情報システムと社会環境　77－10
（２００１．６．２６）

ビジネスモデルとデータモデル

児玉公信
(株)エヌ･ケー･エクサ

ビジネスモデルとは，ビジネスがどのように機能するかを抽象化してとらえたものである．本稿では，モデリングに関する概念を整理し，UML とその拡張を用いてデータモデリングとビジネスモデリングの手法を紹介する．次に，これをいくつかの事例に適用して結果を評価し，最後にいくつかの課題について述べる．

Business Model and Data Model

Kiminobu Kodama
NK-EXA Corporation

Business Model is a model that captures abstractly how the business is functioned. In this paper, I will arrange the concepts of modeling, introduce some techniques of data modeling and business modeling using UML and its extension, then apply them to some cases and evaluate the results. Finally, I will state some issues to be resolved.

1. はじめに

昨今，ビジネスモデル特許という言葉が大きく取り上げられている．ビジネスモデルを特許にしようということだが，有名なところでは，トヨタの生産管理方式や，デルコンピュータの受注組立方式，インターネットでの取引の仕組みなどがあるようである．こうしたビジネスモデルは大体，事業が利益を出す仕組み，生産性や効率を上げる仕組みという意味で使われている．

ここで取り上げようとするビジネスモデルは，事業の仕組みあるいは業務の仕組みというような本来の意味のものとする．すなわち、ビジネスモデルとは，ビジネスがどのように機能するかを抽象化してとらえたものである．この意味でのビジネスモデルのポイントは，ビジネスに関わるいくつかの概念，たとえば，目的，プロセス，エンティティ，組織などについて述べるものでなければならない．また，そのモデルは一定の表記ルールに基づいて，第三者が了解可能なものでなければならない．

ビジネスをモデリングする意味は，その複雑性を捨象して，ビジネスの違いを論じ，予測し，操作できるようにすることにある．これによって，ビジネスプロセスや組織，事業の目的などを観測に基づいて修正していくことができる．利益を出す仕組みというのも，この延長線上にあるだろう．しかし，現在これが十分に行なえる状態になっているとはいえず，課題となっている部分もある．

ここでは，まずモデリングの概念について述べ，ついでビジネスモデリングの重要な部分であるデータモデリングについて述べ，最後にビジネスモデリングについて述べる．

2. モデリングの概念

2.1. モデリングの考慮点

ここでいうモデルとは，認識対象を思考操作のために必要なものだけを取り出して単純化した概念構造の表現である．思考操作とは，構造を理解して，シミュレーションしたり，予測したりすることである．表現することで他者との認識の共有が可能となる．対象の領域や目的に応じて，表現方法は図や模型，数式など，さまざまな様式が選択される．

(1) 認識主体

ビジネスや情報システムのモデルには認識主体が存在することを意識しておく必要がある．これらのモデルは，認識主体のメンタルモデル[1]だからである．ただし，認識主体はモデルを書く人（モデラー）ではなく，そのビジネスあるいは情報システムのオーナ[2]（の抽象）である．たとえば，会社とか組織といったものになる．

自分の夢に自分が出てくることがないように，認識主体自身がモデルの中に直接登場することはない．1つのモデルは１つの認識主体の立場から記述される．1つのモデルが異なる複数の立場から記述されると，それは誤解を導くことになる．また，異なる認識主体のモデルを接続しようとする場合は，意味のギャップが問題となる．

(2) 関心

ビジネスや情報システムという目に見えないものをどのようにモデル化するかについては，昔から検討されてきた．モデルを2 次元の文書に記録するためには，2 次元で表現するのが適当である．まさに，建物などの 3 次元構造物の設計書（これもモデルである）は，正面図，側面図，平面図の 3 つ視座から 2 次元に投影することで記述される．

情報システムを抽象的に多次元（少なくとも時間軸を持っている）の構築物と考えて，これをどういう側面でとらえて 2 次元表現すべきかについて，DeMarcoは，経験的にはデータの保有，機能，状態遷移という3 つの側面を記述すれば十分であると述べている[3]．データの保有，機能，状態遷移などを「関心（concern）」と呼び，1つの関心事について，1つのモデル表現を用いる．各モデル表現においては関心が十分に分離されていなければならない．

