コンカレントエンジニアリングとビジネス際エンジニアリングアプリケーションの為の

IDEF ファミリー手法

IDEF Family of Methods for Concurrent Engineering and

Business Re-engineering Applications

Richard J. Mayer, Ph.D. Knowledge Based Systems, Inc.
Capt. Michael K. Painter Armstrong Laboratory/ HRGA
Paula S. deWitte, Ph.D. Knowledge Based Systems, Inc.

http://www.idef.com/pdf/ideffami.pdf

以下の題名で、別の資料内にも添付されている

IDEF ファミリー手法の概要と、IDEF を実際に使用する際のガイドライン

IDEF Family of Methods Overview and Practical Guidlines for IDEF Use

（IDEF1 Information Modeling の添付資料）

「新しいシステムを創造する事以上に、計画が難しく、成功が疑わしく、取扱い（management）が危険なものは存在しないことを忘れてはならない。古い体制に利権を持つ全ての人と、そして新しい体制によって利益を得るにも関わらず、ただやる気が無いだけで抵抗する人が、創設者の敵になるのだ」

マキャベリ　君主論：1513 年

はじめに

現在、共同情報管理（Corporate Information Management、CIM）と並行エンジニアリング（Concurrent Engineering、CE）、そしてコンピューター統合生産（Computer Integrated Manufacturing、CAM）の各分野のシステム作成者（system implementor）は、2つの大きな壁に直面している。エンジニアリング組織・生産組織への新しいシステムの導入と言うシステム作成者の主要な責務に加えて、自身の開発プロセスに対しても新しいシステムを導入する必要に迫られているのである。こうした新しいシステム開発プロセスでは、統合フレームワークモデリング手法（integrated framework modering method）を採用しなければならない。すなわちシステムの開発プロセスで、開発と進化（evolve）とを支援する手法、ルール、手続、そして道具によって構成された集合物を使用するのである。このフレームワークは、システム開発のライフサイクルの中で、ユーザーが適切な手法を適用できるように誘導してくれる。この論文の目標は、モデリングの目的に対して、ある種の洞察を提供する事である。特に、システム開発の支援として、モデリング作業の結果を選択し、使用し、評価する必要のある、CIM もしくは CE プロジェクトのマネージャー／エンジニアという立場からの洞察を考慮している。

コンピューターのハードウェアと一部のソフトウェア技術（データベース等）が急速に進化する中、大規模な情報システム開発のエンジニアリングの為の、効率的で、十分にかつ幅広く理解された手法（？understood method）が欠乏し続けている。それに加え、システムの統合と進化までもが要求されたならば、複雑性は圧倒的なものになる。手法支援という確かな基礎が、こうした種類のシステム開発作業において必要不可欠なものとなる。複雑で大規模で、そして進化し統合されたシステムには、個々の特定の目的に対する複数の多様な手法が必要である。その為、効率的な手法の開発が非常に切望されており、多くの手法が開発の途上にある。

手法開発には、二股（two-prong）のアプローチが必要である。1つ目のアプローチの目標は、進化する統合情報システム（evolving, integrated information system、EIIS）を支援する為に必要な手法を識別し、それを開発する事である。この論文の背景（context、文脈）として、CE と CIM を支援する為に必要な、分析手法と設計手法の 2 つに重点を置いている。この手法開発には、手法間での変換能力はもちろんのこと、正確な、数学による、個々の手法の定型化（formalization）の開発も含まれている。

2つ目のアプローチは、大規模な EIIS の計画と分析、そして設計を支援する為の手法の適切な選択、使用、拡張、作成の為の、エンジニアリング規律（engineering discipline）の開発である。つまり、開発された手法は、予測可能な（predictable、当然の）効率を持つ新しいシステムエンジニア手法の開発へと、再帰的に使用されるのである。このアプロ

ーチでは、手法の分析と比較の技術も開発しなければならない。

手法（と、特に手法エンジニアリング規律（a methods engineering dicipline））の開発には、手法作業を基にした、特殊な経験が要求される。手法開発チームには、技術を実際に導入した経験に長けた人だけでなく、手法論（methodization）や言語構文設計（language syntax design）（図形的、語彙的言語の両方）、そして語義学（sematics）の形式モデル（formal models）の各分野で経験を持つ人も、含めなければならない。

語義的視点から見ると、2 つの興味深い手法の観察結果が、互いに否定し合っていることがわかる。1 つ目の顕著な観察結果は、そうした手法は当然に、いかなる論理的、追跡可能なやり方（manner）にも由来していないということであり、そして 2 つ目の観察結果は、それにも拘らず、その手法は「機能している」ということである。手法の開発史をたどることが出来たらなば、手法が恣意的に開発されたもので無い事か判るだろう。いやむしろ、手法というものは、多くの人々によって少しづつ、過大な時間をかけて開発される傾向にあり、ある分野において上手く機能していることが結果的に発見されただけに過ぎないのだ。実際に、その手法が何故機能しているのかという根底に存在している論理的理由が、発見者によって全く理解されていないのである。むしろ、勘（hunch）や直感（intuition）、もしくは偶然的な環境によって、発見に至ったのである。手法の理論的基礎（theoretical basis）が理解されない、もしくは十分に理解されていない、まさにその理由から、手法を強化する事もまた困難となる傾向にある。以下に挙げる逸話からも、論理的理由無しに、誰かにある手法を使うよう説得する事が、本当に難しいことが判る：

ある鉄道製造工場へと新しく入ってきた若い溶接作業者が、貨車の上でコンテナとして使用する大型で木製の樽を支える架台の、加工と溶接を行う作業場に配属された。職場長は、作業指示書（job-instructions）を実際にどのように実行していくかについて長い時間をかけて新人に教育を行おうとしたのだが、溶接の腕を自慢していた新人は、彼にとっては時代遅れのこのやり方にいらだっていた。新人は職場長に対して、彼自身のやり方で、もっと早く仕事ができることを主張した。職場長は職場長で、作業はこの堅実な方法で行うべきであり、そうでなければ正しく出来上がらないと断固として指導した。

納得できなかった新人は、職場長に結果を示すことにした。新人は 1 週間の仕事を 2 日間で仕上げ、誇らしげに職場長へと差し出した。職場長は黙ったまま、架台の一つを手に取ると、給水塔の上に登り、そこから地面へと架台を投げ落した。架台は地面

で 2 度ほど跳ね上がると、溶接部分から壊れてしまった。給水塔から降りると、職場長は黙って職場長のやり方で組み立てた架台を新人に渡すと、給水塔を指差した。新人は職場長の架台を同じように給水塔の上から地面へと投げ落としたが、何度も跳ね上がりながらも壊れることは無かった。

若い溶接作業者が、どうして職場長の手法が機能したのかを理解する事は決して無いだろうが、彼がそのやり方を採り入れ、以後それを誠実に守ったであろうことは間違いない。職場長の独自の手法の実際の本質が何であれ、それが手法というものの典型なのである。何故なら、溶接された架台という分野では、それが「最良の実地経験（best practice)」を示しており、その最良の実地経験は長い間、幾つもの経験を重ねることで得られたものだからである。事実、認知活動もしくは身体活動の分野（domain of cognitive or physical activity）では、手法は、最良の実地経験を要約（encapsulation、カプセル化）することで抽象的に記述されるものである。

手法の本質と重要性（Nature and Importance of Methods）

ある手法の持つ目的は、人の意思（human mind）によって手法が使用されることにより、実現化される。シャベルは、それそのものだけでは穴を掘ることが出来ないが、人が穴を掘る際に梃子効果（leverage）を与えてくれる。それと同様に、手法は人の意思に梃子効果を与え、作業をより効率的に実施させてくれるのである。手法は、人間の知的活動を補助し動機付けしてくれるが、しかし決断を行ったり、洞察したり、問題を発見するといった事はしてくれないのである。

手法の本質を作成者（creator）としての立場ではなく、実現者（？enabler）の立場として認識する事は、手法の重要性に対する認識を薄くしてしまうことにはならない。前章に挙げた逸話のように、熟練者（expert）から非熟練者（novice）へと、最良の実地経験から得られた知識を伝える事は容易でない。手法の基本的な重要性は、長い間、製造業において認識され続けてきた。労働力（manpower）、手法（method）、資材（material）、機械（machine）、そして資金（money）から成る生産の「5M」の中でも、手法は際立ったものになっている。資材と機械、そして資金でさえ、他の物に置き換える事が可能であるが、労働力と、その労働力の知識を拡大する手法の 2 つは、工業における極めて重要な構成物なのである。

手法の構成（Components of a Method）

非公式な見解として、手法は何かを行う為の手続きである。すなわち、手法は「最良の実地経験」もしくは経験を捉えようと試みるのである。それに加え、こうして得られた手続きを、より効率的に伝達する（communicate）具象的表記法（？representational notation）を持っている。

より公式な見解では、手法は図 A-1 に描かれているように、3 つのものによって構成されている。全ての手法は、定義（definition）と規律（discipline）、そして幾つかの使用（use）で構成されている。定義は、概念と動機付け（motivation）、そして手法の背後にある理論（theory）によって構成されている。規律には、手法の構文（syntax）、コンピューター変換フォーマット（computer interpretable format）（図 A-1 内では ISyCL 構文として書かれている）、そしてその使用を統括する為の手続を含んでいる。多くの手法は複数の構文を

持っており、時間の経過による進化の過程や、異なった局面の中で、それぞれ使い分けられている。恐らく、手法の中の、最も明白な構成物は、規律に関連した言語である。多くのシステム分析手法とエンジニアリング手法は、図形的構文（graphical syntax、箱や丸や矢印による表記）を使用する事により、不明瞭な状態の重要な情報の中からデータを収集し、視覚的に提示するのである。そして手法の使用は単品で行われるか、もしくは一組の手法の一部として行われている。

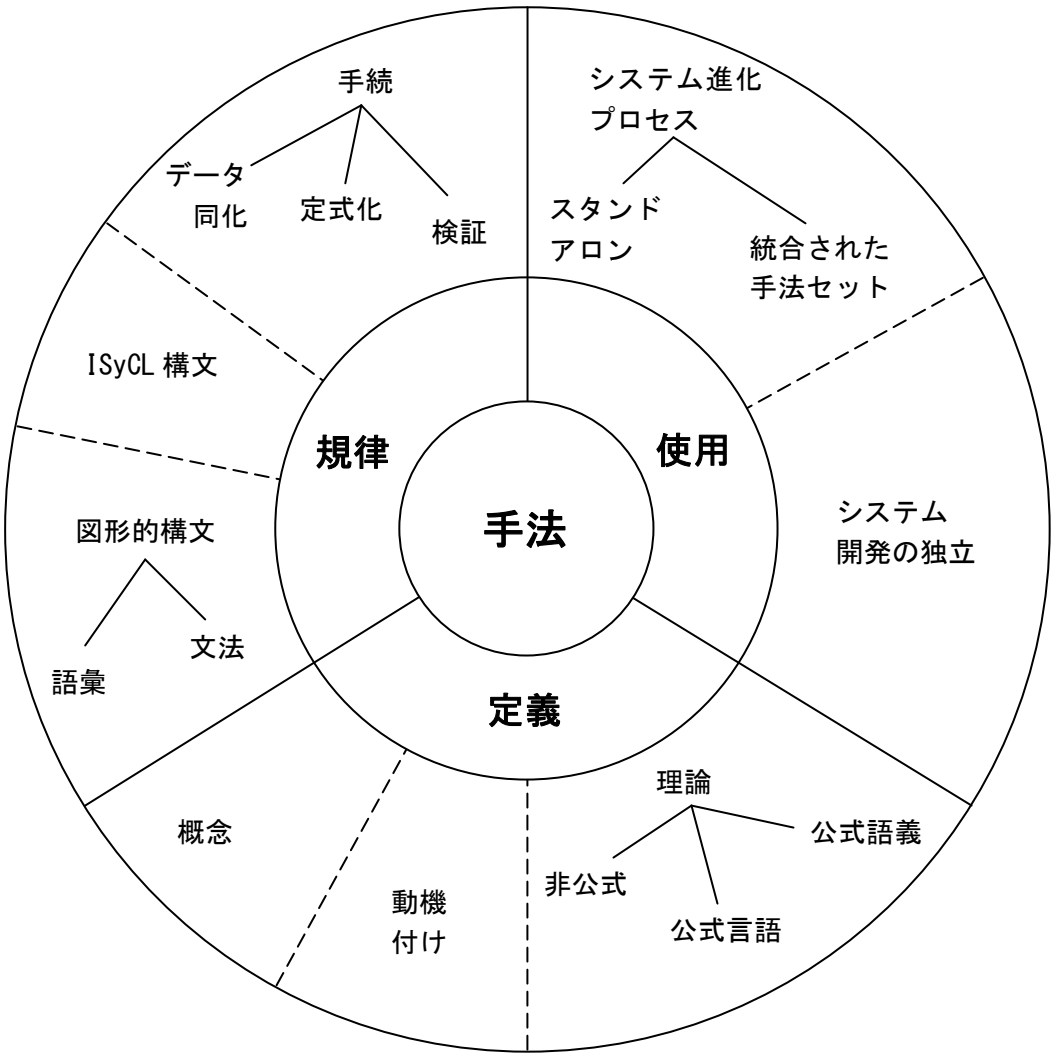


図 A-1　手法の構成

手法の型式（Types of Methods）

EIIS（進化する統合情報システム）という意味合いでの手法は、殆どが「モデル」を作成する為の手法のことであるが、中には「記述物（descriptions）」を作成する手法も含まれている。この、モデルと記述物は、どちらもダイヤグラムと説明文（text）によって構成されているという点において、似た物になっている。モデルは、対象となる現実世界のシステムの特徴を模倣するために、ある関連した視点の下、明確な構造の枠内で設計された、対象物と特性、そして関係性から成る理想化されたシステムとして見なす事が可能である。モデルの力は、表現対象の現実世界のシステムを単純化し、そのシステムについての事実を、それと一致しているモデル内の事実から予測する能力にある。従ってモデルは、それ自体が（in its own right）設計されたシステムであり、インスタンス（instance）という抽象化されたシステムにより、ある状態を満足させるように強制されている。

