$と定める。

上で定めた DEVS Mは有限時間消費である。実際、任意の$c \in S(\Lambda)$をとる。$\Gamma(c)$が有限個と仮定する。するとtaの定義からある正整数pがあって$ta(\lambda_\Lambda{}^p(c)) = \infty$となるので有限時間消費である。$\Gamma(c)$が無限個と仮定し、かつ、$M$が有限消費時間ではないとする。すると、ある時間区間$[t,t')$が存在して$\Gamma(c) \cap [t,t')$が無限個の要素を含む。しかし、これはSの出力空間が離散事象出力空間であることに矛盾する。以上から、DEVS Mは有限時間消費である。

Mが有限時間消費な DEVS なので、Mの状態システムを構成できる。これをD_Mと書くことにする。D_Mの状態表現である DEVS 状態表現を$<\Phi,\mu>$とするときD_Mは次を満たしている。

$(x,z) \in D_M \subseteq X \times (Q_M)^T$であるとは、ある$(c,e) \in Q_M$があって任意の$t \in T$において$z(t) = \mu(\phi_{0t}(c,e,x^t),x(t))$となっていること。

離散事象システムSの状態表現は、D_Mの状態表現$<\Phi,\mu>$を用いて$<\Phi,\lambda_M \circ \mu>$となる。$<\Phi,\lambda_M \circ \mu>$を$S$の DEVS 実現と呼ぶ。したがって、$S$の標準状態表現を$<\Phi^*,\mu^*>$とするとき次が成立する：すべての$(c,e) \in Q_M$と$x \in X, t \in T$に対して、

$$\mu^*(\phi^*{}_{0t}(c,e,x^t),x(t)) = \lambda_M(\mu(\phi_{0t}(c,e,x^t),x(t)))$$

この等号を証明することはなんら困難はなく機械的で退屈なので省略する。

さて、以上で定義してきた離散事象入出力システムSの標準状態表現と DEVS 実現の関係は図 9-10 のようになる。メサロビッチ&高原（1975）にもあるように任意の状態表現から等価で規約な状態表現を得ることができる。DEVS 実現$<\Phi,\lambda_M \circ \mu>$はひとつの状態表現でありから、やはりそれを規約な状態表現にすることができるので、図 9-10 では$<\Phi,\lambda_M \circ \mu>$を規約化したものを$<\hat{\Phi},\hat{\lambda}>$と書いている。標準状態表現$<\Phi^*,\mu^*>$と規約化された DEVS 実現$<\hat{\Phi},\hat{\lambda}>$が同型となる。

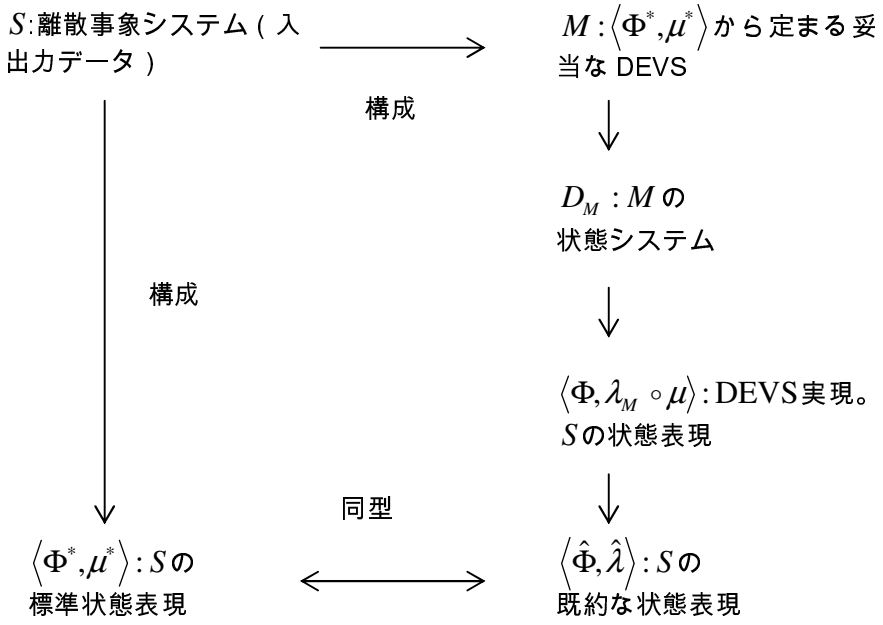


図 9-10. DEVS 実現

定理３（DEVS のユニーク性と普遍性）

任意の離散事象入出力システム $S \subseteq X \times Y$ を考える。

（１）DEVS の普遍性：Sの DEVS 実現が存在する。つまり、任意の離散事象システムは DEVS として実現可能である。

（２）DEVS のユニーク性：Sの規約化した DEVS 実現は、Sの標準状態表現と同型である。この意味において、任意の離散事象システムは最小の DEVS 実現を持つ。

（証明）

（１）について：これは明らかである。というのは、任意の $(c,e) \in Q_M, x \in X, t \in T$ に対して、$\mu^*(\phi^*_{0t}(\lambda^e c, x^t), x(t)) = \lambda_M(\mu(\phi_{0t}(c,e,x^t), x(t)))$ であるから。

（２）について：

任意の離散事象システム $S \subseteq X \times Y$ をとる。$\langle \Phi, \lambda_M \circ \mu \rangle$ を Sの DEVS 実現とし、$<\Phi^*, \mu^*>$ を Sの標準状態表現とする。定理１と系によって、$\langle \Phi, \lambda_M \circ \mu \rangle$ と $<\Phi^*, \mu^*>$ が可換で、かつ、関数 $h : Q_M \to S(\Lambda), h(c,e) = \rho_0(c,e,\Lambda)$ が全写的であればよい。ここで、ρ_0は DEVS 実現から定まる Sの初期状態応答関数であって、状態表現から自然に定められる関数である。つまり、任意の $(c,e) \in Q_M, x \in X, t \in T$ に対して $\rho_0(c,e,x)(t) = \lambda_M(\mu(\phi_{0t}(c,e,x^t), x(t)))$ によって定義されている。