(3) 観点

モデリングの目的が，対象の理解か情報システムの構築にあるかによって記述すべき内容が異なる．この目的を観点（perspective）という．理解目的のモデルは概念的観点といい，構築目的のモデルは，設計目的と実装目的に分けて，それぞれ，仕様の観点，実装の観点という[4]．これらの観点の境界は明確でないので意識されにくい．

概念的観点では，その問題領域（ドメイン）における概念とそれらの構造を記述する．ビジネスモデリングは，概念的観点で行われるのが普通である．ここでは，基本的に実装を考えない．仕様の観点では，モジュール間のインタフェースを決定することに重点が置かれ，実装の観点では具体的なプログラムを設計する．

2.2. 本質性

(1) 深さ

そのモデルがどの程度，対象世界の本質を捉えているかは重要である．概念的観点のモデルはメンタルモデルであるので，認識の深さがそのまま表現される．対象を表層的に見てモデルを記述することはできる．たとえば，図 1 のような型モデルを見ることがある．申込書があって，それに対する手付金を受け取り，それに対する領収書を作成する作業らしいのだが，このモデルは，誰がどんな申し込みをしたかにはまったく興味がないと述べている．こうした手続

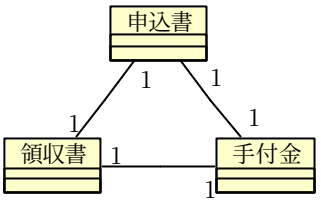


図1 表層的な概念モデル

きだけの窓口の作業としてはこれでいいのであろうが，それはビジネスのほんの上っ面であり，本質からは程遠いといってもよいであろう（大体，○○書などがビジネスの「概念」であるということが不思議である）．問題は，こういうモデルでも情報システムは作れてしまうことである．

(2) 耐変更性

ビジネスの要求は絶えず変化するものである．仮に図 1 のモデルで要求が満足されていたとしても，すぐに，誰がどんな申込みをしているかを知りたくなる．モデラーは，それは申込書エンティティの属性なのだと言い逃れをするかもしれないが，では，1 人が複数の申し込みをしていることを管理したいとしたらどうであろう．また，同じ「申込み内容」が複数の「申込み」事象で使われていることを管理したいとしたらどうであろう．このような要求に対応するためには，このモデルは変更されなければならない．こうした要求がありうることは，普通はビジネスのオーナ，つまり認識主体が窓口の担当者ではなくて「会社」であることがわかっていれば，そこから容易に推測できる．本質に迫るモデルは，当然起こりうる要求の変更に強いものである．

2.3. モデリングの方法

(1) モデリングの方法論

モデリングの表記法そのものは簡単だが，モデリングは難しいといわれる．これまでに多くの開発方法論が提案されてきたが，モデリングについての本質的な方法論は知られていない．それは，人はどう認識するかという問いに対して答えが言えないのと似ている．

モデリングは概念化の過程であり，そこでは解の探索と抽象化が行なわれる．これには周辺知識が必要であり，一方で reasoning というべき過程もある．しかし，これらをどう体系づけて形式化すればよいか，今のところははっきりわかっていない．名詞抽出法や

動詞抽出法といった手法が提案されているが，実際のモデラーはこのようなことはやらない．

方法論が確立されていないので，モデリングの手法を身につけるには，経験的に行なうしかない．とにかくモデリングしてみて，結果的に良いモデルであったかどうか，改善するにはどうすればよいかを検討して，モデルを磨いていくことになる．経験的に良いとされるモデルをパターンとして記録して，それを学んだり再利用するというのも，モデリングを習得する次善の策である．

(2) モデルの表記法

モデルの表記法は，モデリングそのものに比べて本質的な問題ではないと思われているが，理解を共有するという面では重要である．これまでは，分析方法論の一部のようになっていた表記法を，Rumbaugh, Booch, Jacobson（three amigos）は UML（Unified Modeling Language：統一モデリング言語）としてまとめ，1997 年に OMG はそれを標準として採用し，以降，改訂タスクフォースが鋭意改訂を進めている．最新バージョンは 1.3 であるが，まもなく 1.4 がリリースされる．また，これと平行的に 2.0 が検討されている．このように，UML は現在のソフトウエア工学の知見が集約されている．