（訳注：現実世界を全く正確に表現する事は不可能であると言う意味で）モデルは不正確であるのは当然であるが、対象となる領域の実際の特質について、信頼性のある予測因子（predictors）が得られる程度までは正確でなければならない。一方の記述物は、世界についての事実もしくは信念の記録物であり、そして記述物はそれ自体が、一般的に不完全なものである。なぜなら、記述物を作成する人が無関係に見えた事実を省略してしまうかもしれないし、システムを記述する過程の中で事実を書き忘れてしまうかもしれない。その為、記述物が正確であるべきものだとしても、抽象的で検証可能な、満足させるべき状態によって強制されていない。

モデルと記述物を、概念的（conceptually）に区別する事が重要である。不幸なことに、「モデル」という用語は、一般的に記述物とモデルの両方の意味に、曖昧なまま使用されている。このことは、一部の手法にのみ見られるものではなく、幾つもの手法に共通なモデリング活動について議論を行う際に、非常に多く見受けられる。この論文では、IDEF3 と IDEF5 のような記述的手法と、IDEF0 と IDEF1 のようなモデリング手法を含む、IDEF ファミリーについて説明して行く。この論文内の「モデル」、「モデル作成者」、そして「モデリング」という用語は、最も一般的な意味におけるモデル（モデルと記述物の両方について言及したもの）と、より狭義の意味でのモデル（記述物に対するモデル）の、両方を意味している。その為、用語を使用する際の文脈（context、背景）を説明し、どちらを意

味しているのかを明確する。

システム開発プロセスにおける手法（Methods in the System Development Process）

モデル作成者は殆どの場合、モデリング活動の不足は許容できないと主張するシステム開発者と、モデリング活動が法外に高価だと主張する出資者（funding sourece）の板挟みに遭っている。しかし実際には、システムモデルやシステム記述の作成に必要な、時間、労力、費用を正当化する、多くの理由が存在する。システム開発プロセスのモデルから、システムモデリング作業が、どこに適合するのかが判る。図 A-2 は、システム開発に関連する各種活動と活動間の関係を、顧客視点で描いたものである。この図では、何のシステムであるかということは重要ではなく、暗視ゴーグルや情報システム、更には家のようなものでも構わない。

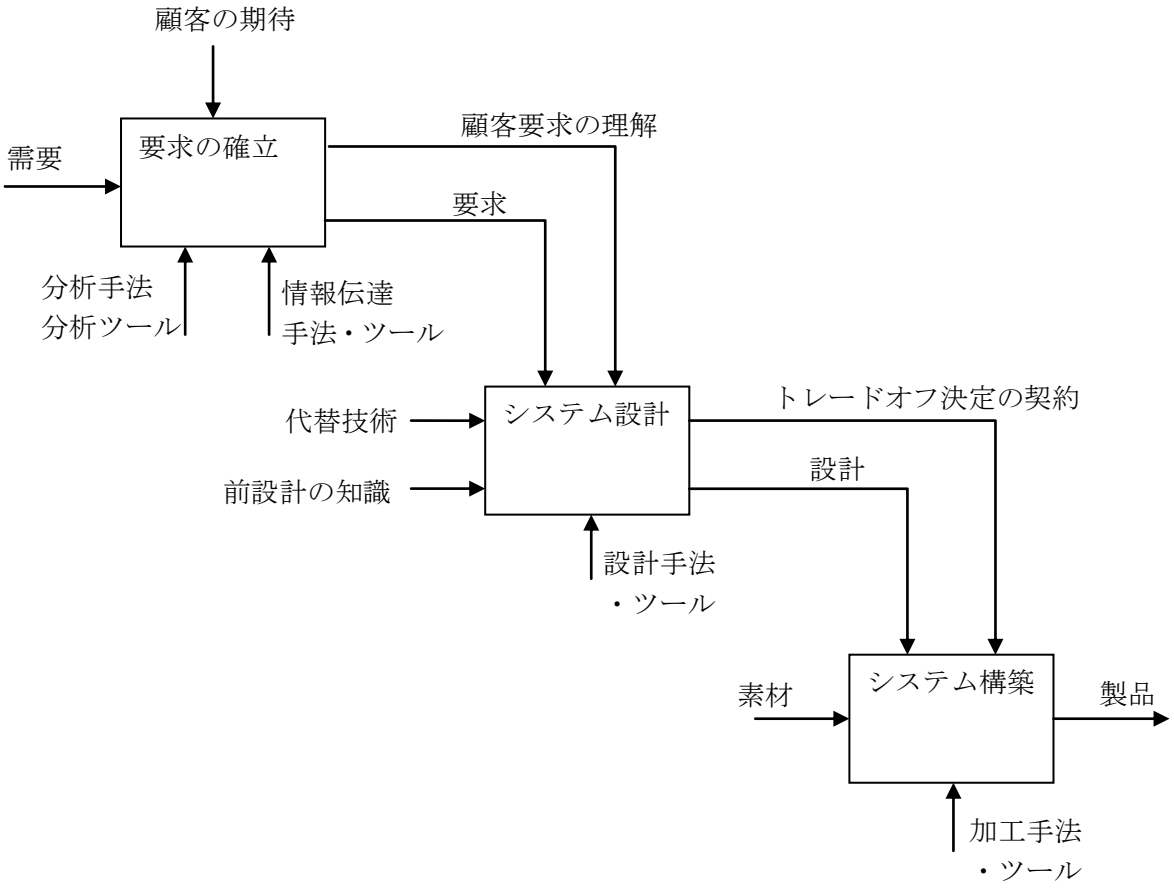


図 A-2　システム開発（IDEF0 モデル）

「要求の確立」（Establish Requirements）という活動において、顧客は需要という形で入力を行う。この時点で、顧客が環境内に問題を認識していても、一般には本当の原因を理解できておらず、実際に症状を誤診しているものである。この「需要（needs）」とは、問題が解決されることにより満足されるべき状態である。システム開発者（system developer）は、この満足させるべき状態の上に、制限の強制（limiting constraints）を確立してゆかなければならない（すなわち、要求として）。例えば、新しい家を設計する建築家（architect）は、「より広く、より暖房費が少ない」と表現された需要を受けると、要求を決定してゆく為に質問を行う。

・土地の広さは？
・どのような種類の空間が他に必要ですか？　倉庫？　寝室？
・暖房費の許容範囲はどのくらい？
・この家に幾ら支払いますか？

この活動の結果、（１）明確な要求と（２）環境が十分に理解され、開発者によって顧客の需要を叶えてもらえるという顧客の認識（perception）とが得られる。この認識は、開発者が問題（symptom、症状）の発生した環境を注意深く調査する事により、問題の原因を隔離しようとした試みの、副産物である。この分析により、顧客が認識している問題を再発見するだけでなく、他の既存もしくは潜在的な問題が識別されることになる。

顧客の承諾がある限り、開発者は、目標を達成し、コストを引き下げ、時間を短縮するものであれば、どのような分析手法も分析ツールも使用可能である。この意味合い（context）において、ツールとは、規定された手法の使用を（通常は自動的に）支援してくれるものである。事実上、手法によって生成される物の多くを、顧客は直接見ることができない。その為、定期的にシステム開発者が実際に何を考えているのかを、顧客へと伝える必要がある。それにより、顧客に対して「これは、あなたの意味するもので合っていますか？（Is this what you mean?）」という質問を行わずに済むことになる。

2 つ目の「システム設計（Design System）」という活動は、明確な要求のセットと、顧客の期待が満足されるという感触の、両方が揃わなければ進められないものである。そこでシステム開発者は、要求を満足する最適な設計手法と設計ツールを使用することになる。設計活動の中で顧客との情報伝達を行うために、システム開発者が情報伝達の為の手法とツールを使用する場合もある。例えば設計者（architect）は、顧客と相談すること無く、

要求からたたき台（blueprint）を作成しても良い。しかし、トレードオフ決定を行わなければならない際には、引き続き顧客と関係を結ばなければならないことが多い。この場合、システム開発者は、制約（constraints）（コスト等）の用語により、対立している設計の選択肢（design decisions）の与える影響を表現する。こうしてトレードオフ決定が作成される事により、暗黙的な口頭による合意（verbal agreement）もしくは、明確的な公式の署名契約（formal sign-off）へと至るのである。

設計が完了し、トレードオフが顧客によって受け入れられると、システム構築が開始される。このプロセスにおける製作（fabrication）の手法とツールは、厳格な手作業でのアプローチから、全くの自動的なシステム作成アプローチに至るまで、細かな間隔で豊富な品揃えで構成されている。

システム開発プロセスの一環として構築されるモデルは、少なくとも（１）顧客に対して、システム開発者が顧客の環境や需要、そして期待やシステム要求を満たす為に必要な状況を、それぞれ理解してくれているという安心感を与え、そして（２）顧客をトレードオフ決定とそのドキュメントの作成に、巻き込んだ物でなければならない。顧客の視点でのモデリングの重要な目的の一つは、こうした需要を満足させることにある。つまり、開発者がこうした期待を満足させないモデルを構築してしまうと、それが顧客の需要に合致することは、まず有り得なくなるのである。そうならない為にも、どういう種類のモデルが、どれだけ必要かを知る為の補助具として、こうしたガイドライン（上の（1）と（2）のことか？）を用いるべきである。

顧客の視点からのモデリング目的をより完全に理解するために、CIM システムと CE システムの開発を支援するモデリング活動を実行する、IDEF 手法（Integrated Computer-Aided Manufacturing（ICAM） DEFinition、統合的コンピューター支援生産定義、とでも訳すのか？）の実使用例（experience）を中心にして、説明を行ってゆく。まずは IDEF0 機能モデリングと IDEF1 情報モデリング、そして IDEF1X データモデリングの 3 つの IDEF 手法の実例を詳細に記述する。それに続いて、IDEF3 プロセス記述獲得（Description Capture）、IDEF4 オブジェクト指向設計、IDEF5 存在論記述（Ontology Description）、そして IDEF6 設計理論的根拠獲得（Design Rationale Capture）という最近出てきた（emerging）IDEF 手法についても紹介し、それらが想定する、CIM の実装の為の能力について説明して行く。

IDEF0 を使用した、機能モデリング（Function Modeling）

IDEF0 機能モデリング手法は、組織もしくはシステムでの決定（decisions）や行動（actions)、そして活動（activities）（訳注： action には決定や活動の意味も含まれている。決定、行動、活動と訳し分けてはいるが意味が無い。決断から行動に至るまでの一連の流れ、とでも採るべきなのか）をモデル化する為に作られたものである。IDEF0 は、構造化分析・設計技術（Structured Analysis and Design Technique、SADT、[Mayer 90]）として知られている、十分に確立した図形的言語（well-established graphical language）に由来している。空軍は、システムの機能的視点での分析と情報伝達を行う為の、機能モデリング手法の開発を SADT の開発者に委託した。効率的な IDEF0 モデルは、システム分析の体系化（organize）と、分析者と顧客との間の効率的な情報伝達の促進を、それぞれ補助してくれる。それに加え、IDEF0 モデリング手法は、ある特定の機能分析の為か、もしくは別のシステムの見地からの未来分析（future analyses）の為の、分析範囲（scope of analysis）を設定（establish）してくれる。情報伝達ツールとしては、IDEF0 は単純化された図形的機構（graphical devices）を用いる事により、その領域の専門家の関与と同意形成とを強化する。一方の分析ツールとしては、その機能を実行する為に何が必要かとか、現在のシステムが正しく動くとどのようになるのか、正しくなければどのようになるのかといった、という機能の実行の識別作業において、モデル作成者を補助してくれる。その為、システム開発作業における最初に行う仕事の 1 つとして、IDEF0 モデルが作成されていることが多い。

IDEF0 の視点からのモデリングシステム
（Modelong Systems from an IDEF0 Perspective）

IDEF0 には、組織における決定、行動、活動のモデル構築のための、プロセスと言語が含まれている。IDEF0 手法を適用する事により、活動と活動間の関係性とを、時間軸と組織とは無関係な形式で、体系化して表現することが可能である。IDEF0 は、組織が行っている事を、ユーザーが 1 つの流れとして表現（to tell the story）できるように作られている。レシピやプロセスといったものの詳細（specification、仕様）作成を支援するものではない。そうした活動に関係する特定の論理（logic）やタイミングの詳細な記述は、IDEF0 ではなく IDEF3 プロセス記述獲得手法（Process Description Capture Method）で行う。IDEF0 モデルは、組織部門に跨る共通の機能スレッド（functional thread、糸、流れ）を

識別し、組織から独立した分析を促進してゆく事で、「組織（organization）」から「機能（function）」を分離している。

IDEF0 は、幾つものアプリケーション分野において、分析ツールとしても、情報伝達ツールとしても使用されるようになった。図 A-2 を振り返って見ると、この特性は、IDEF0 を要求確立（Establish Requirements）活動を実行する為の機構（mechanism）として、適用可能な事を示している（訳注：下図）。それに加え、IDEF0 の持つ情報伝達促進能力は、CIM や CE の主導（initiative）に必須な他分野間協調チームプロジェクト（cooperative interdisciplinary team project）の為の、効率的な分析ツールとなっている。