任意の $(c,e) \in Q_M$ に対して $h(c,e) = \rho_0(c,e,\Lambda) = \lambda^e(c)$ であることに注意する。任意に $k,t \in T$ と $(c,e) \in Q_M$、$x \in X$ をとる。このとき次が成立する。

$$h(\phi_{0t}(c,e,x^t))(k) = \rho_0(\phi_{0t}(c,e,x^t), \Lambda)(k)$$

$$= \lambda_M \circ \mu(\phi_{0k}(\phi_{0t}(c,e,x^t), \Lambda^k), \Lambda) = \lambda_M \circ \mu(\phi_{0,t+k}(c,e,x^t \cdot \Lambda_{t,t+k}), \Lambda)$$

$$= \mu^*(\phi^*_{0,t+k}(\lambda^e c, x^t \cdot \Lambda_{t,t+k}), \Lambda) = \mu^*(\phi^*_{0k}(\phi^*_{0t}(\lambda^e c, x^t), \Lambda^k), \Lambda)$$

$$= \mu^*(\lambda^k(\phi^*_{0t}(\lambda^e c, x^t), \Lambda) = \lambda^k(\phi^*_{0t}(\lambda^e c, x^t)(0) = \phi^*_{0t}(\lambda^e c, x^t)(k).$$

したがって、$h(\phi_{0t}(c,e,x^t) = \phi^*_{0t}(\lambda^e c, x^t) = \phi^*_{0t}(h(c,e), x^t)$ であり、状態遷移関数の可換性が示された。

次に、すべての $(c,e) \in Q_M$ と $a \in A$ について $\lambda_M \circ \mu(c,e,a) = \mu^*(h(c,e),a)$ という可換性が成立することは、（１）の式において、$t = 0, x(0) = a$ とおくと得られる。

最後に、関数 hが全射的であることを示す。任意の $y \in S(\Lambda)$ をとる。命題１によって、ある $r > 0$ と $c \in S(\Lambda)$ があって $(\Lambda, c^r \cdot \sigma^r(y)) \in S$ である。ここで $r > e > 0$ であるような eをとると、$h(\lambda^{r-e}(c^r \cdot \sigma^r(y)), e) = \lambda^r(c^r \cdot \sigma^r(y)) = y$ である。つまり、hが全射的である。

Q.E.D.

上の定理の意味を言い換えると次のようになる。

（1）離散事象システムを扱う際には、DEVS で動的構造をモデル化しておいてそれをエンジニアリングの対象にすればよい。なぜなら、DEVS モデル化は普遍的なので、相手が離散事象システムであれば必ず DEVS 表現が存在するからである。たとえば、ビジネスプロセスの離散事象的側面をとらえるために、業務取引システムという結合 DEVS システムを構成できることは、4 章やその後の章で見てきた通りである。

（2）標準状態表現は最小実現であるが、実際に設計を行う際には、標準状態表現と同型な具体的なモデルが必要となる。DEVS の最小性は、たとえば離散事象システムの例であるビジネスプロセスの状態表現の場合、ファイルの個数や業務活動の数と関連することが分かっているので（4 章）、それらに注目して実践して行けばよいことがわかる。直感によってプロセスを改善する場合には、雑多なものが思い浮かぶが、DEVS に基づいた状態表現を使うことで、何がプロセスの動きを決めているかがわかるため、改善や新規設計の場合の概念的基盤となるのである。

あとがき

ビジネスプロセス工学は優れて社会システムの構築に関わる工学領域である。工学や数学を勉強すると、当たり前に見える定義を使って、いろんな深い定理を導いている。「定理を理解して用いることで、分析や設計において定理が威力を発揮する。設計理論にとって重要なのは定理であって、定義はそのためのおまけのようなものである。」このイメージは間違ってはいないが、しかし、工学や理論を「作る」ほうの立場からするとずいぶんちがう。最重要なのは定義である。定理ではない。定義が先で定理は後なのである。応用領域や適用範囲は定義が決める。存在の本質的なところを見抜いたような、うまい発見をともなった定義は、ほぼ自然に定理を導く力を宿している。多少なりとも新しい分野の発展段階では、非常に凝ってみたり、さまざまに広範な可能性を試しながら、最後には実に「あたりまえ」に見えるような定義にたどり着くことが多い。

ビジネスプロセスや、関連する情報システムの領域でも事情は同じである。ビジネス実務では、販売や開発や生産、購買や物流、財務や設備投資などについて、いろいろな作業標準や帳票、計画・管理の方式が用いられている。これらは、ビジネスプロセスの定義にほかならない。それらの定義は表現形式の論理構造と伝票や集計表やデータベースでの実現による操作方法を持つことで、ビジネスプロセスの中に取り入れられて、作業者や IEer やシステムエンジニアやコンサルタントが使うと同時に磨き上げてきた。つまり、これまでのビジネスの現場において「うまい発見をともなった定義」が既に数多く考えだされて来たのである。非常な知的エネルギーが注がれてきたことが理解できる。こうしたビジネス定義が導くはずの「定理」は、しかし、暗黙的でありその原理や意味が理論的には明示化されてこなかったといえる。

この世のすべての物的存在に対して等しく作用する万有引力や、力と重さと加速度の関係は、ニュートン以前にも日常生活の中で経験的に理解されていただろう。たとえば、物を強く投げるとより速く飛んで行くことは、体を通して誰でも知っている。しかし、それに適切な表現を与えると、論理的な計算ができるようになる。しまいにはロケットで人間を月に届けられるようになる。さらに電磁気の現象にも同様の表現を与えることも行われ、モーターや通信を可能にした。微分方程式として抽象化されると、微分方程式自体が存在感を持つので（つまり微分方程式にまつわる論理構造をいろいろと検討できるようになるので)、物的世界から離れて研究することが可能になり、さらにそれが物的世界の深い理解や新しい利用法や産業・ビジネスにつながって行く。

ビジネスプロセス工学は、偉大な運動方程式との荒唐無稽的アナロジーからすれば、まったく未発達といえよう。第２次世界大戦後 1950 年代からコンピュ

ータを産業に利用し始めて以来、叡智を集めて、情報システムがらみのビジネスプロセスの構造の記述方法を試して検討している段階なのである。

いろいろに提案された記述方法の背後にある論理構造に表現を与え動的特性を計算できると、もっと可能性が開けるのではないか。

本書で提案したのは、第１に、ビジネスプロセスが離散事象システムであることだ。また、データモデルで定義されたファイルシステムがビジネスプロセスの状態の一部を決定することである。このような定義が、工学の歴史からみて微分方程式や差分方程式による時間変化の認識と整合することは、一般システム理論で開発されていた状態遷移メカニズムの一般的論理構造と整合的であることを論拠にしている。おそらくビジネスプロセスの場合の万有引力の法則は、リトルの定理ではないだろうか。この見方は Hopp and Spearman(2001)と軌を一にする。さらに本書は、離散事象システムの有力モデル方法論の一つである時間ペトリネットをビジネスプロセス用に特化させて業務取引ペトリネットとすることが可能であり、その結果、将来のビジネスプロセス方程式につながるような方程式表現をほのめかしたものである。