UML は開放されたモデリング言語であり，使用，流通にあたってなんら制約はない．モデルの基本要素を使って，記法を拡張することもできる．また，OMG での議論を通して改訂の提案ができる．こういう意味で，UML は事実上の標準とみなしてもよい．

UML は，共通の表記ルールと，ダイアグラムごとの表記ルールからなる．上述した DeMarco のデータの保有，機能，状態遷移という3 つの側面をUML で書くとすれば，それぞれ，クラス図，シーケンス図とコラボレーション図，ステートチャート図とアクティビティ図が対応するだろう．以下では，この UML のクラス図とアクティビティ図を使用して説明する．

3. データモデル

データモデルは DeMarco のいうデータの保有側面のモデルに相当する．ただし，データモデルというと，情報システムの構築目的のモデルととらえられることがあるため，概念的観点で書かれるモデルを「型モデル」と呼んで使い分ける場合もある．ここでは，後者の意味でのデータモデルについて論じる．

型モデルで表したいことは，モデル対象のビジネスで現れる概念とそれらの構造である．その上で，オブジェクト指向ではこうしたデータ構造を維持する責任を，型（クラス）の操作として持たせ，非オブジェクト指向では，その責任を機能や状態遷移の側面で述べられる操作で実現する．

次では，UML のクラス図[4][5][6]を使ったデータモデルの表記法について述べる．

3.1. 型

概念は集合[7]として記述することができ，その集合を識別するいくつかの属性が存在する．このように識別された概念を，「型（type）」または「エンティティ（entity）」と呼ぶ．ER 図（Entity-Relationship Diagram）を提案した Chen[8]は，Entityset と呼んで，集合であることを明示している．型は，定義により正規形である．

型の要素をインスタンスと呼ぶ．型は矩形で表し，その中に型名を記す．さらに矩形の中に区画を設けて属性を記述してもよい．

こうした「型」は本来，実装とは独立であるが，方法論の中には関係型データベースの表として実装することを前提としているものもあり，多対多の関連を許さないなどの不当な制約を課したり，主キー，外部キーなどの実装要素の識別を強制するものもある．

3.2. 関連

(1) リンクと関連

異なる型のインスタンスどうしの対応関係をリンクという．リンクの集合を関連（association）という．関連に対してその関わり方を説明する名称を定義できる．関連は型間の線で記述し，関連名はその線の中央付近に記述する．

関連自体を型として識別することもできる．これを Chen は関連（relationship）と呼んだが，UML ではこれを特別に扱うことはしない．

(2) ロール

関連に参加する 2 つの型の一方が他方に対して負う役割をロール（role）という．ロールはコンテクストに依存する．ロールを的確に識別することは，対象の本質に迫る一つの道である．たとえば，ある会社における上司と部下の関連は，それぞれのエンティティが存在するのではなく，社員というエンティティがその関連において，上司と部下という役割を演ずるだけである．ロールは，関連を表す線とそれを演ずる側

のクラスシンボルの付近に記述する.

(3) 多重度

関連の対応数を多重度（multiplicity）という. 一般に，対応数は 0, 1，それ以上の 3 段階でとらえることが多い．これは，対応数の最小と最大を. . でつないで表記する．これを関連先のクラスシンボルの付近に表記する．最小が 0 の場合は省略して，単に最大のみを記すことが多い．多重度の最小が 0 の関連を任意といい，1以上の場合を必須と呼ぶこともある.

(4) 集約

集約は関連の一種で，関連の線に対して集約する一方の型（コンポジット）をそれに付けた白抜きの菱形で明示する．これは，モデルの実質的な解釈には寄与しないので，省略されて，通常の関連として記述されることも多い.