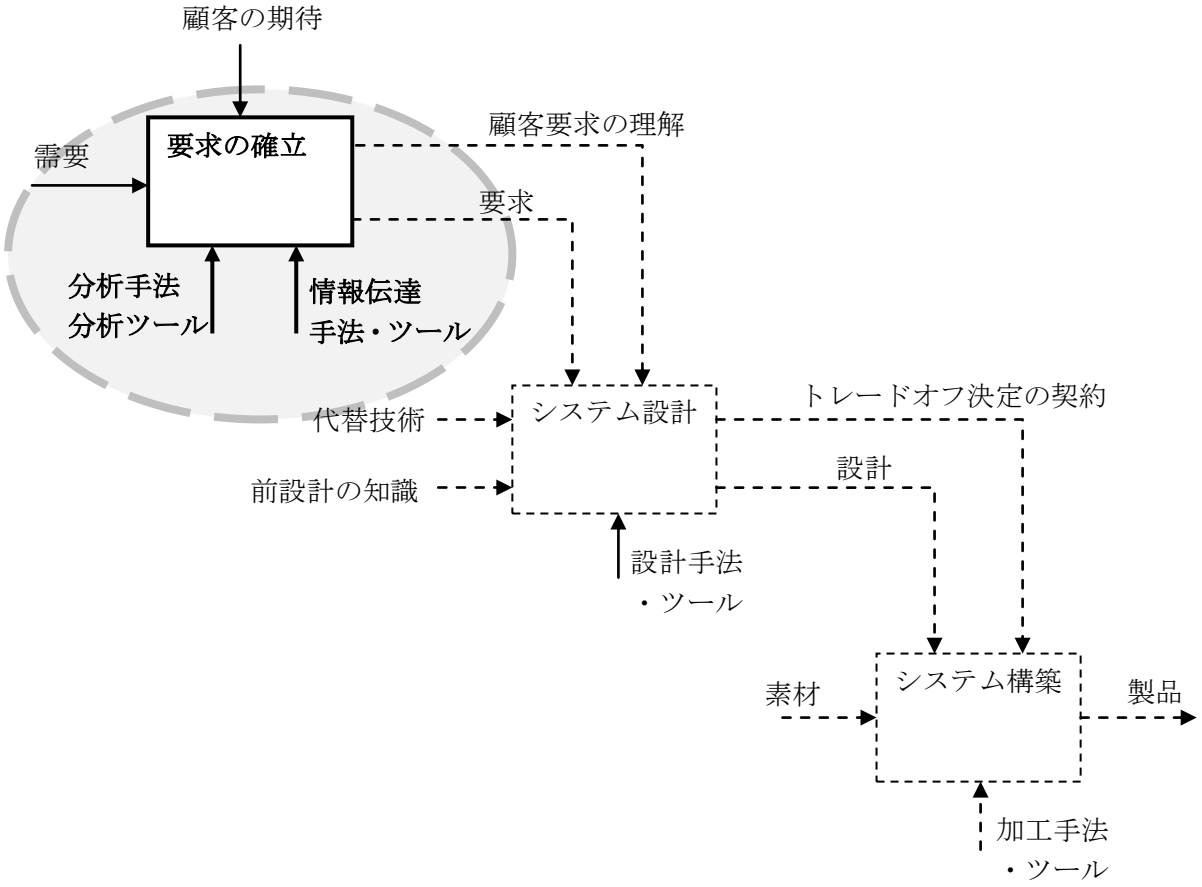


図 A-2b　分析ツール・情報伝達ツールとして使用可能という特性を利用可能な機構の範囲（訳注）

IDEF0 の、体系化された構成と戦略

（Organizational Structures and strategies of IDEF0）

IDEF0 手法に組み込まれた多くの体系的戦略（？organization strategy）により、情報伝達が非常に表現豊かに、簡単になる。しかし不適切に使用されたり、理解が不足していると、モデルは分かり難いものになるか、もしくは不合理な宣言を、あたかも根拠のある物のように誤らせてしまう原因となる。IDEF0 の体系化戦略の例としては、（１）目的と視点、背景（context）のそれぞれの宣言（statement）と、（２）階層構造（hierarchical）的分析もしくはトップダウン分析によるモデル開発のアプローチ、そして（３）抽象化レベル（level）、といったものがある。

モデル作成者にとって、こうした体系化戦略により、モデルの一部分に仕事を集中し、分析対象範囲の境界を明確にすることが可能となると共に、顧客にとっては、システム内の最も関係深い部分を素早く発見し、監査する事が可能となる。また、システム全体の学習や、システム内でのモデル作成者の理解範囲の情報伝達に便利な、ブラウジング機構も持っている。この後、IDEF0 手法による体系化戦略としての目的、視点、背景の記述の使い方について触れて行き、その後に、モデル開発での階層的もしくはトップダウン分析アプローチと、抽象レベルの観念について説明してゆく。

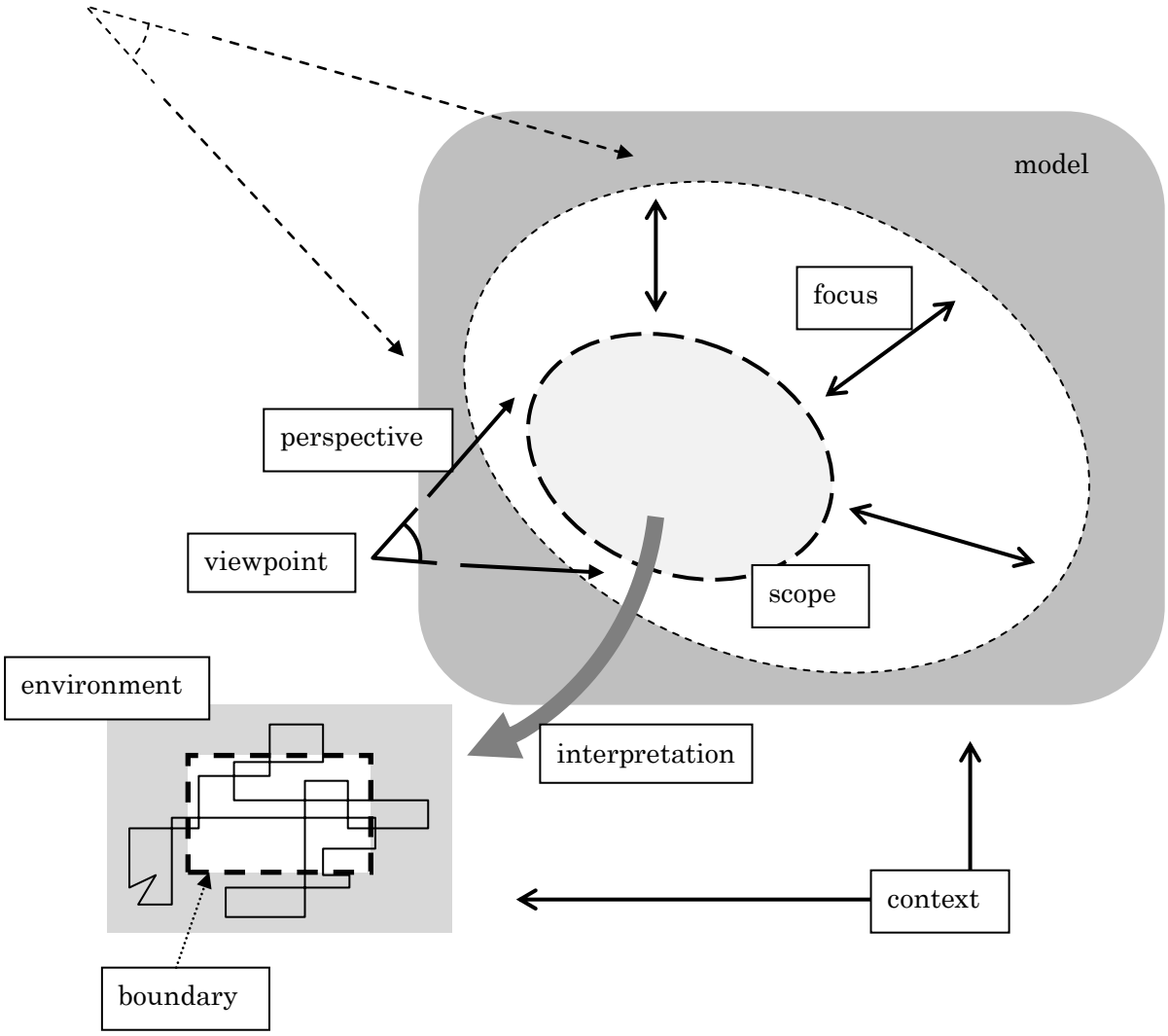
IDEF0 モデリング作業を開始する為に、モデル作成者はまず、そのモデルの「目的」が何なのか、どういう「視点（viewpoint）」から記述を成形してゆくのか、そしてどういう「背景（context）」を持つのか、をそれぞれ決定し、明確に記述しなければならない。目的とは、モデリング作業の目標を宣言（statement）したものである（どういった情報を組み込む必要があるか、その情報はどのような決断を支援するよう提案されているか、どのような同意が必要か、等）。例として挙げるなら、「ある IDEF0 機能分析の目的の１つは、新しい CIM 戦略の下で、既存の複数の機能を統合整理するための機会を識別すること」というような感じである。そして目的が認可されることにより、モデリングチームは評価基準（criteria）を完全なものとすることが出来る。つまり、目的が固まるということは、モデルが完成するということなのである。

視点の宣言は、モデルをいつ構築し、批評し、解読すべきかという展望（perspective）を記述している。この視点により、読解者（reader）がモデルをどのように解釈するのかということと、モデル作成者が調査対象のシステムに含まれる活動の、理想化もしくは抽象化をどのように制限するのかということとが、確立されることになる。視点の宣言が認可されることで、モデリングチームはモデルの対象範囲（scope、視野）と詳細化の度合い

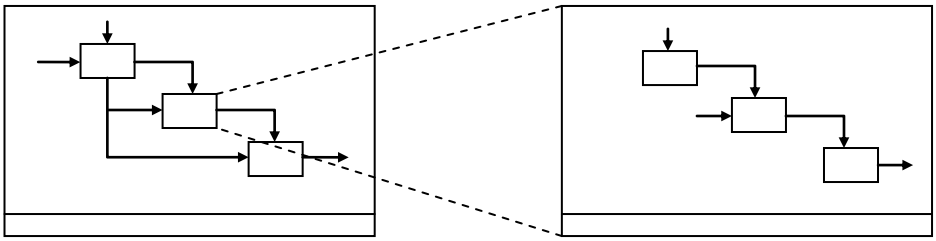
とを管理する機構（mechanism）を持つことが出来る。

背景（context）は、より大きな範囲（larger scope）の一部としての、モデルの解釈（interpretation）と対象範囲（scope）とを確立する。この焦点（focus）により、モデルの、環境（environment）に対する境界（boundry）が作成される。

（訳注：今一つ整理し切れていない語群。下図はこんな感じ？というスケッチ）



IDEF0 モデルの開発を体系化する、もう一つの戦略は、活動の「階層構造分解（hierarchical decomposition）」という観念（notion）である。IDEF0 モデルは、階層構造もしくはトップダウン（top-down）アプローチを用いて開発されているが、SADT を創り上げた Doug Ross は、こうした用語にしばしば混乱していたと考えられている。実際に、Ross はアウトサイドインアプローチ（Outside-in approach）にすることで、このプロセスをより正確なものにすることができると提案している[Ross 85]。結局のところ、IDEF0 モデルの箱（box、IDEF0 では「機能」を表す）は、幾つかの活動の周囲に引かれた境界線（boundary）を表している。その箱の中を見る事で、上階層では箱であるものが、小さな活動へと分解されていることを発見することになる（訳注：下図のような関係）。



この階層構造により、分析者は活動の分解によって表される境界線内のモデルに対象を絞ることができる。顧客もまた、この体系化戦略により、その時点で不要な複雑さを視界から遮蔽しやすくなる。そして遮蔽した部分も必要となれば、分解された箱の中を見ることで、要求された詳細レベルで内容を理解する事が可能である（図 A-3）。

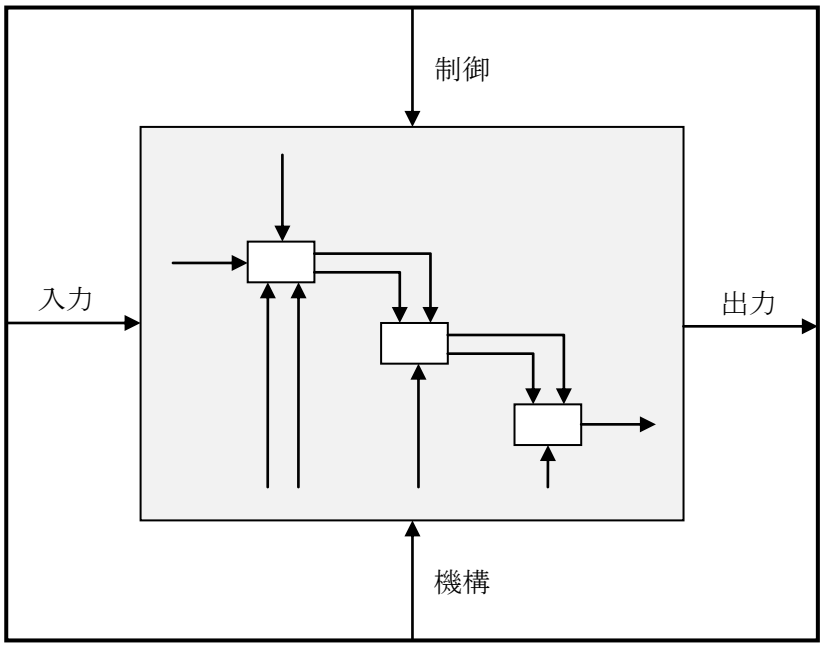


図 A-3　アウトサイドイン

この複雑性の遮蔽（hiding）は、IDEF0 で使用されている抽象化メカニズムの特徴の 1 つだが、一般的には、この階層的な抽象化が「活動」のみに適用されるものだと誤解されているが、モデル上にある矢印もまた、異なった抽象化階層毎に整理されている。実際、箱の抽象化レベルと、その箱に付属する矢印の抽象化レベルとの間のバランスを取る事は、時には困難な場合すらある。例えば、図 A-3 にある内側の箱の中には合わせて 4 本の機構矢印（IDEF0 の箱の下辺に付く矢印）があり、それぞれの活動を実行する為に使用される異なった種類のツールを表現している。この 4 本の矢印は、より抽象的見方によって「ツール群（tools）」という名前の 1 本の矢印へと束ねる事が可能である。そうすることにより、図 A-3 にある外側の箱は、その箱の抽象化レベルとして 1 本のみの機構矢印を持つことになる。同じモデルの、より抽象的な階層と、より詳細的な階層の両方で、4 本の矢印がそのまま描かれているのは、余りにも不細工である。モデル作成者が、IDEF0 で利用可能な情報遮蔽構成を効果的に使用しているかどうかを簡単に見分ける 1 つの方法は、全ての階層の各箱に対して、何本の矢印が付与されているかを数えて行く事である。もしもモデル内で階層毎に矢印の数が偏ったものになっていれば、矢印の集束に使用されている抽象化レベルと、活動の抽象化レベルとが合っていない可能性が高いことなる。
（訳注：この辺は、IDEF0 の詳細な論文を参考のこと）