このように、本書のねらいは（尊大ではなく）遠大なものなのだが、設計論としての完成形にはほど遠いことも認めなければならない。ビジネスプロセス方程式のきっかけとなるものを示した段階である。また、ビジネスプロセスには計画システムが生産や販売、仕入れや倉庫管理などでひろく使われており、RFID・IC タグはいうまでもなく、トータルビジネスプロセスとして効果が上がるならタイマーと演算回路も入れて RFIS（無線情報システム）も可能であろう。このような将来的デバイスも取り入れて、ビジネスプロセスのイノベーションやビジネスプロセスとしてのサービス連携の仕組みの改善を続けて行くためには、実践だけでなく、計画・再計画機能と動特性との関連を計算可能な方程式で表す展開も必要である。ビジネスプロセス工学のこうした領域が我々の実践と発見を待っているのである。

参考**文献リスト**

Asahi, T. and Zeigler ,B.P. (1993), "Behavioral Characterization of Discrete Event Systems." *Proceedings of AISP'93: 4th. Annual Conference on AI, Simulation and Planning*, IEEE computer society, pp.127--132.

Baccelli, F.L., Cohen, G., Olsder, G.J., and Quadrat, JP, *Synchronization and Linearity -- An algebra for discrete event systems*, John Wiley, 1992.

チェックランド、ピーター：新しいシステムアプローチ：システム思考とシステム実践、オーム社、1985.

Chen, Peter Pin-Shan, "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transactions on Database Systems*, 1-1, pp 9-36, 1976.

CIM 研究グループ、生産革命 CIM-構築のアプローチ、工業調査会、1988.

Curran, T. and Keller G., *SAP R/3 business blueprint*, Prentice-Hall, 1998.

Cutts, G. （浦昭二監訳）、情報システムの分析と設計、培風館, 1995.

DeMarco, T. , “Structured Analysis and System Specification,” Prentice-Hall, 1979. （高梨, 黒田監訳,『構造化分析とシステム仕様』, 日経 BP 社, 1986）

Dewiz, S. D., *Systems analysis and design and the transition to objects*, McGraw-Hill, 1996.

ダウンズ, E., クレア, P., コー, I（伊藤武夫訳）、 構造化システム分析と設計技法 SSADM - その適用と状況について - 、近代科学社、1991

ゴールドラット(1992), 三本木訳、ザ・ゴール ― 企業の究極の目的とは何か, ダイヤモンド社, 2001.

長谷川健介, "ペトリネットの生産システムへの応用,計測と制御, 28-9, pp.775-783, 1989.

Hopp,W.J. and Spearman, M.L., *Factory physics : foundations of manufacturing management*, Irwin/McGraw-Hill, 2001

Ho, Y.-C. (1989), Special issue on dynamics of discrete event systems. Proceedings of the IEEE, 77(1).

穂鷹良介、データベースシステムとデータモデル、オーム社、1989

サイモン、ハーバート（稲葉、吉原訳）：システムの科学、パーソナルメディア、1999.

ジョンソン, H.T.: 米国製造業の復活 : 「トップダウン・コントロール」から「ボトムアップ・エンパワメント」へ、辻・河田訳、中央経済社、 1994

Keller, G. and Teufel, T: *R/3 Process Oriented Implementation: Iterative Process Prototypin*g, Addison-Wesley, 1998（田熊博志訳、SAPR/3:プロセス指向型の ERP 導入、ピアソンエデュケーション、2000）

Little, J.D.C., "A proof for the queuing formula: $L = \lambda W$," *Operations Research*, 9, 383-387, 1961

McDermid, D.C. (1990): Software Engineering for Information Systems, Blackwell Scientific Publications.

Mesarovic, M. D. and Takahara, Y. , *Mathematical foundation of general systems theory*. Academic, 1975.

Mesarovic, M. D. and Takahara, Y. , *Abstract systems theory*. (Lecture Notes in Control and Information Science 116), Springer, 1989.

三菱総合研究所、企業間電子商取引事例等に関する調査研究報告書 - 企業間の情報連携・協調実現のための EDI の導入・利用拡大に向けて、三菱総合研究所、2005.

モデル, M.E.、荒川・岡野・井上訳、第 1 線技術者のための実践システム分析、マグロウヒル出版、1990.

Monden, Y. , *Toyota Production System - Practical approach to production management.* Industrial Engineering and Management Press, Institute of Industrial Engineers, 1983.

門田安弘,『新トヨタシステム』, 講談社, 1991.

門田安弘,「トヨタプロダクションシステム-その理論と体系」、ダイヤモンド社、2006.

村田忠夫,『ペトリネットの解析と応用』, 近代科学社 1992.

中根甚一郎：総合化 MRP システム, 日刊工業新聞社, 1984.

ピーターズ, L (1987) : 実践的構造化分析と設計（白井、奥島訳）、近代科学社、1990

Peterson, J.L., *Petri net theory and the modeling of systems*, Prentice-Hall, 1981.

Pickle H. and Abrahamson R. , *Small business management* , Wiley, 1990.

ポーター, マイケル ; 竹内弘高訳、競争戦略論,ダイヤモンド社,1999.

Praehofer, H. , Systems theoretic formalisms for combined discrete-continuous systems simulation,International Journal of General Systems, Vol.19, pp219-240, 1991.

Project Management Institute、A guide to the project management body of knowledge, 1996.（エンジニアリング振興協会（訳)、プロジェクトマネジメントの基礎知識体系、1997）

ローゼンヘッド、ジョナサン：ソフト戦略思考、日刊工業新聞社、1992.

SAP ジャパン：IDES 機能詳細日本語版(4.6B)、SAP ジャパン、2001.

Sato, Ryo : " Toward a unified theory of discrete event systems", (Pichler, F., Moreno Diaz, M., and Albrecht, R (Eds.), Computer Aided Systems Theory -- EUROCAST '95, Lecture Notes in Computer Science, vol. 1030), pp 62-72, Springer, 1996.

佐藤亮,「仕事はイベントで動いていく」, 飯島・佐藤編『システム知の探究 2』, 日科技連出版社, 1997a.

Sato, R. and Praehofer H. "A discrete event model of business system - A Systems

Theoretic for Information Systems Analysis: Part 1." IEEE Trans. Systems, Man, and Cybernetics, Volume 27, 1997, pp.1-10.

Sato, R.: Meaning of Dataflow Diagram and Entity Life History - A Systems Theoretic Foundation for Information Systems Analysis: Part 2, IEEE Transactions on Systems, Man, and Cybernetics, 27-1, 1997, pp.11-22.

佐藤亮: ビジネスプロセスの制御システムについて, 経営情報学会誌, 8 - 1, pp17 - 28, 1999.