もう一つの集約としてコンポジション集約がある．これは集約する型のインスタンスが存在しなくなると，集約される型（コンポーネント）のインスタンスも存在しなくなることを黒い菱形で明示する．これは，Chen が「弱いエンティティ」と呼んだものであり，参照制約が適用される.

(5) サブタイプ

型の部分集合をサブタイプ（subtype）と呼ぶ．これは図 2 のように記述する．部分集合の認識（分類）は任意にできるので，概念的観点では多重分類（multiple classification）を認める．このときの分類軸は，それぞれの三角形のそばに示す「弁別子」（discriminator）．その分類軸で補集合がない場合には {complete} を付けてその旨を示す.

状態を持つエンティティのサブタイプは，個別の状態であることも多い.

(6) ステレオタイプ

ステレオタイプはクラス図だけでなく，他のダイアグラムでも共通に使用できる UML の拡張メカニズムの一つである．型図では，たとえば，類似した型を識別するために使用できる．ステレオタイプは，その名称を<< >>で囲んで示す.

実は，クラス図を型図として用いるためには，クラスシンボルにステレオタイプ<<type>>をつけて明示することになっているが，概念的観点ではすべてが型なので，ここでは省略する．それ以外のステレオタイプは，必要に応じて記述する.

(7) 制約とルール

型図のような構造モデルでは，各インスタンスが整合的に保持されることを保証するために，インスタンスやリンクの存在を許す条件，許される関連の条件，導出条件などを記述する．これは，ビジネスレベルでは，ビジネスルールとも考えられる．UML では OCL（Object Constraint Language）[9] による表記を推奨している.

3.3. データモデルの例

ここまでに説明したモデル要素を組合わせることでデータモデルが記述できる．図 3 は Chen の中で述べられた従業員管理のモデルをシンタックスの説明のために拡張したものである.

このように，ビジネスで扱われるエンティティとその間の構造をデータモデルとして記述する方法は規定できたが，このスコープをどのように限定するべきかについては方法が示せない．基本的には，関心の範囲に限定すべきである.

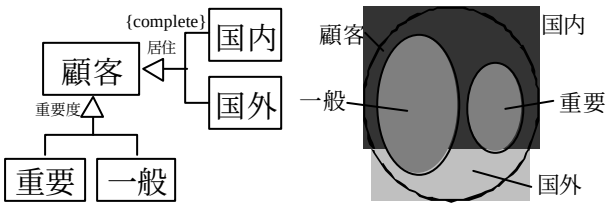


図 2　サブタイプ

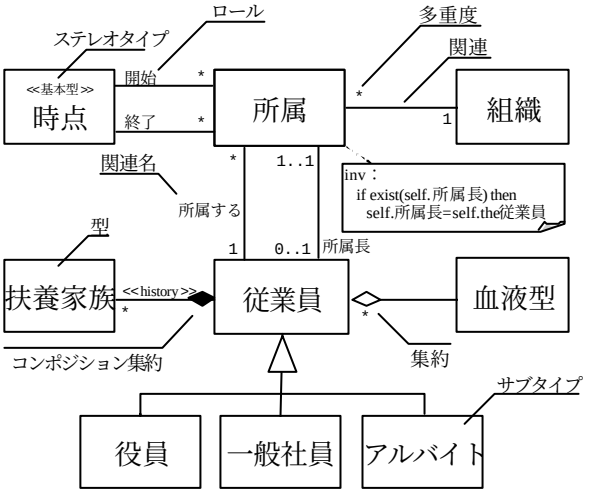


図 3　従業員管理のデータモデル

3.4. データモデルの理解

第三者の書いた型図を理解するために，インスタンス図を使うことがある．これは型図を基にして，事例を想定して必要なインスタンスとリンクを記述したものである．インスタンスは，インスタンス名：クラス名と表記する．

逆に，インスタンス図が与えられて，これからデータモデルを作成することは非常に困難である．

4. ビジネスモデル

ここで取り上げるビジネスモデルは，事業の仕組みあるいは業務の仕組みを鳥瞰するものである．一般に，ビジネスモデルを単一のモデルとして記述することはできない．戦略モデル，実行モデル，収益モデル，競合モデルという4 種で議論する例[10]もあるが，ここでは，Marshall[11]の事業の目的，プロセス，エンティティ，組織という4 面からとらえてみる．