その抽象化レベルでは不要な詳細さを遮蔽する為の IDEF0 構造の中で、最も理解が難しく、かつ誤用されることが多いものが、矢印の集束（bundle）と解束（unbundling）の観念である。図 A-3 の、いずれも同じ抽象レベルにある、構成ツールという名前の付いた 4 本の機構矢印の内の、2 本だけを解束することは論理的に矛盾する事になる。同様に、同一の抽象レベルある矢印を、同じ抽象レベル内で集束したり解束したりすることも、殆ど意味を持たない行為である。残念なことに IDEF0 文法（literature）では、矢印集束の情報遮蔽構成の誤用による論理的矛盾を適切に回避するための方法にまでは、十分に対応し切れていない。最適なアプローチ方法は、奨励される構造に強制しれくれる自動支援ツールを使用してモデルを作成するか、IDEF0 モデル作成者に、集束問題をどのように認識し回避するかを、数年かけて学ばせることである。

IDEF0 ガイドライン

IDEF0 モデル作成者が待望するヒューリスティック（heuristic:プログラムを行う際の、独学的なアプローチもしくはアルゴリズム、裏技）の膨大な知識ベースが存在する[Cullinane et. al. 90、KBSI 90a、Mayer 89b、Softech 81a]。次の段落から、IDEF0 モデリング課題（exercise）で最大の成果を得るための、最も一般的なヒントの幾つかを説明して行く。特に、このガイドラインにより、これまで簡単に説明してきたモデリングの目的をモデルがどのようにして上手く満足させているのか、理解しやすくなるだろう。

IDEF0 モデリングを習得する上で最も困難な部分は、モデル上の異なった階層間で、統一された矛盾の無い目的と視点とを維持し続ける事である。この作業を困難なものにしている原因は、視点の移動の認識が、困難な事にある。この事に向いた良い裏技（good heuristic）は、モデル作成者が行っている分解の範囲内の境界線を注視し、「この活動は、上階層の活動のスコープ内に収まっているか？」とか「この活動は、モデルの確立された視点と目的に一致しているか？」といった質問を、定式的に繰り返して行くことである。

次に、手法の数値的拘束を進めるモデルを探す事である。例えば、IDEF0 手法の規律として、分解の際に作成する活動の数は 3 個以上 6 個以下にすべきであるというルールが定められている。同様に、1 つの活動箱の 1 辺に、6 本以上の矢印を付けないというルールもある。モデリングで起こり易い失敗は、分解を行う際に 7 個目の活動が必要不可欠な物だと判断してしまう事である。分解を行う際に、取り敢えず 6 個の箱を描き、その全てに名前を付けようとしてしまう（誤った）傾向もある。もう 1 つの失敗として、箱とその間の矢印が、まるで電子基板のように複雑な網目になってしまうことである。こうした失敗に至る大きな原因は、関連する活動箱と同じ抽象化レベルになるように矢印を集束するという、矢印の論理的組織化に失敗しているためである。また、確立しているルールを固守していても、不適切な集束が発生する事がある。こうした種類の間違いは、独断で矢印のグループ化が行われた時に良く見られる。

新しい協定（convention）を既存の手法へと導入すると、問題が発生する事がある。例えば、数名のモデル作成者が、入力と出力をデータ要素（訳注：data element、コンピューター用語で、正確に意味を表現可能な（認識可能な）最小のデータ単位、もしくはそれを示すコード。例として、「名前」、「性別」、「都市名」、「識別番号」等。整数・文字列等の要素種類とデータサイズによって、それぞれ定義される）のみに限定するという新しい協

定を追加しようとしている。このような制約を課すことにより、入力と出力を情報モデル要素へと直接に変換可能な状態にすることが目的である。直感的には、このアプローチにより、活動間のデータ需要の追跡が容易になるだけでなく、その情報モデルの対象範囲（scope）を明確に描写できるように見える。しかし、このアプローチでは、活動に割り当てられているリソースに機構（mechanism）を追加するという協定変更を、モデル作成者が強制されることになる。（入出力をデータ要素に制限するという新しい協定に伴い、不可能となってしまうモノを補完する為の機構・メカニズム、という意味？）例えば、「壊れた飛行機の修理（Fix Broken Airplane)」という活動について考えてみよう。こうした（入力と出力をデータ要素のみに限定するという）協定の下では、活動に対する入力としての、たった 1 ヶ所の場所（there is only one place）だけが、壊れた飛行機の為に存在している。しかしこのアプローチでは、出力もデータ要素だけという協定から、活動の結果として現れた物が、修理済の飛行機であることを示すことができる場所が存在しないのである。（この協定では、壊れた飛行機の修理という活動を表現できない）そこで、（いきなり入出力をデータ要素に限定してしまうのではなく、）この場合では、取り敢えずは既存の協定を使用した後に、入力と出力を使い、何がデータ要素の候補と成り得るのかについて調査と区別とを行う方が、（新しい協定の追加が）断然に容易になる。

訳注：

以上の段落が、全く意味不明（丸 2 日考えたが、判らないので諦めた)。判り易いようにと例を挙げているが、言葉足らずで何が言いたいのか判らない。それともこうした世界を十分に理解している人なら、この説明で十分に理解可能なのだろうか？

IDEF0 で、しばしば誤解されててしまう協定（convention）の 1 つとして、IDEF0 手法の中に織り込まれた、非時間的な暗黙の概念がある。前にも説明したように、IDEF0 では活動間の時系列的制約（time-ordered constraints）を明示的には捉えてはいない。実際、表現をより強く、一般的なものにする為に、IDEF0 には時間的論理（temporal logic）は意図的に含められていない。IDEF0 は、時間順に並べられたプロセス（time-ordered process）の境界内部で運用管理される活動セットの記述も可能ではあるが、分析を目的とする場合には、どのような時間順プロセス内の活動セットを通る全ての経路（path）を、少なくとも分析と関連のある全ての経路を、説明可能な一般化モデルを提供した方が、より一層便利なのである。

このようにモデルは、ある程度（in part)、有り得そうな活動順序もしくは発生し得る活動順序に順応する度合いによって判断されるべきであり、特定の手順を暗示する意図の下で

モデルを作成すべきではない。

訳注

IDEF0 の本文にもあるように、IDEF0 のダイヤグラムには、順序や時間といった概念が含まれていない。矢印が描かれている為に、順序や時間といった概念を見てしまいがちになるが、あくまでも矢印は箱（機能）間に存在する制約関係を示しているだけである。その為に[箱１]→[箱２]とあっても、それは「箱１の出力の影響を、箱 2 が受ける」という制約関係は示しているだけで、「箱１の後に箱２」という順序関係を表しているわけではない。箱１が実行されないと箱 2 を実行できないような順序が関係するフローであっても、十分な回数を繰り返し、十分な時間をかけて安定したフローとして読まなければならない。

IDEF0 モデルの品質を評価する為の最も便利な試行（exercise）は、1 ページ当たり 2 分以内という制限下でモデルを読み、内容を理解可能か判断するというものである。大きなモデルであっても、2 時間以内でなければならない。もし、IDEF0 表現によりモデル化された環境を、その時間内に読者が理解し、その環境内で何が起こっているのかを説明する事が可能なように思えたならば、そのモデルは価値がある物と見なすことができる。全体を注意深く調査する為に 2 日も必要となるようなモデルは、IDEF0 モデルとしては失格である。

IDEF1 を使用した、情報モデリング（Information Modeling Using IDEF1）

図 A-2（？）を見ると、IDEF1 は、要求を作成する際の分析並びに情報伝達の手法であると見なされている。しかしこの場合、IDEF1 は、企業によって（by enterprise）どのような情報が運用管理されているか、もしくは運用管理されるべきかという要求を作成しているのである。

CIM アプリケーションにおいて、IDEF1 は一般に以下の用途に使用されている：

（1）組織において現在、何の情報が運用管理されているのかの識別
（2）需要分析の最中に認識された問題の内、適切な情報の運用管理の不足が原因のものはどれなのかという識別
（3）「あるべき（TO-BE)」CIM の適用では、どのような情報が運用管理されているのか、を明確化する

IDEF1 情報モデリング手法は、ヒューズ航空機（Hughes Aircraft）によって開発された ELKA（Entity-Link-Key-Attribute、実体・リンク・キー・属性）と、Peter Chen によって提唱された ER（Entity-Relationship、実体関係性）手法、そして Codd の関係性モデル（Relation Model）の 3 つを基にして作られたものである（図 A-4）

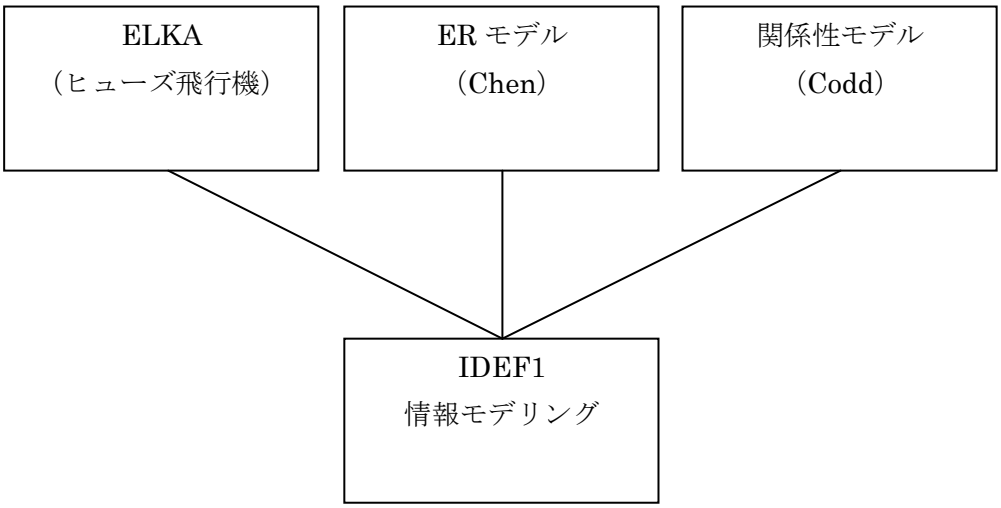


図 A-4　IDEF1 の源流

IDEF1 が作られた元々の意図は、企業としての視野（scope）で、対象物（object）についてどのような情報が存在しているか、もしくはどのような情報を運用管理すべきか、ということを把握する為のものであった。IDEF1 の考え方における情報システムには、自動化システム構成物（automated system component、コンピューター上のアプリケーションやシステム）だけではなく、自動化されていない人やファイル棚、電話といったものも含まれている。IDEF1 はデータベース設計手法として特化されたものではない。IDEF1 の開発が行われていた頃、データベース業界（community）では、情報リソース運用管理の需要と要求を、分析・記述（state）する為の手法が必要であると信じられていた。IDEF1 はそうした意図の下で開発されたのである。IDEF1 は設計手法であるというよりも、以下に挙げるようなものを識別する為に使用する、分析手法であると言える：

（1）企業によって収集、保存、運用管理される情報
（2）情報の運用管理を統括する（governing）ルール
（3）情報に反映されている企業内部の論理的関係性（logocal relationships）
（4）情報運用管理の失敗により発生した問題

情報分析の結果は、企業の戦術・戦略計画の担当者により、競争力の向上と、情報資産の有効活用に使用される。このような計画作業には、企業で利用可能な情報を効率的にしてくれる自動化システムの設計と導入も含まれている。IDEF1 モデルは、自動化システム設計の決定作業を支えてくれる。また IDEF1 は、データベース設計で使用されているというよりも、むしろ運用管理者（manager）に対して、優れた情報運用管理ポリシーの確立に必要となる見識（insight）と知識（knowladge）を提供する為に使用されている。次節では、IDEF1 の基本コンセプトと幾つかのルールについて、大まかに説明してゆく。この手法についてのより詳細な内容に興味を持った読者は、論文[Softech 81b, KBSI 90b, Mayer 89b, Mensel 89]を参照して欲しい。

IDEF1 の考え方に基づいた、モデリングシステム

（Modeling System From The IDEF1 Perspective）

IDEF1 では簡単な図形的制約（convention）を使用する事で、モデル作成者が以下のものを**区別する**為の強力なルールセットを、補強している。

①現実世界の物体（real-world objects）　→知識エンジニアの領域
②現実世界の物体間に結ばれている、物理的もしくは抽象的関連性（association）
→知識エンジニアの領域
③**現実世界の物体について運用管理されている情報**　→IDEF1 の対象領域
④情報の獲得、適用、運用管理の為に、情報表現に使用されているデータ構造
→ソフトウェアエンジニアの領域

簡単にまとめるなら、IDEF1 は、企業によって収集され、運用管理され、管理され、そして究極的には対価が支払われている、もしくはそうあるべきである情報を、表現する為に作られたものであると言える（③）。手法に設けられているルールは、項目①と②のモデリングを防ぐ手助けとなっている（①と②のモデリングは通常、知識エンジニア（knowledge engineers）の領域であると考えられている）。そしてまた、モデル作成者の注意をデータベース設計から逸らせている（④は通常、ソフトウェアエンジニアの領域であると考えられている）。

モデル作成者が情報要求を決定する際に重要となる、2 つの領域（realm）が存在している。1 つ目の領域は、組織内の人間によって認識（perceive）されている現実世界である。この領域には、物理的もしくはコンセプト的な対象物（object）（人、場所、物、アイデア、等）と、こうした対象物の特性（property）、そしてこれらの対象物間の関係性とが含まれている。2 つ目の領域は、情報領域である。この領域には、現実世界にある対象物の情報イメージが含まれている。この情報イメージとは、現実世界の対象物そのものではなく、現実世界の対象物に関して収集、保存、運用管理されている情報のみを指している。IDEF1 は、この情報イメージの発見と組織化、そしてドキュメント化を補助する為に作られている。こうした作業は、どのような CIM の導入においても欠かせないものであるが、1 つ目の領域の、組織の知識を構築する作業が重要でないということを意味するものではない。大規模な知識ベースシステムを補強するための、知的な CIM システムもしくは CIM 導入を目指すのであれば、この作業（オントロジー（ontology、存在論）定義として度々触れられる）はより重要なものとなる。しかし、この知識の構造化の補助には、別に専用の IDEF5 が用意されている。この IDEF5 とそのアプリケーションについては、後ほど述べる。ここ