Sato, R.: Realization theory of discrete-event systems and its application to the uniqueness and universality of DEVS formalism, Int. J. of General Systems, 30 (5), 2001, pp. 513-549,.

佐藤亮、蔡東倫、二村暢之、小野栄一：ERP を用いてビジネスプロセスを作り出すための情報システム方法論：quickIPP― 日程計画業務の場合、Institute of Policy and Planning Sciences Discussion Paper series No. 1017, University of Tsukuba, 2003.

Sato, R. and Tsai,T.L.: An Agile Production Planning and Control System with Advance Notification to Change Schedule, Int. J. Production Research, 42-2, 2004, pp321-336.

佐藤亮: ビジネスプロセスの離散事象モデリングと業務取引ペトリネットによる特性分析, システム/制御/情報、45-8、pp470-479、システム制御情報学会、2001.

Sato, Ryo : Realization theory of discrete-event systems and its application to the uniqueness and universality of DEVS formalism, International Journal of General Systems, 30 (5), pp. 513-549, 2001.

佐藤亮: ビジネスプロセスのＤＦＤモデルとペトリネットモデル, 経営情報学会誌, 8 - 1, pp1 - 15, 1999.

Sauber, T. and Tschirky, H: *Structured Ceativity: Formulating an Innovation Strategy*, Palgrave Macmilan, 2006.

Scheer, A.W, *Business process engineering-Reference models for industrial enterprises*, 2nd ed, Springer, 1994

Sheer, A.W. , *ARIS-business process modeling*, Spinger, 1999。

Scheer, A.W.(1994): *Business Process Engineering*, 2 edtion, Springer.

Scheer, A.-W(1998), *ARIS-Business Process Framework*, 2nd edition, Springer.

Silver, Pyke, Peterson: *Inventory Management and Production Planning and Scheduling*, 3rd edition, John Wiley & Sons, 1998.

島田達巳, 高原康彦著：経営情報システム,日科技連出版社, 2001

新郷重夫：ノンストック生産方式への展開、日本能率協会、1987.

サイモン, H.A. ：稲葉, 吉原訳、システムの科学、パーソナルメディア, 1999

Stein, R.E.: Re-Engineering the manufacturing system – applying the theory of constraints, Marcel Dekker, 1996.

鈴木ユミ子、佐藤亮：プロトタイプを用いる情報システム分析の定式化と分析支援システムの実現、経営情報学会誌、2 - 2、pp47 - 68、1993.

高原康彦、飯島淳一：「システム理論」共立出版、1990.

Takahashi, S. , General morphism for modeling relations in multimodeling, Transactions of the Society for Computer Simulation International, pp169-178, 13-4, 1996.

Takahashi, S., Kijima, K., and Sato, R. （編), *Applied General Systems Research on Organizations*, Springer, 2004.

田中一成、生産・物流統合管理システム D/SNS 法、日刊工業新聞社、1988.

田中一成：時間生産性をどう高めるか、東洋経済新報社、1993

田中一成：生産の実務　製番管理、日本能率協会マネジメントセンター、2004

トーマツ・コンサルティンググループ編、購買・固定資産管理システムの設計、中央経済社、1997.

鳥羽登：SE のための MRP, 日刊工業新聞社, 1995.

Tsai,T.L. and R. Sato: A UML model of agile production planning and control system, Computers in Industry, 53, 2004, pp133-152,.

椿正明、データ中心システムの概念データモデル、オーム社、1997.

Vollmann, TE, Berry, WL, Whybark, DC (1997), *Manufacturing Planning and Control Systems*, 4th edition, Irwin McGrw-Hill.

渡辺幸三, 業務別データベース設計のためのデータモデリング入門、日本実業出版社、2001

ウイルソン、ブライアン：システム仕様の分析学：ソフトシステム方法論、共立出版、1996.

ウォマック, ジョーンズ（稲垣訳)、ムダなし企業への挑戦、日経ＢＰ社、1997.

吉田裕之、山本里枝子、上原忠弘、田中達雄： UML によるオブジェクト指向開発実践ガイド、技術評論社。1999

Zadeh, L. A. and Desoer C. A. (1963), *Linear System Theory: The State Space Approach*, McGraw-Hill.

Zeigler, B. P. *Theory of modelling and simulation*. John Wiley, 1976.

Zeigler,B. P. *Multifaceted modeling and discrete event simulation*. Academic, 1984.

Page-Jones, M.: *The Practical Guide to Structured Systems Design*, Yourdon Press, 1980

Zeigler, B. P. (1996), B. P. Zeigler からの sixth annual conference on AI, Simulation and Planning in High Autonomy Systems held in San Diego における個人的示唆。