4.1. 目的

目的の一般的構造は，ヴィジョン，ミッション，ゴール，努力目標という階層構造を持っている．末端の努力目標は属性として完了予定日と目標値をもち，これに対する実績値が対応づけられて，その達成度が評価される．

目標の達成のために，作業計画が立てられる．これも目的の階層構造と同様な階層構造になる．具体的には，WBS（Work Breakdown Structure）やマイルストンの設定，スケジューリングである．

また，計画立案にあたっては，どのような状態が望ましいかを定める方針が大きな役割を持つ．たとえば，在庫を最小にするとか，価格政策などである．

これらは型モデルとして一部は表現可能であり，何に関心を持つか，どのような作業構造となるかを記述することになろう．Fowler の観測パターン[12]も参考になる．

4.2. プロセス

ビジネスプロセスについての明確な合意はないが，ここでは，目的の具体的な達成手順であるとする．これは，一連のプロセスステップを持ち，分岐や合流を伴うワークローとして記述できる．

(1) Customer-Performer

ワークフローの最小単位は，Customer（作業の依頼元）とPerformer（実行者）の間での一連のやり取りである[13]（図 4）．Customer が Performer に作業を依頼するとき，その条件について打診をする．Performer からも受諾の条件が提示され，代替案が提示されたり，条件を譲歩するなどして合意に至る．Performer は実行して，その結果をCustomer に報告し，その結果を受け入れる，あるいは拒否する，というのが一連のフローである．この Customer とPerformer 間のコミュニケーションに仲介者が入ることもある

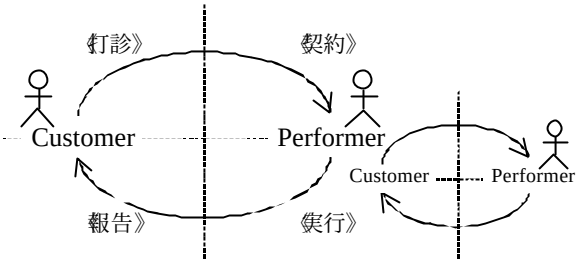


図 4　Customer-Performer モデル

Performer はそれを別のPerformer に再依頼することができる（サブワークフロー）．このとき，最初の Performer はロールを変えて，Customer を演じることになる．再依頼の作業は分解されることもあり，それが並列に実行されることもあるが，一連のフローは，最終的には，最初の Customer への報告で終わる．これが報告責任（accountability）である（Fowler の責任関係パターン[12]も参照）．

(2) ワークフロー図

UML ではワークフローを表現するのにアクティビティ図[4][5][6]が使用される．これは，先行するアクティビティの完了によって次のアクティビティがトリガされることを表す．これは，状態図の変形であり，トリガには事前条件や事後条件が指定できるので，結果的に制御フローになる．DeMarco のいう状態遷移のモデルに相当するといえよう

アクティビティ図にロールごとのレーンを設けて記述することもある（図 5）．制御のフローがレーンをまたぐとき，そこには何らかの作業者の目に見える指示が必要となる．伝票やカンバンなどである．

ワークフローの表記法としては，他に，WfMC（Workflow Management Coalition）が規定したものや，IDEF0 などがあるが，基本的にその記述内容は大きく違わない．

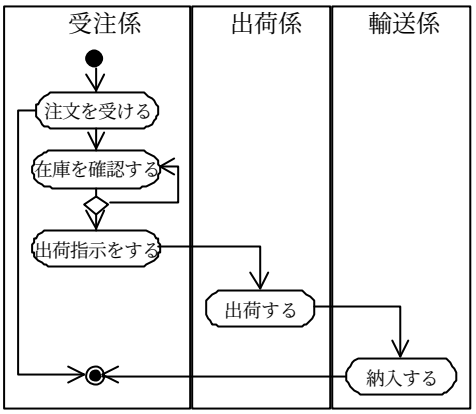


図 5　レーンつきのアクティビティ図

(3) ビジネスユースケース

アクティビティは，ロールの作業であり，基本的には job description として与えられる．情報システムとの関連を問題にするとき，その作業の中で情報システムとどのような対話をするかを記述するのがビジネスユースケースである．これがDeMarcoのいう機能に相当する．