では、組織の情報運用管理要求を捉える為に作られた、IDEF1 の役割に集中して説明してゆく。

IDEF1 の基本コンセプト

まずは現実世界領域に注目してみる（図 A-5)。「現実世界対象物（real-world object)」という用語は、現実世界の人、場所、物、アイデア等の記述に用いられる。この意味では、会社の販売部署も、そこで働いている従業員も、現実世界対象物となる。そしてこうした対象物は、名前や年齢、性別といった、その特徴を表す特性を持っている。更に、ある現実世界対象物は、他の現実世界対象物との間に、何らかの関係を持っている。例えば、従業員は、何かしらの部署に所属して働いている。

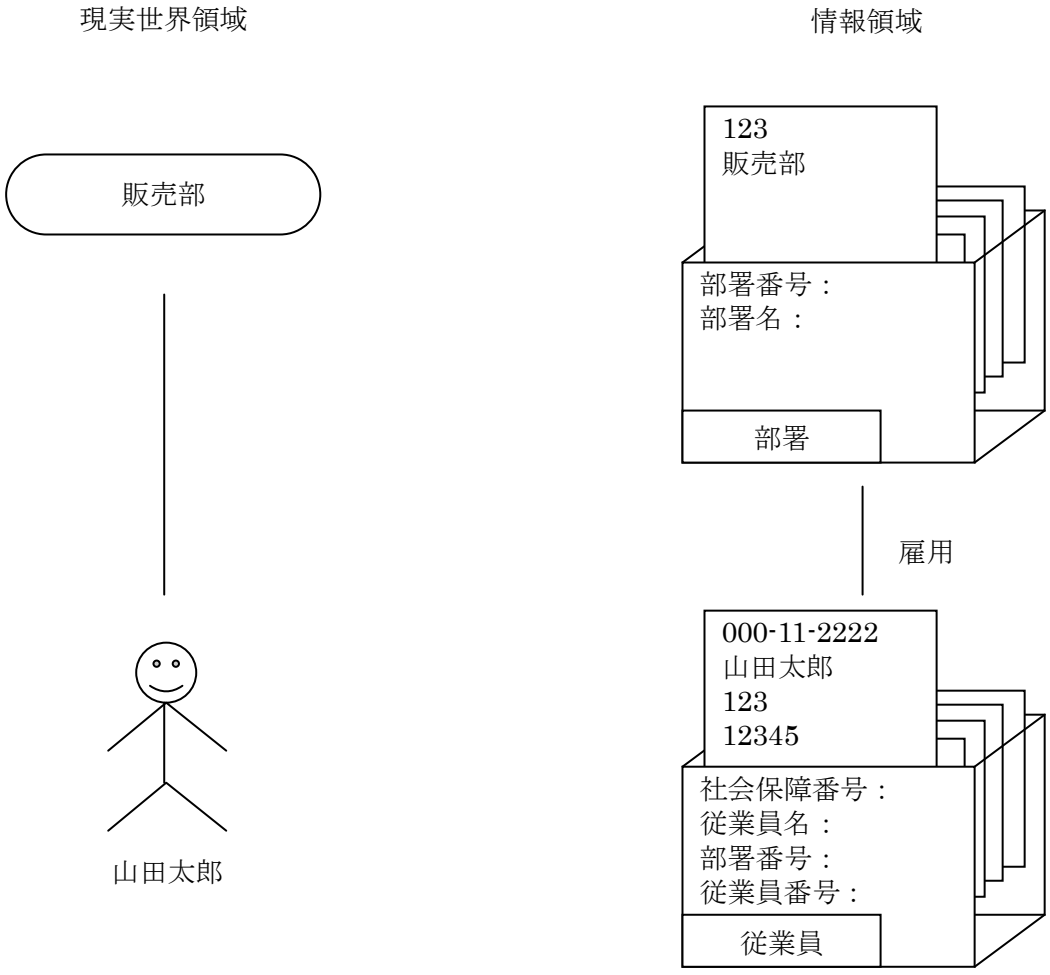


図 A-5　IDEF1 の基本コンセプト

次に情報領域について見て行く。ある IDEF1「実体（entity)」は、ある特定の組織における、物理的もしくはコンセプト的対象物（人、場所、物、アイデア等）についての情報を表現している。例えば、組織が販売部についての情報を所持し、その結果、組織の情報システム内に販売部の情報イメージが存在している場合に、IDEF1 の実体（entity）が存在することになる。IDEF1 における実体クラス（entity class）という用語は、実体の集合か、もしくは現実世界の対象物に割り当てられた情報のクラスを指している。実体クラスは、7×12 ㎝サイズのカード（個々のカードが実体となる）用の空箱のようなものとして考える事ができる。箱の外側には、（１）実体クラスの名前（箱の中のカードがどういう種類のものかを記述する)、（２）個々のカードのテンプレート（これを基にしてカードを作成し、箱に収納する事になる）の 2 つが書かれている。

実体（entity）は、現実世界の対象物の特性の値を記録する、実体の特徴を表す属性（attributes）を持っている。カードファイルの例で言うならば、属性クラスとは、個々のカードの上に書かれている一対の属性値（attribute-value pairs、訳注：身長＝165 ㎝等）の為のテンプレートである。属性クラスという用語は、ファイル箱の外側に書かれている属性の名前の集合体として構成された属性値対のセットを示しており、個々の実体クラスのメンバー（つまり実体）毎の属性値は、個々のカードに記載されている。個々のカードを区別するか、もしくは、実体クラスのあるメンバーを別のメンバーと区別する 1 つ以上の属性クラスの集合は、キークラス（Key Class）と呼ばれている。キークラスは、テンプレートの左上隅に配置し、下線を引いておく。

訳注：

オブジェクト指向言語で、実体クラス・属性クラスは散々出ているのだが、今一つわかり難い。この資料でも従業員カードという実例を図として出しているのに、実体クラス・属性クラスという説明の際に、具体的な対象にまで掘り込んで説明してくれていない。

図 A-5 の例で言うならば、恐らく：

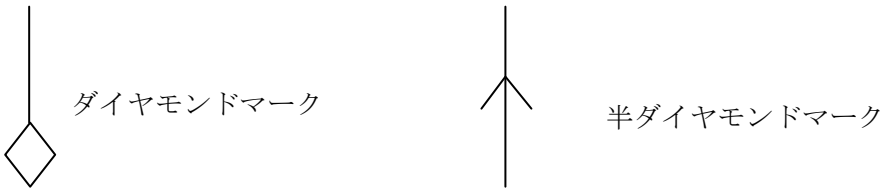
実体（entity）：個々の部署カード（例えば販売部)、従業員カード（山田太郎）
実体クラス（entity class）：部署カード用の箱、従業員カード用の箱
（箱毎にカード用のテンプレートを持つ）
属性（attribute）：部署カードの部署番号と部署名、従業員カードの社会保障番号、
従業員名、部署番号、従業員番号
一対の属性値（attribute-value pair）：

部署カードにおける部署番号＝123、部署名＝販売部、
従業員カードにおける社会保障番号＝000-11-2222、従業員名＝山田太郎、
部署番号＝123、従業員番号＝12345

属性クラス（attribute class）：部署カードなら部署番号と部署名のセット、
従業員カードなら社会保障番号、従業員名、部署番号、従業員番号のセット

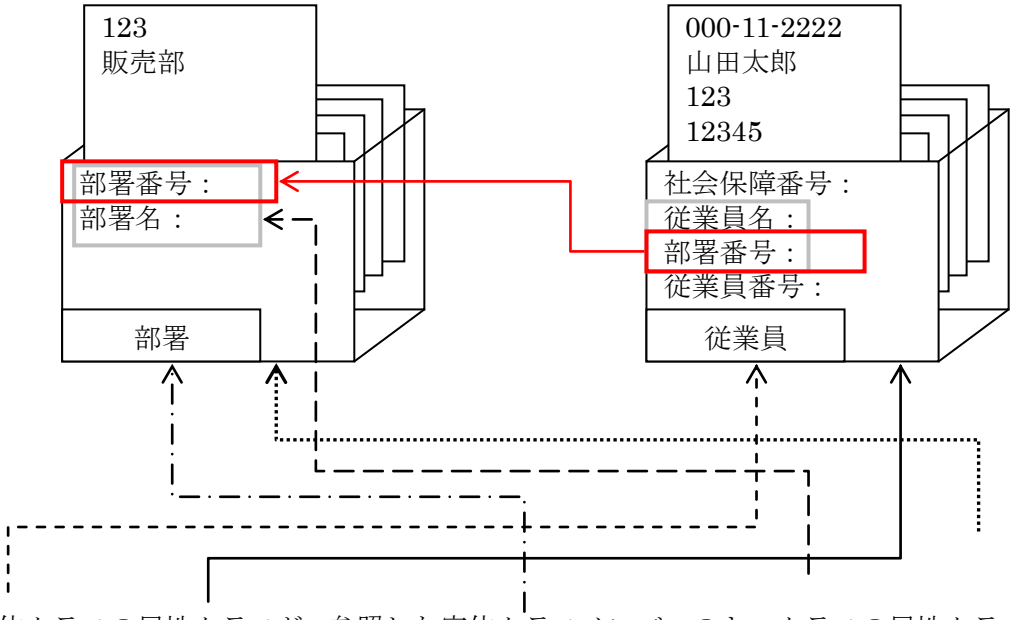
キークラス（Key Class）：部署カードは部署番号と部署名、従業員カードは従業員番号と
従業員名のセット。

IDEF1 における「関係（relation）」は、2 つの独立した情報イメージの間の関連性（association）である。ある実体クラスの属性クラスが、参照した実体クラスメンバーのキークラスの属性クラスを含んでいる事に注意を払う事で、そうした参照の存在を発見（もしくは確認）することができる（次ページの訳注図を参照）。例えば、ある従業員についての情報には、部署番号（ある部署に割り当てられた情報の集合体の属性クラスの１つ）が含まれている。「関係クラス（relation class）」は、実体クラスの間に存在する関連性（association）のテンプレートであると考える事が可能である。この IDEF1 での関係クラスの例として、図 A-5 の、情報実体クラスの「従業員」と、情報実体クラスの「部署」の間に張られたリンクの上にある「（そこで）働く」（work for）というラベルが挙げられる。重要な事は、現実世界の 2 つ以上の対象物の間の関連性について、何の情報も割り当てられていないのであれば、IDEF1 の観点では関係は存在しないということになる。関係クラスは、IDEF1 ダイヤグラム上にある実体クラス箱の間に張られたリンクによって表現される。リンクの端のダイヤモンドマークとリンクの中間の半ダイヤモンドマークは、関係クラスに関する追加情報を符号化したものである（ダイヤモンドマークは基数（cardinality、関係 1 組内の要素数）、半ダイヤモンドマークは依存関係（dependency））。こうしたリンクは、組織の経営ルールの存在を示している事が多い。その為、これらのリンクの分析から矛盾が見つかれば、経営ルール内の矛盾を発見できることが多い。



訳注：　リンク上のダイヤモンドマーク

IDEF1 手法の手続き部分（procedure portion）は、小さな部署レベルでの分析から、大規模な経営プロジェクトに至るまで、規模の変更が可能なように作られている。ある問題の原因が、情報の運用管理不足、もしくは運用管理ミスである場合には、この IDEF1 手法が効率的な問題解決手法であることを証明してくれる。しかし、分析を正しく行うことが重要である。この後、IDEF1 を使用した情報分析における、品質指標の幾つかについて説明して行く。



ある実体クラスの属性クラスが、参照した実体クラスメンバーのキークラスの属性クラスを含んでいる事に注意を払う事で、そうした参照の存在を発見（もしくは確認）することができる

訳注図　上記のややこしい文章の説明

IDEF1 モデリングガイダンス

IDEF1 モデルを素早く監査し、モデル作成者が環境を理解し、かつ既存と将来の情報運用管理要求を正しく認識する上で、必要となる表現を取ることが出来ているかどうかを判断する為の、幾つかの簡単な方法が存在する。また、これらの監査技術により、IDEF1 モデル作成者がモデリングしている物が、現実世界領域の物（一般的に誤りであると考えられている）か、情報領域の物か（一般的に正しいと考えられている）ということを、顧客の側でたちまちに識別する事が容易になる。

1 つ目の方法は、実体クラスの名前に使われているラベルのチェックを行う事である。もし、ラベルが単数形の名前よりも複数形に近いものであれば、情報領域であることを念頭に置いてモデルが作られていない可能性がある。例えば、「従業員（単数形）」というラベルの付いた箱の方が、「従業員達（複数形）」というラベルの付いた箱よりも、「実体クラスは、現実世界の人間を運用管理する為に重要な情報セットを表している」という協定（convention）に、より適合しているといえる。「従業員（複数形）」というラベルを用いているのであれば、モデル作成者は、同じ特性（その会社で何らかの形で働いている人達、等）を共有した人の集団ということを意味しようとしている。忘れてはならないのは、IDEF1 モデル作成者は、現実世界の対象物に直接関係するわけではなく、実際には組織によって運用管理されている対象物に関する情報を、単に取り扱っているだけなのである。

訳注：

今一つわからない。「実体クラスとは対象物そのものでなく、あくまでも対象物を運用管理する為に必要となる情報セットである」という認識がしっかりとしてれば、どれだけ従業員が多人数であったとしても、単数形でクラスの定義を行うものである、ということが言いたいのだろうか。

2 つ目として、キークラス以外に属性クラスが無く、キークラスも無駄に長い（laundry lists）属性クラスのリストを持つ箱がモデルに含まれている場合、それはモデルが現実世界の対象物を表現しており、情報イメージを表現していない兆候（indication）と見ることが出来る。こうした、キークラス以外の属性クラスを持たないモデルは、一般的に組織単位（organizational unit）のような対象物である。長いリストを持つ箱をしっかりと監査することで、その箱が発注書（Purchase Order）のような、フォーム上の実フィールド（actual field）を属性に持つフォームを表現している事が判ることが多い。