問題略解

問 2-1

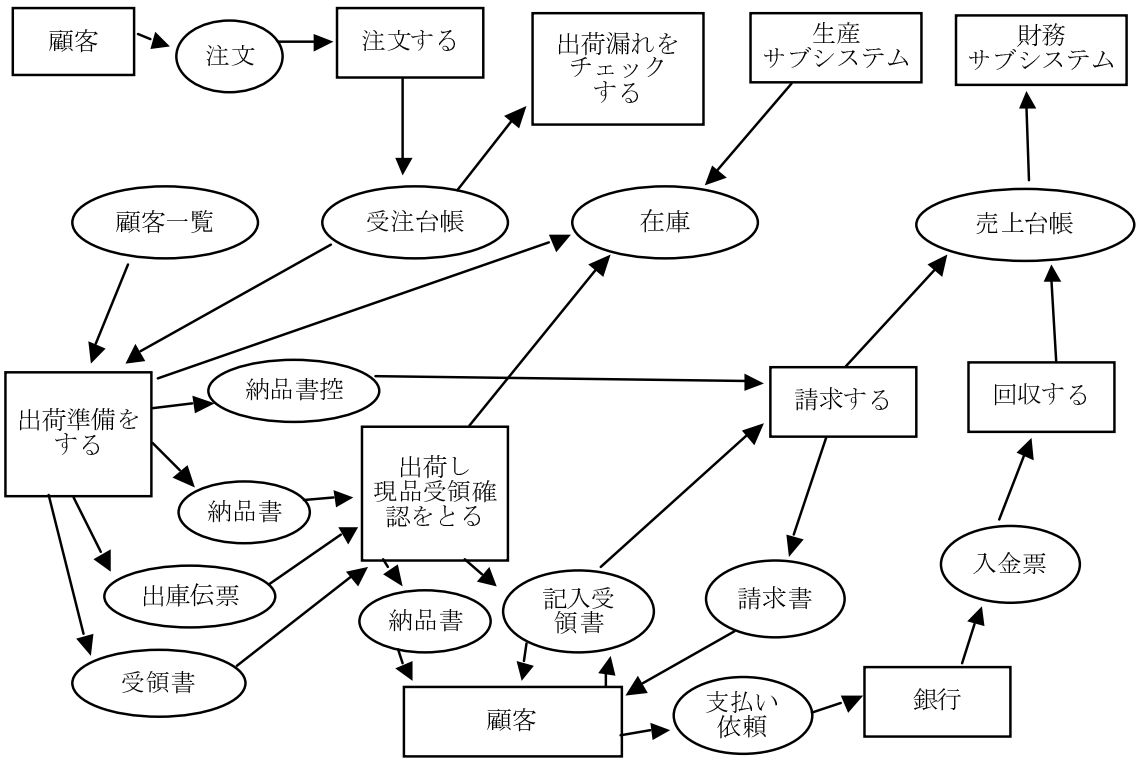


図 2-4 販売業務の AID モデル化

問 2-2

（時刻、到着する、入り口での待ち、店内で買い物をする、支払い計算待ち、支払い、キャッシュ利用可能数、帰宅）の順に次のようになる。

（30, (C5, 10), 1, (C3,5), --, --, 2, C1・C2)

問 2-3

$\delta_\Lambda(s)$ において、作業を行っている活動は３つであり、それぞれ、$ta_{Stg}(1)=5$, $ta_{AssS}(1)=5$, $ta_{AssB}(1)=6$ である。同時に、

$t'=\max\{t_{Stg},t_{AssS},t_{AssB},t_{Ass},t_{Isp}\}=\max\{14,14,9,-1,-1\}=14$ である。よって

$ta(\delta_\Lambda(s))=\min\{14+5,14+5,9+6,\infty,\infty\}-t'=15-14=1$ を得る。

問 2-4

各活動について、(1)それを開始するためにはどの情報が何のために必要か、(2)処理時間は何によって定まるか、(3) 終了時には何を出力するか、を指定する。たとえば、各活動の作業担当が 1 人（または 1 台）であって、カンバンが部品や製品 1 個に対応し、各作業の標準作業時間（段取り＋加工＋移動）を定める。または、ひとつの部品入れ（コンテナ）には 5 個を入れていて、カンバンが 3 枚蓄積したとき（つまり 3 枚分＝15 個使ったとき）に前行程が作業開始するときには、カンバンについてのその条件が成立したかと、自分がつかう部材が利用可能かを確認するという条件が成立する必要がある。

問 2-5

連続システムの場合：行列の指数関数についても、形式的にはスカラーの場合と同じ形式の微分が成立することを用いて簡単に示される。

$$\frac{dz}{dt} = \frac{d}{dt}[e^{Ft}z(0) + \int_0^t e^{F(t-s)}Gx(s)ds]$$

$$= Fe^{Ft}z(0) + \frac{d}{dt}[e^{Ft}\int_0^t e^{-Fs}Gx(s)ds]$$

$$= Fe^{Ft}z(0) + \frac{d}{dt}(e^{Ft})\int_0^t e^{-Fs}Gx(s)ds + e^{Ft}\frac{d}{dt}(\int_0^t e^{-Fs}Gx(s)ds)$$

$$= Fe^{Ft}z(0) + Fe^{Ft}\int_0^t e^{-Fs}Gx(s)ds + e^{Ft}e^{-Ft}Gx(t)$$

$$= F[e^{Ft}z(0) + e^{Ft}\int_0^t e^{-Fs}Gx(s)ds] + Gx(t)$$

$$= Fz(t) + Gx(t)$$

離散時間システムの場合：ほとんど自明である。

問 2-6

連続システムのとき状態遷移関数の半群性は以下のようになる。

. $\phi_{0t''}(c, x_{0t''}) = \phi_{t't''}(\phi_{0t'}(c, x_{0t'}), x_{t't''})$を示す。ただし、$\phi_{t't''}(c', x_{t't''}) = \phi_{0,(t''-t')}(c', \sigma^{-t'}(x_{t't''}))$であり、$\sigma$は$\sigma^{-t'}(x_{t't''})(u) = x_{t't''}(u + t')$というシフトオペレータであった。

任意の$0 \le t' \le t''$と$x_{0t''}$をとる。

$$\phi_{t't''}(\phi_{0t'}(c, x_{0t'}), x_{t't''})$$

$$= e^{F(t''-t')}(e^{Ft'}z(0) + \int_0^{t'} e^{F(t'-s)}Gx(s)ds) + \int_{t'}^{t''} e^{F(t''-s)}Gx(s)ds)$$

$$\phi_{0t''}(c, x_{0t''}) =$$

$$e^{Ft''}z(0) + \int_0^{t''} e^{F(t''-s)}Gx(s)ds$$

$$= e^{Ft''}z(0) + \int_0^{t'} e^{F(t''-s)}Gx(s)ds + \int_{t'}^{t''} e^{F(t''-s)}Gx(s)ds$$

$$= e^{F(t''-t'+t')}z(0) + \int_0^{t'} e^{F(t''-t'+t'-s)}Gx(s)ds + \int_{t'}^{t''} e^{F(t''-s)}Gx(s)ds$$

$$= e^{F(t''-t'+t')}z(0) + \int_0^{t'} e^{F(t''-t'+t'-s)}Gx(s)ds + \int_{t'}^{t''} e^{F(t''-s)}Gx(s)ds$$

$$= e^{F(t''-t')}(e^{Ft'}z(0) + \int_0^{t'} e^{F(t'-s)}Gx(s)ds) + \int_{t'}^{t''} e^{F(t''-s)}Gx(s)ds)$$

$u = s - t'$とおくと、最後の項の積分は$s = u + t',\ ds = du,\ t''-s = t''-t'-u$だから積分範囲は以下の通りとなる。

s	t'	…	t''
u	0	…	$t''-t'$

計算を続けると、

$$= e^{F(t''-t')}(e^{Ft'}z(0) + \int_0^{t'} e^{F(t'-s)}Gx(s)ds) + \int_0^{t''-t'} e^{F(t''-t'-u)}Gx(u+t')du)$$