ビジネスユースケース[14]は，要求の形式的記述を行なうものであり，システムの内部を記述しないことが重要である．

(4) Eriksson-Penker の拡張

アクティビティ図はビジネスモデルを記述するには詳細すぎる．Eriksson とPenker は UML を拡張してプロセス図を導入した[15]．プロセスは一連のアクティビティおよび／またはプロセスそのもののパッケージ¶として定義されたステレオタイプである．プロセス図によって，ビジネスレベルでのプロセスが記述できるようになった（図 6）．

さらに, Eriksson と Penker は，プロセスが進行する過程で状態を変えていくオブジェクトに注目し，それを記述するアセンブリライン図という拡張も提案している（図 7）．これは，図の形が製造組み立てラインに似ているということで命名された．

このアセンブリラインの部分はパッケージであり，その中に注目するオブジェクト（1 つのオブジェクトかも知れないし，オブジェクト群かもしれない）が write（●）されたり read（○）されたりしながら，そのオブジェクトが状態を変えていく様子が表されている．

¶ さまざまなモデル要素をグループ化するUMLのメカニズム.

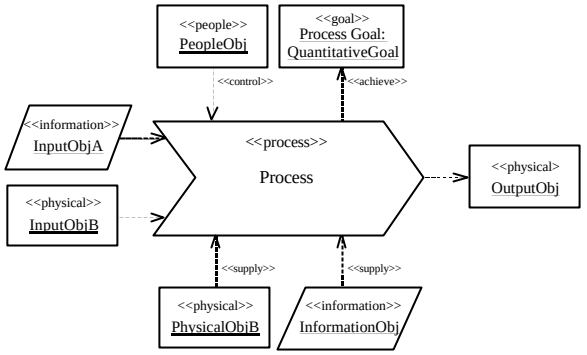


図 6　プロセス図の一般形

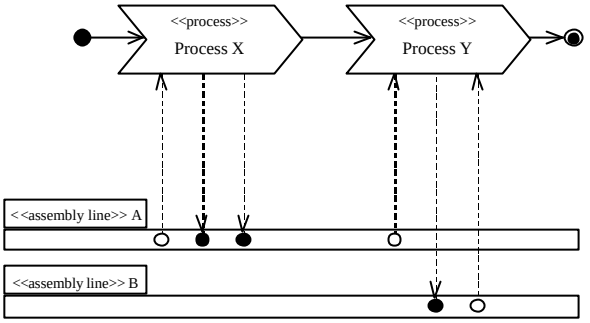


図 7　アセンブリライン図の一般形

4.3. エンティティ

エンティティは, 3 章で述べたデータモデルで表現される実体である．

Marshall[11]は，エンティティを，文脈やプロセスによって振舞いが違って見えるエンティティロールと，振舞いが変わらないエンティティバリューに分類している．エンティティロールの例としては，パーティ，顧客，従業員，アクタ，人工物，資産，リソース，製品，場所など，エンティティバリューの例としては，勘定，アドレス，アスペクト（能力，品質，サイズなど），容量，記述，在庫保有，価格などが上げられている．エンティティをこのような 2 種類に分けてとらえる点はユニークで有用であろう