訳注：

「キークラス以外に属性クラスが無い」という事は、属性クラスが全てキークラスとなっていると言う事だが、そういう実体は、見るからに上手くモデル化が為されていない

ことがわかる。ただ、そうしたモデル化に失敗した実体の例として組織単位と発注書という例を挙げてくれているのだが、これが良く判らない（このような、例になっていない例が多い…）。IDEF1 についても詳細に読解しないと、この辺は良くわからなさそうである。

3 つ目のチェックとして、モデルに含まれる殆どの実体箱（entity box）に 1 本ないし 2 本の関係しかない一方で、それ以外の少数の実体箱に多くの関係が結ばれているモデルを探す事である。このような場合、高い確率で、全てと関係を持っている箱が、「人」のようなラベルを持った実体箱になっている。この例で言うならば、その実体箱に結び付けられている多くの関係は、人が装う事が可能な役割を示している。例えば、このモデルは次のように読むことが出来る：人は部品を監査する、人は監査を証明する、人は人と結婚する、等。

訳注：

組織の中で働く実体は「人」であることが暗黙の前提であり、わざわざ定義するまでも無い。もっと具体的な実体の定義から始めなければ、本来は不要な部分のモデル化、検討に時間を取られ、モデルが複雑に扱いにくいものになってしまう。

IDEF1 を初めて使う人は、簡単に目に入る現実世界について知っている事をモデリングしてしまう傾向にあるが、こうした IDEF1 の適用ミスにより、利益が殆ど無い、巨大なモデルが出来てしまう事になる。指針が殆ど無いと、初心者モデル作成者は、現実世界で目に入る全てをモデリングしようとする。こうしたモデルは成長し続ける為、運用管理はどんどんと困難になって行き、そしてモデリングそのものの目的が失われてしまう事になる。情報領域ではなく現実世界領域をモデル化してしまった情報モデルは、何の情報が必要とされているのかとか、より効率的な情報運用管理ポリシーにより、どの部分で競争力が向上するのかといった事について、ほとんど何の見識も、もたらしてくれない。

IDEF1 を使用する際のガイドライン

情報システム開発者は、ある情報実体を固有の物として他と区別する事と、現実世界の対象物を固有の物として他と区別する事とを、無意識に混同してしまいがちである。忘れてはならないのは、情報モデルは、現実世界の対象物に関して実際に運用管理されている情報を表現したものだ、と言う事である。それ以外の知識は、情報システムにとって利用不可能なものなのである。現実世界の個々の対象物に対して情報が割り当てられているのであれば、特定の現実世界の対象物を識別する為の手段として、情報モデルを使用する事が可能である。例えば、あるエンジンを他のエンジンと区別する為に使用しているシリアル番号は、それと同時に「エンジン」という実体とそれ以外の実体（タイヤとか発電機？）とを区別している。しかし、現実世界の対象物の種類のみに情報が割り当てられた場合には、同じ状況にはならない。例えば標準ボルトの場合、一般的に個々のボルトに対して固有の部品番号が付与されることはなく、情報モデルでは、ボルトの特定の種類（family）を固有なものとして区別する為に部品番号を使用している。

混乱を避ける為に、IDEF1 モデル作成者は関係クラスの命名と、実体クラス箱間に接続されたリンクの命名に、注意を払わなければならない。図 A-5 を見てみると、このモデルは、非常に直感的に「部署（単数）は、1 人以上の従業員を雇用している」と読める。これは「部署群（複数）に関する情報は、従業員達（複数）に関する情報を雇用している」と同じ事だろうか？。明らかに 1 回目の読解で混乱してしまい、ファイル箱を、対象物に関する情報ではなく、現実世界の対象物であると誤解させてしまっている。(IDEF1 ダイヤグラムの）箱の間に張られたリンクは、1 枚のファイルカードから別のファイルカードへの参照、もしくは情報の関係性を表したものであって、現実世界での関係性を表したものでは無い事に、注意しなければならない。

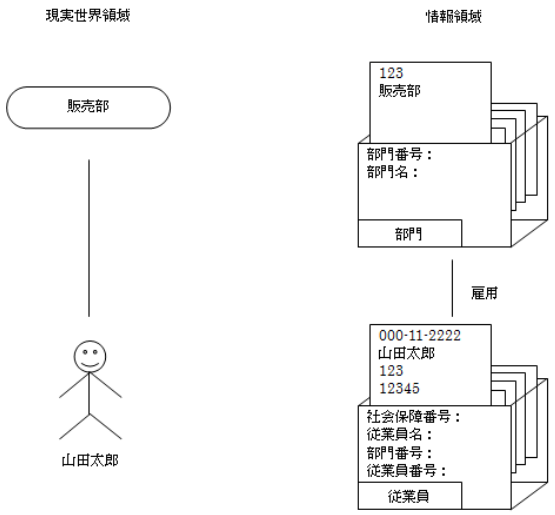


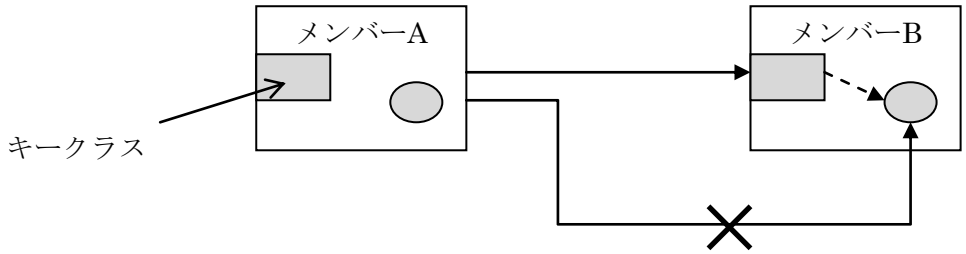
図 A-5（再掲）

「部署は 1 人以上の従業員を雇用する」というような文章は、自然言語（natural language：普通に使う日本語等のことで、曖昧な表現やルール無視もある程度許容されている。それに対してコンピューターのプログラミング言語等は人工言語（artificial language）であり、曖昧な表現やルール無視は認められない）で語られた事実（natural language fact)、もしくはビジネスルール（business rule：これに従って会社組織が動いているものの、不明瞭で不一致も存在する慣習ベースのルール集）である。ビジネスルールは、その内容を知り、運用管理する為に重要なものであることから、ソースファクトリスト（？source fact list、対象に関する重要な事実に絞って簡潔にまとめたリスト）として、別に運用管理されているものを除き、IDEF1 のソースデータ項目の追跡・運用管理方法と同様なやり方で、追跡、運用管理すべきである。その際、実体クラスの間のリンクに、L1、L2 といったラベルを付ける事ができる。このラベル付けにより、関係クラスというものが、ある実体クラスのメンバーの１つを入力すると、その実体クラスの情報関係性に関連したメンバーを返す関数（function）を表現していることが、非常に明確になる。
（訳注：よくわからない。理解するには IDEF1 を詳細に調査する必要がある）

情報関係性（information relationships）と現実世界の関連性（associations）を無分別に混在させてしまうと、混乱や矛盾、エラーを引き起こしてしまうことになる[ISO 87]。例えば、「システム」という実体クラスを作り、この「システム」実体クラスから始まり、「システム」実体に終わる関係を持たせ、この関係に「構成されている（is comprised of)」という関係クラスラベルを付けようと提案されたとき、これをどうやってモデル化するのだろうか？（訳注：この提案内容は矛盾した構成らしいのだが、今一つピンと来ない）

直感的に、システム／サブシステムという階層を持った実体箱（entity box）としてモデル化することが妥当なように見える。「自動車」のように、最上層のシステムは「自動車」という個体でありながら、その中では個々の構成物のレベルに至るまで、多層のサブシステムによって構成されたものになるだろう。このアプローチを採った時、乗り物に関する情報についても、同様なシステム／サブシステム階層構造とすることで、より直感的な思考が可能になる。このアプローチでは、システムの最上層に割り当てられている情報は、システム階層の下方にあるサブシステムに対して割り当てられた情報によって構成されることになる。しかし、自動車に割り当てられた情報の中には、所有者や、認可された州（state)、何年製、といった情報も含まれており、こうした情報については階層構造をとることができない。また、こうした情報は、サブシステムであるブレーキシステムや、ブレーキシステム内に含まれるフロントブレーキ／リアブレーキの種類、ブレーキフルード内の水分量、ブレーキパッドの厚さといった情報とは、何の関係も持っていない。このように、情報の関係性が、現実世界の関係性と同じ挙動を示すとは限らない。

IDEF1 の観点からの唯一の正しい解釈は、ある対象物（object）に関する情報のクラスの１つのメンバーによる、他の対象物に関する情報の参照は、参照されたメンバーのキークラスを通して可能である、というものである。
（訳注：下図のような事を言っているのだと思われる。この事そのものは、情報の構造を判り易いものにする為に必要であることは判るが、しかし何故ここで触れているのかが良くわからない）



訳注：こういう事？？

IDEF1 のルールと手続きは、CIM の実装を目標とする組織で現在運用管理されている情報の、正確なモデルの構築作業を補助してくれると共に、有るべき（TO-BE）システムに必要な情報定義のメカニズムも提供してくれる。しかし IDEF1 は（具体的な）技術とは独立して作られたものであり、CIM の実装設計が開始されると、リレーショナルデータベースの実装の為の IDEF1X と、オブジェクト指向の実装の為の IDEF4 という、別の 2 つの IDEF 手法が使用されることになる。この 2 つの手法については、次の節で説明する。

図 A-2 では、IDEF1X は設計システム（Design System）活動を仕上げる為の手法として作られたものである。IDEF1X は設計手法であることから、IDEF1 の代替として提案されることが多いものの、「現状の（AS-IS)」分析手法には明らかに向いていない。IDEF1X は、（１）既に要求されている情報が判明している、（２）リレーショナルデータベースを使用した導入が決定されている、という 2 つの条件を満たしている時に、論理的データベースの設計に最も向いている。その為、IDEF1X の情報システム的な観点として、リレーショナルデータベース上の実際のデータ要素に焦点が置かれている。もしも、目標となるシステムがリレーショナルデータベースではなく、オブジェクト指向システム等の場合には、IDEF1X は最適な手段にはならない。より詳細を知りたい読者は、論文[GE 85, KBSI 90c, Mayer 89b]を参照のこと。

IDEF1X の開発は、Chen の実体関係性モデル（Entity Relationship (ER) Model）と Codd の関係モデル（Relational model）、そして Smith の集約／一般化モデル（Aggregation/Generalization model）の影響を受けている。これらを基にして、データベースデザイングループ有限会社（Database Design Group, Inc）が論理的データベース設計技術（LDDT, Logical Database Design Technique）を開発した。LDDT は後に、Dan Appleton 会社（DACOM）によって、データモデリング技術（DMT, Data Modeling Technique）という市販品としてまとめられる。そして General Electric 社によって主導された SDRC、CDC、そして DACOM の会社提携により、こうした手法と、それぞれの会社の IDEF1 での経験を基にして、IDEF データモデリング手法、もしくは IDEF1X が開発されたのである（図 A-6)。