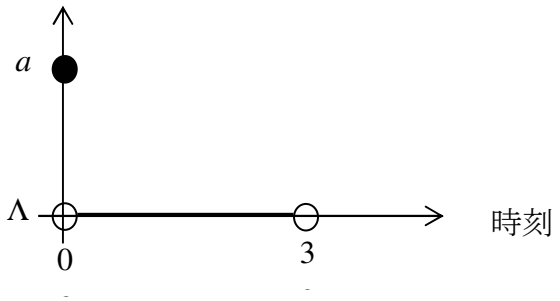
$$= e^{F(t''-t')}(e^{Ft'}z(0) + \int_0^{t'} e^{F(t'-s)}Gx(s)ds) + \int_0^{t''-t'} e^{F(t''-t'-u)}G(\sigma^{-t'}x)(u)du)$$

$$= \phi_{0,(t''-t')}(\phi_{0t'}(c, x_{0t'}), \sigma^{-t'}(x_{t't''}))$$

を得る。

離散システムの状態遷移関数の半群性は省略する。

問 2-7

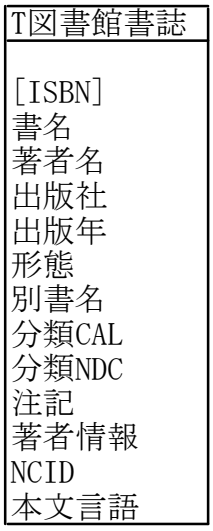


問 2-7

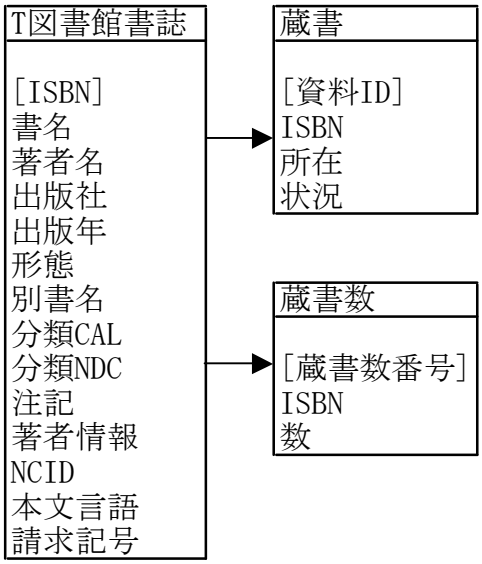
$D(b^2) = [t_1, t_2)$のときの関数$\sigma^{-t_1}(b^2)$のグラフは、問 2-6 解答のように時刻 0 から$t_2 - t_1$までの間で定義され、時刻 0 での値がbであるような関数である。

問 3-1

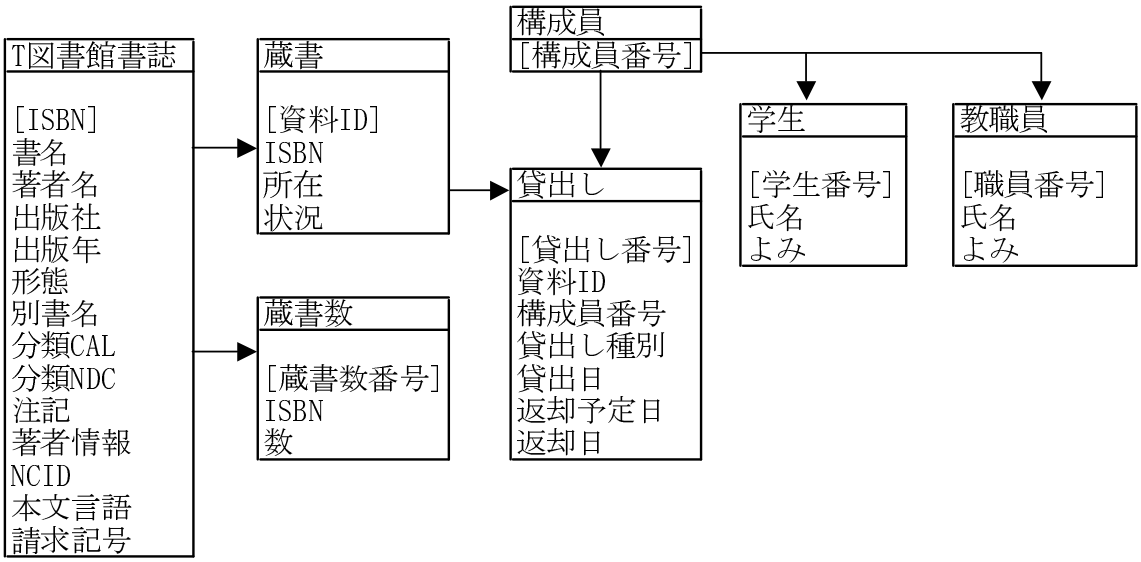
（１）



（２）

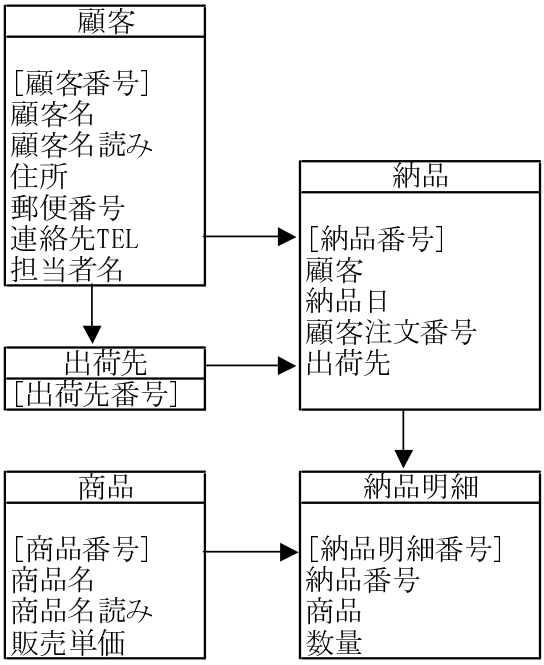


貸出しも考慮すると次のようにモデル化できる。



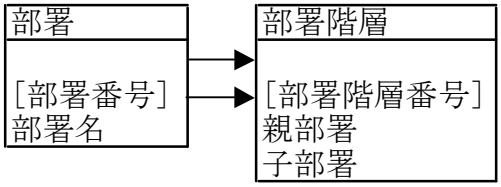
問 3-2

管理実体型、属性、参照関係を示す。ドメインは省略する。また、継承関係はない。ひとつの管理実体型がひとつの表として DBMS を使って実現できる。１枚の納品書は、DBMS の画面フォームやプリントフォームとして実現できる。その際に、納品細目ごとの「金額」欄は、納品数量と販売単価から計算される。顧客ごとの販売実績（受注実績）は、納品とは異なるできごとなので、別途管理する。また、「納品番号」は納品書においては納品書 No として表示する。