これとは別に，筆者は，ビジネスプロセスの進行に伴って状態を変えるエンティティという識別も有効であると考えている．これは先に述べたアセンブリライン図で記述される．

4.4. 組織

自社の組織はモデリングの認識主体であるため，エンティティモデルには現れない．他社はパーティと

して現れ，その関係が重要なモデル記述となる．

一方，組織はプロセスの実行主体でもあり，その実行をスムースに行なうためのコミュニケーションと監視機能を持つ．そのために階層構造をなす場合が多い．

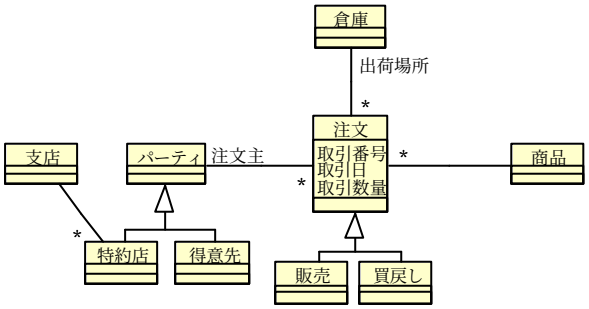
組織を，上で述べたようなビジネスモデルの中に位置づける具体的な手法は提案されていない．モデリング上の課題としては，自律的な組織ユニット間の調整過程であろう．

5. ビジネスモデルの事例

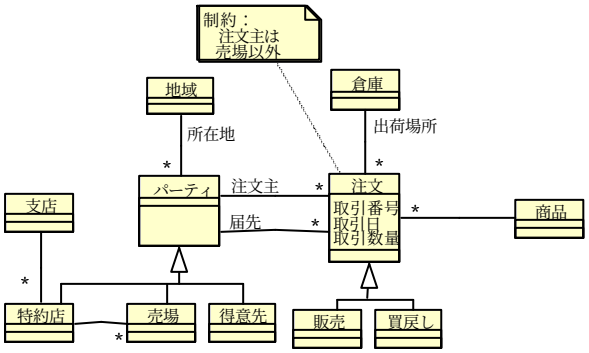
これまでに導入したビジネスモデリング手法を用いて，いくつかの簡単なケースについてモデリングを試みる．

5.1. 型モデルの変遷

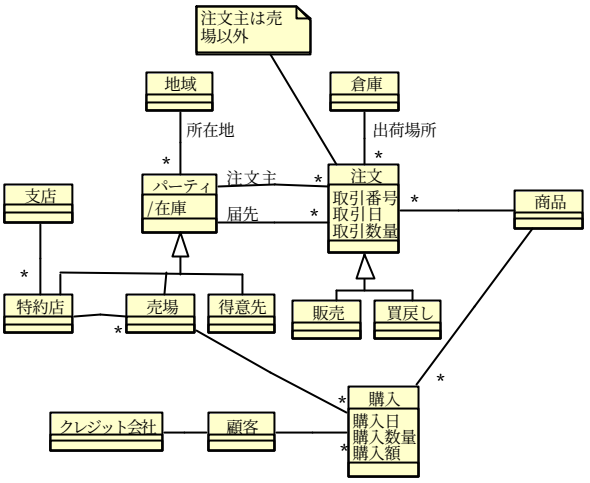
最初のデータモデルが要求の変更に伴ってどのように変わるかを追跡してみる．この事例は，在庫商品の特約店販売および直売のビジネスである（図 8a）．当時のコンピュータの入力機器や記憶媒体の物理的制約によって，情報システムがゆがめられていたこともあって，このモデルは実態を反映しきれて



a 最初のモデル



b 最終ユーザ把握のためのモデル拡張



c 売場の在庫を抑えるためのモデル拡張

図 8 ビジネスモデルの変遷

いない．

このモデルは，物理的な制約が緩められるに連れて，最終ユーザを把握することで物流コストを把握できるように拡張された（図 8b）．

次に，売場ごとの在庫を把握し，出荷量の予測を行なえるようにした（図 8c）．

5.2. ATO のモデル記述

有名な Beneton 社の事例のように，最終工程を残して中間製品を在庫しておき，市場の売れ行きを見て最終加工をコントロールするようなビジネスモデル（Assemble to Order）は，プロセス図を使って図 9 のように記述されるであろう．中間製品を在庫しないプロセスとは明確に書き分けられる．