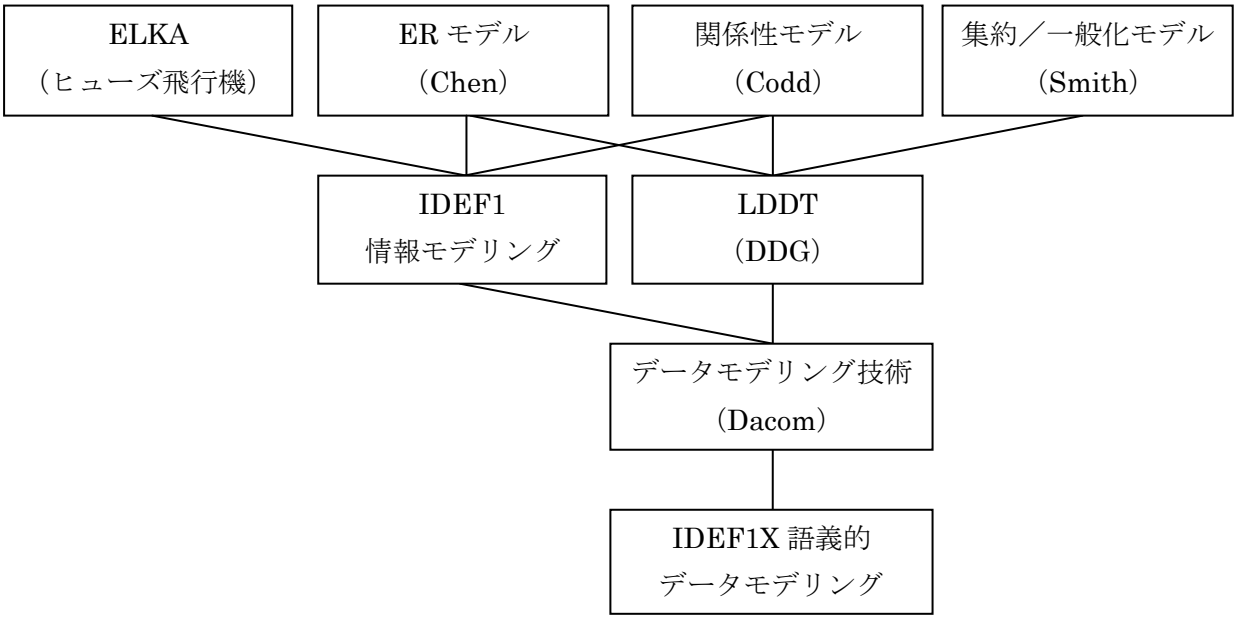


図 A-6　IDEF1X の源流

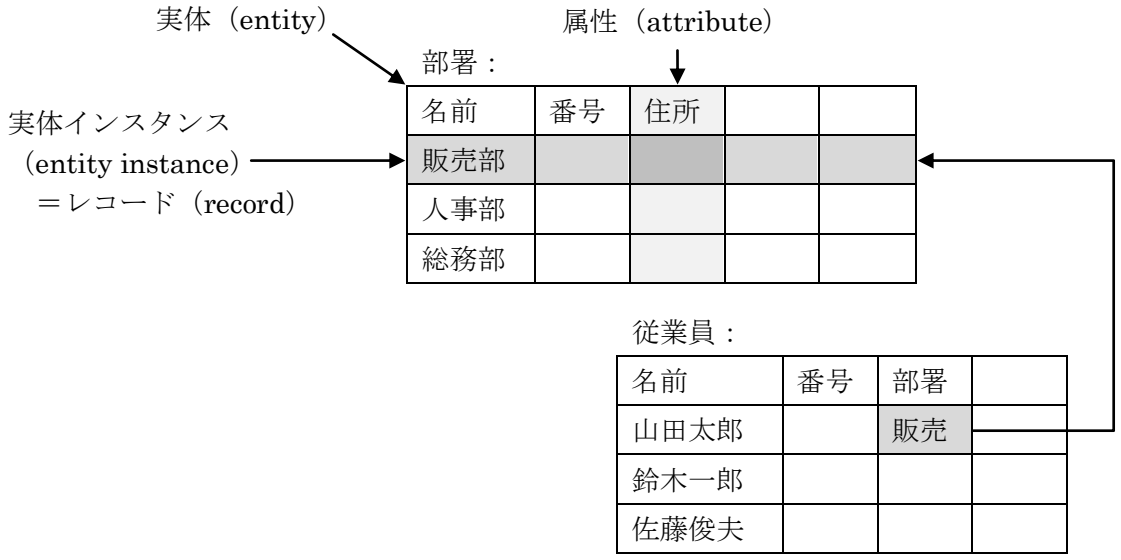
IDEF1X の観点からのモデリングシステム

情報要求が完全に理解されたら、その情報をどのようにして、より効率的に運用管理するのかという決断が可能となる。実行可能な決断の１つとして自動化システム（コンピューター化、プログラム化）の適用があるが、これには適切な設計手法を選択しなければならない。適切な設計手法が選択されることにより、耐久性が高く、高品質で、しかも効率的なコストでの導入が、ライフサイクル全体に渡っても割安なコストで実現される。従って、ある特定の技術に関連する適用を最も成功させるには、それに最適な設計手法を選択する必要がある。このことは通常、設計手法が一般的に「上手く（well)」作動するのは、その設計手法をベースにした経験をカプセル化（encapsulate、要約？）した技術を用いた時のみである、ということを意味している。一連の IDEF 手法は、この事実を認識させ、そして特定の導入技術を念頭にした設計の為の、明確な手法を提供してくれるのである。もしも選択した技術がリレーショナルデータベース技術であれば、IDEF1X は論理的なデータベース設計には最適である。もしもオブジェクト指向データベース方法論が選択されるのであれば、IDEF1X よりも IDEF4 の方が適している（IDEF4 については、後ほど説明する)。

IDEF1X が、非リレーショナルシステムの適用に向いていない理由が幾つかある。IDEF1X では、モデル作成者は、個々の実体を区別する為にキークラスを定義する必要があるが、オブジェクト指向システムでは、個々のオブジェクトの定義にキーは必要とされない。1 つ以上の属性もしくは属性のセットが、独立した IDEF1X 実体として平等に扱わなければならない IDEF1X の状況下では、モデル作成者は属性の 1 つを主キー（primary key）に、そして他は全て代替キー（alternate key）にしなければならない。また、ある実体によって所有されている属性が、別の実体ではキー属性の役割を果たしているものについては、外部キー（foreign key）として明確にラベル付けする必要がある。IDEF1X を用いた作業による結果物（論理的設計モデル）は、情報システム、もしくは論理的データベースの設計の為のたたき台（brueprint）としてプログラマーによって使用されることを意図しており、そしてリレーショナルデータベースの設計に適用される。IDEF1X のモデリング言語は IDEF1 のものと近いことから、情報要求仕様から作成された設計は、対象となるシステムの最終的なユーザーによる批評と理解を、容易にしてくれる。

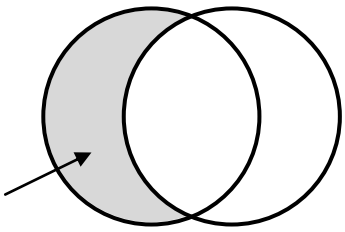
IDEF1X の基本コンセプト

IDEF1X 手法で使用されている用語は、IDEF1 手法で使用されているものと非常に良く似ていることから、混乱を避ける為、更に詳細な定義を行う必要がある。この 2 つの手法には、理論的出発点（theoretical foundations）と概念において根本的な違いが存在している。IDEF1X における「実体（entity)」は、個々に区別可能な類似のデータインスタンス（instances）（人や場所、物やイベント等のデータレコード（data record)）の集合体もしくはセットを示している。つまり、IDEF1X の実体箱は、現実世界領域にあるデータ項目のセットを表現している（？)。また IDEF1X の「属性（attribute)」は、セットの個々のメンバーに属したスロット値（slot value、？）である（このような個々のメンバーは、「実体インスタンス（entity instance)」と呼ばれている（インスタンス：具体的なオブジェクト))。　図 A-7 の「山田太郎」と「販売部」という名前のレコード（record：リレーショナルデータベースにおけるデータ単位、テーブルの 1 行）は、どちらも実体インスタンスである。「部署」は部署を表現しているリレーショナルテーブル上の特定のレコードの集合体（collection）であり、「従業員」は個々の部署によって雇用されている人についてのレコードの集合体である。部署名、部署番号、部署住所は、「部署」実体の属性になる。こうしたセット内の独立したメンバー間に存在する関係性には、名前が与えられている。この場合、この関係性は「参照整合性制約（referential integrity constraint）（参照先が無理やり削除されることで関係性が宙に浮かないようにするストッパーのようなものらしい)」の確立として解釈されている。

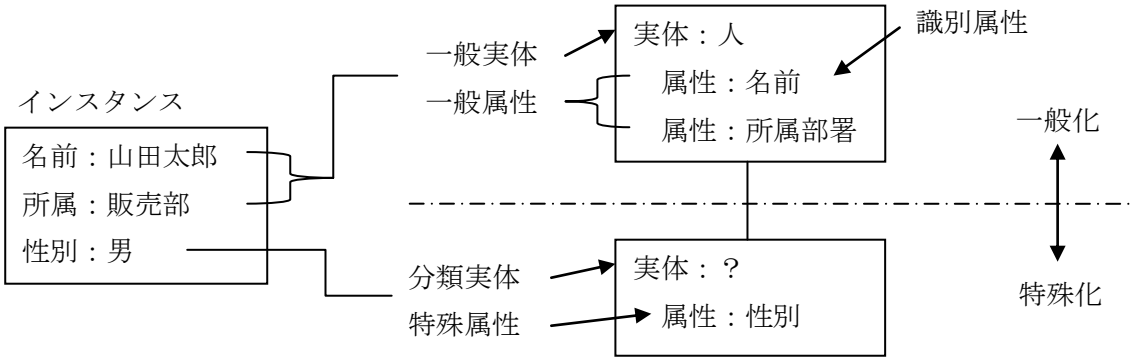


訳注：IDEF1X における実体、属性、インスタンス？

IDEF1X 手法の強力な特色として、分類構造（classification structure）を用いた、論理的データ型（logical data type）のモデリング支援がある。この分類構造は、「一般化／特殊化構成（generalization/specialization construct)」である。箱（もしくは実体）がデータの「型（type)」をモデル化しようとするのに対して、この構造は、データが表現する物の、自然な「性質（kind)」のモデルを積み重ね（overlay）てゆく。この「分類関係性（categorization relationship)」は、ある一般実体（generic entity）もしくは一般実体セットの、相互排他的部分集合（mutually exclusive subsets）を表している。言い換えるなら、共通のインスタンスを持つことが不可能な（複数の）上位集合（superset）から、それぞれ発生した部分集合である。この事の例を挙げるなら、一般実体である「人」から、人間の分類として完全なセットであるところの「男性」と「女性」という 2 つの部分集合を作ることが可能であるが、この時、「男性」セットのいかなるインスタンスも、「女性」セットのインスタンスには成り得ず、逆もまた然りである。男性セットの、あるインスタンスの独自の識別属性（unique identifier attribute）は、当然に一般実体のインスタンスの識別属性と同じデータ型であり、女性インスタンスの場合も同様である。「人」実体の全てのメンバーに適用する一般属性は、一般実体の中に記載されている。特殊属性（この例では「性別」）は、分類実体（category entity）の中に記載されている。



訳注：相互排他的部分集合（mutually exclusive subsets）



訳注：一般実体と分類実体はこんな感じ？

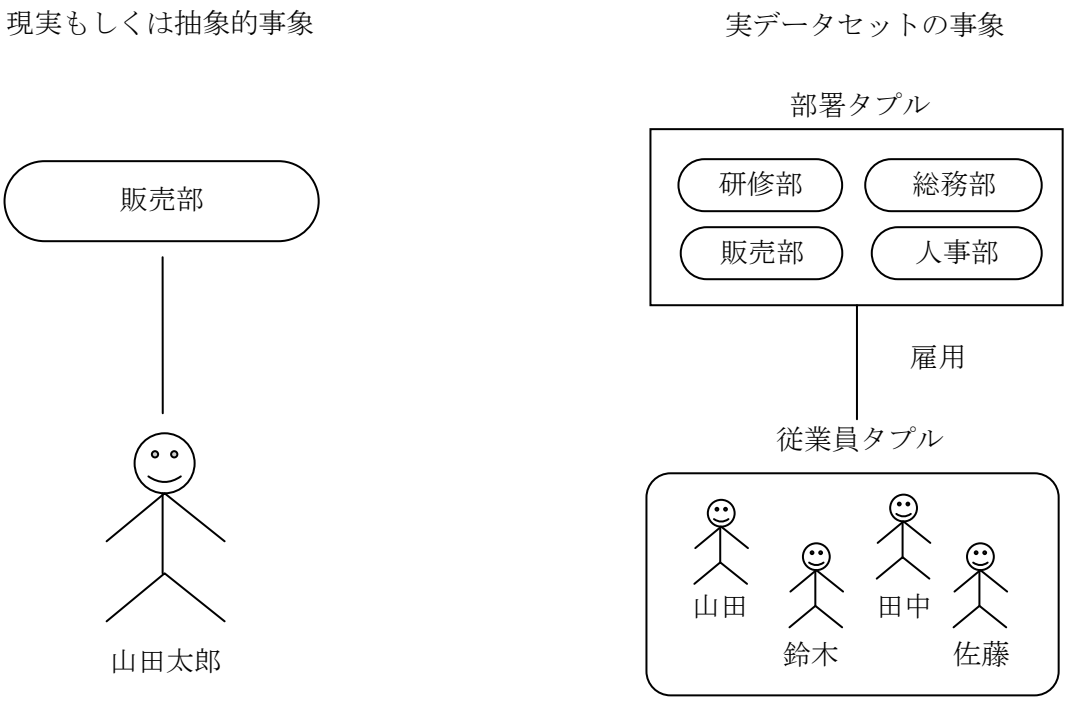


図 A-7　IDEF1X のコンセプト

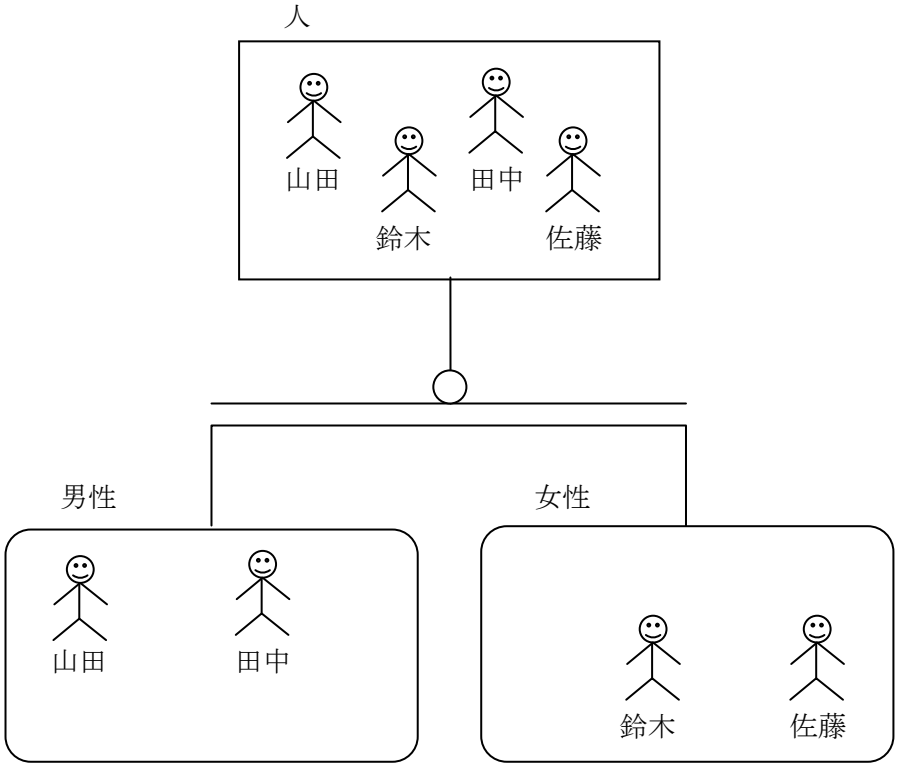


図 A-8　IDEF1X の一般化／特殊化構成

IDEF1X を使用する際のガイドライン

IDEF1X 手法の根底を流れる概念は、人、場所、イベント等の現実世界の物事について、自然言語で語られた事実（natural language fact）のモデリングと、論理的データ構造のモデリングとの、橋渡しをすることである。このことは、現実世界の物事の情報イメージ（物事それ自体でも、物事についての情報を表現しているデータ構造でもない）に、厳密に焦点を当てている IDEF1 の目的とは、非常に異なっている。IDEF1X は、データ項目のこうしたセットに関係するデータ仕様の設計も試みることで、少しづつではあるものの度々拡張されており、またそれと共に、属性と関係性に関する用語集の厚みが増している。