問 3-3

第 5 章の部品表と同様のデータモデルとなる。



このモデルを実現すると、たとえば、実際にテーブルとしては以下のようになる。

部署	
部署番号	部署名
51	人事部
52	人事課
53	人事係
54	人材開発係
55	給与係
56	厚生係
57	寮社宅係
58	厚生課

部署階層		
部署階層番号	親部署	子部署
1	51	52
2	52	53
3	52	54
4	51	55
5	51	58
6	58	56
7	58	57

問 3-7

顧客番号によって顧客名が唯一に決まる。また品目番号によって品目名が唯一に定まる。したがって、３つの表で第 2 正規形にした。受注控え表の主キーは[顧客番号, 品目番号]、顧客表の主キーは顧客番号、品目表の主キーは品目番号である。

受注控え		
顧客番号	品目番号	数量
1150	I025	50
1150	P321	23
1150	S153	15
2030	P321	35
2030	S166	23
3040	N216	155

顧客	
顧客番号	顧客名
1150	川田商店
2030	青木工業
3040	トミカワ

品目	
品目番号	品名
I025	サンドペーパ 25 番
P321	P 社油性ペイント赤 1L
S153	S 社水性ペイント黄 3L
S166	S 社水性ペイント緑 6L
N216	N社殺虫剤小

さらに検討する。正規形の議論を離れて、データモデルの本来の目的と機能にかえって反省してみる。すると、顧客からの１回の注文は普通は明細を含んでいることから、図 3-10 のような受注管理実体型と受注明細管理実体型の２つの管理実体型が必要である。受注管理実体型には受付日や、受け付けた当社の担当者を記載する。しかし、上の受注控えはそのような構造になっていない。受注明細のようであるが、不完全な情報しか持っていない。なぜか。実は、上の受注控えとは、ある一人の受注担当者もしくはセールスマンのメモであることに思い至る。したがって、この特定セールスマン独自の「受注控えメモをターゲットにしてモデル化してデータベース化を考える」コト自体が奇異である。誤りである。セールスマン全員の受注活動のメモを考察の範囲に含めて、データモデル化を考えると、やはり、図 3-10 の中の「顧客、商品、受注、受注明細」の４つの管理実体型をすくなくとも含むようなデータモデルでないと、現実に発生している受注活動の記録たり得ないのである。正規化という考え方のみではなく、データモデルは現実世界の事柄の記録のための概念枠組みであることを認識しないと、現実との対応がない使えないデータモデルを作成してしまう可能性がある。

問 3-8、問 3-9

データの図を見てまず気が付くのは、管理実体型の名前が宿泊予約であるにも関わらず、主キーの候補として、顧客番号を使っていることである。これだけでも、既に、これからスタートして正規化手続きを進めても、まともなデータモデルを得られないことが予想できる。宿泊予約管理実体型は、現実世界で発生する宿泊予約ということを管理したいのである。主キーは「宿泊予約番号」とすべきである。同じことを異なる言い方をすれば、もし同一の顧客がスキーで北海道に行き、帰りがけに東北の温泉の予約をする場合、このモデルでは「同一顧客が複数の予約をする」ことを受け付けられないため記録できない。また、同じ温泉宿に有名なホテルが 2 軒あって、一人の客が日を変えて宿泊したいときも受け付けられない。顧客番号は顧客という現実存在を認識することを自然に表す目的の顧客管理実体型の主キーであるべきである。さらに、ホテル関係

の情報は、予約というイベントには無関係に存在する、宿泊のためのリソースであるから、リソースを管理する宿泊施設管理実体型として別途認識すべきものである。

この例は、データモデリングは「とりあえずデータ項目の候補を並べておいて正規化手続きによって機械的に正規化していくと得られる」というものではないことを示している。現実でのデータの発生のようすと、それをとらえる人間の認識・概念の方から考えると、正しくかつやさしく進めることができる。

顧客、宿泊施設、予約、予約明細という４つの管理実体型を使い、受注と同様のデータモデル基本形を作成できる。

問 4-1

$Q_{id} = \{q_1, q_2, q_3\}$であるので、空集合を$\phi$と表すときに、べき集合は

$P(Q_{id}) = \{\phi, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$

である。

（なお、ひとつの部分集合を決めることは、もとの集合Q_{id}の個々の要素ひとつひとつについてその要素を含むか含まないかの２者択一を指定することに他ならない。したがって、可能な部分集合の個数は、Q_{id}の要素数をnとするとき2^n個となる。）

問 4-2

f_{AK}(請求照合) = {照合担当, 照合済支払い}

f_{KA}　(請求照合) = {請求書, 仕入れ記録, 照合担当}

問 4-3

$$ta(s) = \min\{225+15, 231+7, 225+10, 203+29\}$$
$$= \min\{240, 238, 235, 232\} = 232$$
$$t'(s) = \max\{225, 231, 225, 203\} = 231$$

問 5-1

（１）

単位　10 億円

資産の部		負債および資本の部	
流動資産		流動負債	690
当座資産	560	固定負債	123
棚卸資産	120	負債計	813
流動資産計	680	資本の部	
固定資産		資本金	90
有形固定資産	421	（その他資本の部）	285
無形固定資産	99	当期未処分利益	12
固定資産計	520	資本計	387
資産合計	1200	負債及び資本合計	1200

単位　10 億円

売上高	3,540
売上原価	2,800
売上総利益	740
販売一般管理費	400
営業利益	340
営業外損益	90
経常利益	250

問 5-3

サプライヤ A

期 n	期首在庫 z(n)	n 期の入荷数 x(n)	n 期の出荷数 y(n)	売上総利益 （万円）
1	0	30	10	500
2	20	0	10	500
3	10	0	10	500
4	0	30	10	500
5	20	0	10	500
6	10	0	10	500

問 5-4

リードタイム 3、ロットフォーロット、初期在庫 5、オーダ残が 1 期に 5 個。

タイムバケット		1	2	3	4	5	6	7
総所要量			5		10			
オーダ残		5						
予想在庫バランス	5	10	5	5	0	0	0	0
着手日順計画オーダ		5						

問 5-5

品目表

品目#	品目名	ローレベル
2001	A	0
2002	C	2
2003	D	1
2004	E	2
2005	F	2
2006	B	0
2007	G	1
2008	H	1

部品構成表

構成#	親品目	子品目	構成数
1	2001	2002	1
2	2001	2003	1
3	2003	2004	1
4	2003	2005	1
5	2006	2007	1
6	2006	2008	1
7	2008	2002	1

問 5-6

第 2 期総所要量を 10 減らし、第 3 期にまわす。

タイムバケット	1	2	3	4	5	6
X からの総所要量			10	10	15	
D からの総所要量		4	20			
オーダ残						
（有効在庫）			-30	-10	-15	
正味所要量			30	10	15	
完了日順計画オーダ			30	10	15	
着手日順計画オーダ		30	10	15		