5.3. 生産座席予約

生産計画と顧客注文を割付ける生産座席予約のプロセスを，アセンブリライン図を使って表現する（図 10）．生産のプロセスと受注出荷のプロセスが並行

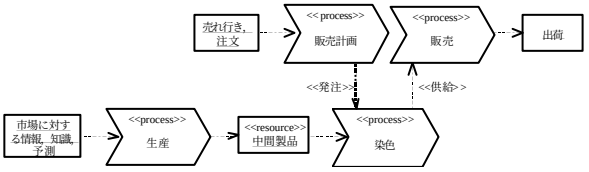


図 9 最終工程を需要に応じて行なうモデル

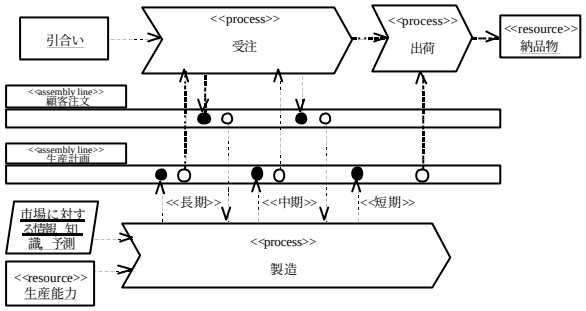


図 10 座席予約のモデル

的に進行し，その間に生産計画と顧客注文オブジェクトが変化していくというライフサイクルが描かれている.

6. 最後に

いくつかの事例について，ここで導入した手法を用いてモデリングしてみた．一応，うなずける結果であるとは思うが，これらはそれぞれ全体のモデルの一部を表しているに過ぎない．全体モデルと部分モデルの関係が整理しきれていない.

一方，これらの手法では十分なモデル化ができない事例もある．たとえば，ある半導体デバイスメーカのビジネスモデルとして, lead user 向け開発を共同で行い安価に納入する一方で，その成果を一般ユーザに妥当な価格で販売するといった事業方針をどうモデル化できるか．新たな流通ルートを開拓することで大きくシェアを拡大した紀伊国屋文左衛門の例，最近のソフトウエアの開発手法としてのオープンソースなどはどのようにモデル化できるであろうか.

また，それらのビジネスモデルがもたらす価値，コストなどをどのように評価するかなど，大きな課題が残されている.

参考文献

[1] 三宅なほみ：「メンタルモデル」，サイコロジー，サイエンス社, No.24, 12-19, 1981
[2] Checkland, P. B. & Scholes, J.，妹尾監訳：「ソフトシステムズ方法論」, 有斐閣, 1994
[3] DeMarco, T, 渡辺訳：「ソフトウエア開発プロジェクト技法」，近代科学社, 1982
[4] Fowler, M.，羽生田監訳：「UML モデリングのエッセンス 第 2 版」, 翔泳社, 2000
[5] UML Notation Guide version 1.3, OMG ad/99-06-08(Part 2), 1999
[6] UML Semantics version 1.3, OMG ad/99-06-(Part 2), 1999
[7] Lipschutz, S.，金井, 清澤訳：「マグロウヒル大学演習―集合論」, オーム社, 1995
[8] Chen, P. P. S.: “The Entity-Relationship Model--Toward a Unified View of Data”, ACM Trans. Database Systems, Vol.1, No.1, Mar 1976, 9-36
[9] Object Constraint Language Specification version 1.1, OMG ad/97-08-08, 1997
[10] 根来龍之，木村 誠：「ネットビジネスの経営戦略―知識交換とバリューチェーン」，日科技連出版社, 1999
[11] Marshall C.: “Enterprise Modeling with UML – Designing Successful Software through Business Analysis, ” Addison-Wesley, MA, 2000,
[12] Fowler, M.，堀内監訳，児玉，友野，大脇訳：「アナリシスパターン」，アジソンウエスレイパブリッシャーズジャパン（現ピアソンエデュケーション），1998
[13] Medina-Mora, R., Winograd, T, Flores, R, and Flores, F.: “The Action Workflow Approach to Workflow Management Technology,” The Information Society, Vol. 9, 199X, 391-404
[14] Jacobson, I., et al，西岡ほか訳：「オブジェクト指向ソフトウエア工学 OOSE :usecase によるアプローチ」，トッパン, 1995
[15] Eriksson, H & Penker, M., “Business Modeling with UML---Business Patterns at Work, ” Wiley, MA, 2000