問 7-1

第（5）行と第（9）行は時刻と p4Q 以外は同じ内容である。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(5)	10	1(10)	0	1(5)	0	Λ	0	1(5)	2
(9)	20	1(10)	0	1(5)	0	Λ	0	1(5)	3

問 7-2

第（2）行と第（4）行は時刻と p4Q 以外は同じ内容である。

行番号	time	Cust	p1Q	act1	p2Q	act2	p3Q	act3	p4Q
(2)	5	Λ	60	Λ	60	Λ	60	Λ	1
(4)	10	Λ	60	Λ	60	Λ	60	Λ	2

問 7-3

これまでの計算から、状態遷移表が定常状態（平衡状態）になる場合には、買掛金決済プロセスのようなサービスの業務プロセスであっても、つまり特に物体を扱わないプロセスであってもリトルの定理が成立する。このことは 7.4 節で定理として証明される。

問 7-4

１周期の長さは 65 分なので、スループットは TH=1/65 [個／分]。

図 7-5 に示された平衡状態の状態遷移から、プロセスを律している業務連鎖パスは組立て２（Asm2）と組立て F（AsmF）である。そのパスにおける WIP を 1 周期分計算する。組立 2 においては時刻 2,080 から 2,136 まで 1 個を保持しているので 57[個・分]の在庫が在る。組立 F の分も同様にして次を得る。

57（組立 2）＋33（組立 F）= 90 [個・分]

よって平均在庫数 WIP は、

WIP = 90/65 [個]

となる。よってリードタイムは

LT = WIP/TH = 90 [分]

つまり、もし新たにリリース（指図）された注文があるとすると、90 分後には完成する。なお、部品 1 に在庫があるが、これはプロセスのリードタイムには無関係である。本例ではリードタイムの意味が「オーダのリリースから完成までの時間」ということなので、プロセスを律している業務連鎖だけに注目することになる。

問 8-1

time	As	qA	Aw	qS	pA	Bs	qB	q1	pB1	pB2	Bw	q2	F
36	1	0	0	0	0	0	0	1(25)	0	1	0	1(17)	1
36	0	0	0	1(25)	0	0	0	1(25)	0	1	0	1(17)	1

問 8-2

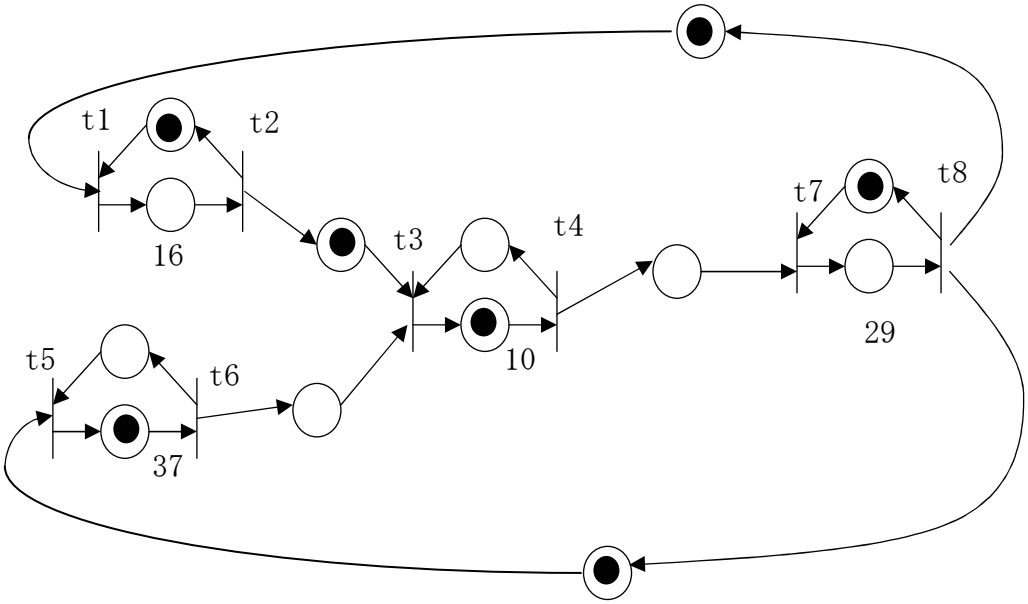
$t_1t_2t_1, t_3t_4t_3, t_8t_7t_8, t_6t_5t_6,$

$t_1t_2t_3t_4t_7t_8t_1, t_7t_8t_1t_2t_5t_6t_7$

問 8-3（強連結な買掛金プロセス）

発火回数	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
1	29	45	66	76	29	66	0	29
2	105	121	142	152	105	142	76	105
3	181	197	218	228	181	218	152	181

状態遷移表から抜き出した発火時刻表（発火回数ごとの時刻）



$$A_0 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 \\ 16 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 10 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 37 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 29 & \varepsilon \end{pmatrix}, A_1 = \begin{pmatrix} \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & 0 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}$$

$$A=\begin{pmatrix} \varepsilon & 0 & \varepsilon & 29 & \varepsilon & \varepsilon & \varepsilon & 29 \\ \varepsilon & 16 & \varepsilon & 45 & \varepsilon & \varepsilon & \varepsilon & 45 \\ \varepsilon & 16 & \varepsilon & 66 & \varepsilon & 37 & \varepsilon & 66 \\ \varepsilon & 26 & \varepsilon & 76 & \varepsilon & 47 & \varepsilon & 76 \\ \varepsilon & \varepsilon & \varepsilon & 29 & \varepsilon & 0 & \varepsilon & 29 \\ \varepsilon & \varepsilon & \varepsilon & 66 & \varepsilon & 37 & \varepsilon & 66 \\ \varepsilon & \varepsilon & \varepsilon & 0 & \varepsilon & \varepsilon & \varepsilon & 0 \\ \varepsilon & \varepsilon & \varepsilon & 29 & \varepsilon & \varepsilon & \varepsilon & 29 \end{pmatrix}$$

$$X(0)=\begin{pmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \\ 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}, X(1)=\begin{pmatrix} 29 \\ 45 \\ 66 \\ 76 \\ 29 \\ 66 \\ 0 \\ 29 \end{pmatrix}, X(2)=\begin{pmatrix} 105 \\ 121 \\ 142 \\ 152 \\ 105 \\ 142 \\ 76 \\ 105 \end{pmatrix}, X(3)=\begin{pmatrix} 181 \\ 197 \\ 218 \\ 228 \\ 181 \\ 218 \\ 152 \\ 181 \end{pmatrix}$$

索引