IDEF1 の説明の際に使用した例で考えるなら、IDEF1X は「部署は 1 人以上の従業員を雇用している」とか、「システムは 1 つ以上のサブシステムで構成されている」といった事実（fact）のデータベースによる表現を、モデル化する為のものである。現実世界の物体間に存在する関係性は、それらの情報間の関係性とは同じ挙動を示さないことから、このアプローチが論理的データ構造の設計に便利かどうかについては、疑問が残ることになる。この疑問については、システムの階層構造を表現したモデルの例で考えた方が、読者にとって判り易く、また IDEF1X の誤用の可能性が非常に高い事を理解も容易になるだろう。IDEF1X を誤用してしまうと、リレーショナルデータベースの実体セットではなく、現実世界の物体モデルが出来てしまうことになる。

この問題についての別の実例として、論理的データベース設計用の仕様書を作成しようとしていた空軍に対して納品された、モデルの中に現れた「技術発注改良 (Technical Order Improvement)」と呼ばれている実体がある。この実体名は、実体名の後ろに「フォーム（Form)」という言葉を付け加えた方が、より正確な意味を表すことになるだろう。何故なら、この実体の持つ属性が、AFTO 22 という空軍のフォーム上にある欄（field）に、そのまま一致してしまっているからである。この実体は、属性リストという製品（artifact）（文字通り、フォームそのもの）をモデル化しただけのものである。

訳注：フォーム（form）

書類の書式。米軍では、様々な用途に対して必要となる項目を全て織り込んだ定型書式が準備されている。この例で挙げられている AFTO 22 というフォームも、TO（Technical Order：技術発注書）の修正・変更に必要な項目が準備されており、それを基に書類の修正が行われる（恐らく）。

TECHNICAL MANUAL (TM) CHANGE RECOMMENDATION AND REPLY	**DATE SUBMITTED:**	**DATE RECEIVED:**	*OMB NO. 0704-0188*

Public reporting burden for this collection is estimated to average of 5 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information, Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302 and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

PART I – ROUTING ***(Use complete 3-4 line* address, *including 9 digit zip code and E-Mail address where applicable)***

1. FROM: *(Prod. Improve. Manager or equivalent)*	**2. THRU:** *(Parent MAJCOM CCP)*	**3. THRU:** *(Lead Command CCP)*	**4. TO:** *(Tech Manual Management Office)*
	SEE USING COMMAND T.O. SUPPLEMENT	SEE USING COMMAND T.O. SUPPLEMENT	SEE T.O. CATALOG
(Name/DSN) ☐ APPROVED ☐ DISAPPROVED	*(Name/DSN)* ☐ APPROVED ☐ DISAPPROVED	*(Name/DSN)* ☐ APPROVED ☐ DISAPPROVED	

PART II – CONTROL INFORMATION

5. LOCAL CONTROL NUMBER: *(IAW TO 00-5-1)*	**6. PRIORITY**	**7. TYPE OF CHANGE:**
PIM Will Provide	☐ EMERGENCY ☒ URGENT ☐ ROUTINE	☐ CORRECTION ☒ IMPROVEMENT
8. INITIATOR: *(Name, Rank, DSN, E-Mail)* YOU	**9. INITIATOR'S SUPERVISOR:** *(Name, Rank, DSN, E-Mail)*	

PART III – PUBLICATION (TM) INFORMATION

10. PUBLICATION No:	**11. BASIC DATE:**	**12. CHANGE No:**	**13. CHANGE DATE:**
T.O. Number	ON T.O.		ON T.O.
14. WORK PACKAGE / WORK CARD ID:	**15. PAGE No:**	**16. PARAGRAPH No:**	**17. FIGURE / TABLE No:**
IF APPLICABLE	First Page if multiple references	First para if multiple references	if applicable

18. SHORT DESCRIPTION OF DEFICIENCY:
EXAMPLES:

(1) Need a non-ODS solvent to replace MIL-C-81302 for oxygen system cleaning.
(2) Need a replacement to the CID XX-6453-8433 paint thinner

PART IV – DEFICIENCY:

19.
SEE PART VII – CONTINUATION BLOCK (DEFICIENCY)

PART V – RECOMMENDED TM CHANGE:

20.
SEE PART VII – CONTINUATION BLOCK (RECOMMENDED TM CHANGE)

21. SAVINGS / YR – DOLLARS:	**22. SAVINGS / YR – MAN-HOURS:**
45000	

AFTO FORM 22, 20031117 (MSWord – V4.01) **PREVIOUS EDITION IS OBSOLETE**

訳注図：AFTO 22 のテンプレート

まず最初に問わなければならないのが「これが、AFTO 22 と呼ばれているフォーム一式の、合理的なモデルであるかどうか？」ということである。この実体は、確かに存在するフォーム上の欄を正確にモデル化したものであることから、これについては全く正しいと言える。次に問われなければならないのが「どのような設計活動であっても、この方法でフォームをモデリングすることにより、作業が促進されるかどうか？」というものである。この質問は、「こうした種類のモデリングが、情報要求のセットをリレーショナルテーブルの論理的設計へと変換する作業を補助し、その結果、トレードオフ判断が可能となったか？」と言換える事が出来る。既存のフォームと全く同じ意味しか持たない、リレーショナルデータベースのテーブル、もしくは関係性を作成することに、果たして意味はあるのだろうか？。この（既存のフォームをそのままテーブルにしてしまうという）アプローチを採っても、製品（artifact）の限られた部分しかモデル化する必要のない小規模な導入であれば、影響は殆ど無い。しかし大規模な導入においては、余りに多くの異種フォームができてしまう為、大量の異種フォームとそれに関係するフォーム欄の棚卸作業を担当するモデル作成者を、大量に雇う必要が出てくるのである。更に重要な事として、そうしたモデルが設計仕様として導入者の手に渡ると、導入は現在の紙ベースのシステムを単にコンピューター化するだけで終わってしまうということである。このような、紙ベースシステムの単なる自動化に終わってしまった例は、CIM の失敗の歴史の中でも数多く存在している。

訳注：
詳しくは書かれていないが、どうもこの AFTO22 と呼ばれるフォームを IDEF1X でモデリングした際に、フォーム欄の数だけ属性を持つ、単純なテーブルを 1 つだけ作るという間違いをしてしまったようである。その分野では有名な話なのか、説明不足気味である（上記の AFTO22 のフォームも、ネットで検索をかけてようやく拾ってきた）。

このような例から、設計手法を誤用すると、設計の混乱や仕様からの逸脱が発生してしまう事は明らかである。これを回避する方法の一つとして、「実体（entity）は、現実世界の対象物（object）そのものではなく、対象物に関する情報を表現しなければならない」という協定（convention）を確立する事が考えられる。しかし、これは一般化／特殊化構造という文字通りの大きな例外を除き、IDEF1 の文法的相違以上の物にはならなかった。そして協定の確立という方法も、十分に注意して使用しなければ混乱を招くことになる。例えば、「人」に関する情報セットを、男性の情報と女性の情報の、2 つの種類へと分割することは正しいと言えるのか、ということである。恐らくこれは間違いであり、もし分割してしまえば、それは恐らく誤ったものになってしまう（訳注：この男女の例の意味が良く

わからない。現実世界の対象物と異なっているからと言って良いわけではない、ということ？)。更に重要な事として、IDEF1X から IDEF1 へと戻してしまうと、良い設計手法が１つ失われてしまうことになる（？)。

恐らく IDEF1X は対象物そのものだけのモデリング手法としては最適ではあるが、対象物の関連データの構築には向いていない。このような種類のモデルは「概念モデル (concept model)」と呼ばれることが多い。繰り返すが、現実世界では通常、種類（kinds）は重なり合い、相互に分離したものでは無い事から、一般化／特殊化構造において困難が発生してしまう。IDEF1X はメンバー間で相互に参照している分類実体（categorization entity）を許容していない事から、IDEX1X は概念モデリング言語としては上手く機能しない。例えば、会社内における従業員（employee）の種類モデル（a model of the type）には、管理者、エンジニア、設計者、司書等が含まれている。この場合での一般化実体 (generalization entity）は「従業員（Employee)」であり、特殊化実体（specialization entity）は管理者、エンジニア、設計者、司書となる。この構造（structure）では、エンジニアは設計者にも管理者にもなれないことが暗示されている（訳注：しかし実際には可能)。

手法（methods）には 3 つの異なった特性（aspect）が必要である。1 つ目は、現実世界と、人や場所、イベント等の間に存在する関係性について、何が知られているのかということを効率的に把握可能であることである。2 つ目は、既存の準備された情報運用管理要求（information management requirement）を獲得可能であることである。3 つ目は、情報運用管理要求に合った特定の技術を適用した、システム設計の支援に適した手法だということである。今後説明してゆく IDEF3 プロセス記述獲得と IDEF5 存在論記述（Ontology Description) 手法は、1 つ目の要求に特化したものであり、IDEF0 機能モデリングと IDEF1 情報モデリングは、2 つ目の要求に対応している。そして IDEF1X データモデリングと IDEF2 システム動的モデリング、そして IDEF4 オブジェクト指向設計は、3 つ目の要求を満足するように作られている。残念なことに、一般的に使用されているモデリング言語の多くは、こうした 3 つの領域を明確に区別できていない。

IDEF2 によるシステムの動的モデリング

生産並びに情報システムの分析、計画、設計に、シミュレーションモデルを使用する利点は、この 20 年で確立したものになりつつある。その結果、複雑な相互関係や統合的生産システムの動的挙動の研究を行うシミュレーションに、より多くの経営者が興味を持つようになっている。シミュレーションモデリングは、幾つものアプリケーション領域に跨る複雑な問題の解決において、強力な支援ツールとなってくれる。シミュレーションは、以下のような場合に便利である：

（1）システムの構成物間の、時間に依存した複雑な相互関係から生じた問題の原因究明を行う場合

（2）既存システムの変更により生じる影響を分析する必要がある場合

（3）提案されたシステムが、これまでに存在しない（新しい）ものである場合

（4）システム効率の改良もしくは計測の選択肢を、数値化する場合

（5）他の数値化分析手法が、コンピューターで取扱い困難な場合

シミュレーションにより仮定を問い、そして既存の知識から新しい情報を引き出すことが可能となる。代替となる設計と行動の評価を伴うシミュレーションを行う事で、システム運用（system operation）と管理ポリシー（management policy）を、より深く理解する事ができる。

シミュレーションを、効率的な生産もしくはシステムの開発判断の補助ツールとして幅広く使用するには、シミュレーションモデリング技術の設計と使用において、膨大な訓練と技能とが必要であることが、シミュレーション導入の障害となっている。シミュレーションの不満点（frustration）は、実際にそのシステムを運用し、システム運用の詳細を記述可能な該当分野の専門家（expert）が、シミュレーションモデリング作業を主導できない事である。運用の専門家は、シミュレーションモデルの設計と開発を、設計と開発の経験を持ったシミュレーション分析者に依存しなければならない。その為、該当分野の専門家とシミュレーション分析者の間で、情報交換が効率的に行われるようにしなければならない。シミュレーション作業が成功するかどうかは：

（1）実務運用の専門家がシステムの事をシミュレーション分析者に対して、
　　どれだけ十分に説明（記述）できるか
（2）シミュレーション分析者が、該当するシステムの事をどれだけ理解可能か
（3）システムに関する説明（記述）と、シミュレーションモデルで果たすべき目標の
　　正確な伝達を、どれだけ効率的に行えるか

の 3 つにかかっている。IDEF2 では、モデルの仮定と設計を、その分野の専門家へと伝達する能力を向上させることにより、シミュレーションモデル作成者の生産性を改善することに、焦点を置いている。

　このように IDEF2 はシミュレーションモデルの設計ツールであり、時間によって変化してゆく製造システムのリソース（resource ※）をモデルとして表現する為の言語と共に、シミュレーションを基にした数式モデルの仕様作成の為の枠組み（framework）も提供してくれる。IDEF2 は、システムが特定の状況下でどうなるかの予測を実行可能にしてくれる、典型的なシミュレーションモデルの設計プロセスを改善する為に作られたものである。この手法について、より詳細に知りたい読者は、論文[Softech 81c]を参照のこと。

訳注：
　resource は「資源」と単純に翻訳されてしまいがちであるが、「資源」という言葉から連想される材料・素材といったものの他に、人や設備・施設、システムや環境といった幅広い意味での「資源」も含まれている。

IDEF2 の観点からの、シミュレーションモデル設計

IDEF2 手法の開発は、連続型（continuous）、離散型（discrete）、ネットワーク型の各シミュレーション言語の設計作業と、設計されたシミュレーション言語の工業界への導入作業とで得られた、膨大な経験を基にして行われた。IDEF2 の主要な設計者は、アメリカ合衆国における工業シミュレーションを主導する Pritsker & Associates 社の A. Alan B. Pritsker と Robin J. Miner、そして John F. Ippolito の 3 人である。

IDEF2 はシミュレーションモデルの設計を、以下の 4 つのサブモデル（submodel）へと分解（decompose）している：

（1）施設サブモデル（システムと環境のエージェントモデルを明確する為に使用）

（2）実体フローサブモデル（実体の変化モデルを明確にする為に使用）

（3）リソース配分サブモデル（resource disposition submodel、
実体変化要求に対するエージェントの論理を明確にするために使用）

（4）システム制御サブモデル（モデル化されたシステムとは独立したイベント、
もしくはシステム外のイベントによる変化の効果を明確にする為に使用）

このサブモデル分解の目標の 1 つは、分析チームが、大規模なモデルの構築作業を、容易に分割（partition）可能にしたいという需要を満たすことである。また、シミュレーションの施設サブモデルを、他の数値化されたプラント配置分析の敲き台として使用したり、リソース配置サブモデルとシステム制御サブモデルを、工場床面制御仕様（specification）の為の制御論理仕様として使用するというような、サブモデルの仕様を実際のシステム詳細で再利用可能にしたい、という意図も目標に含まれている。そして最終的には、行動分割（behavioral partitioning）による仕様分解を使用することで、モデル仕様の再利用が達成されることが望まれていた。

IDEF2 の基本コンセプト

IDEF2 はコンピューターを利用可能な図形的仕様言語（graphical specification language）である。これは、図形的文法により、シミュレーションモデル設計を明記するという意味である。そして文法は完全である為、明記されたシミュレーションモデルを直接実行することが可能である。この論文では概略を説明するだけで、文法の個別の要素については説明は行わないが、サブモデルは下図の例のように記述されている。

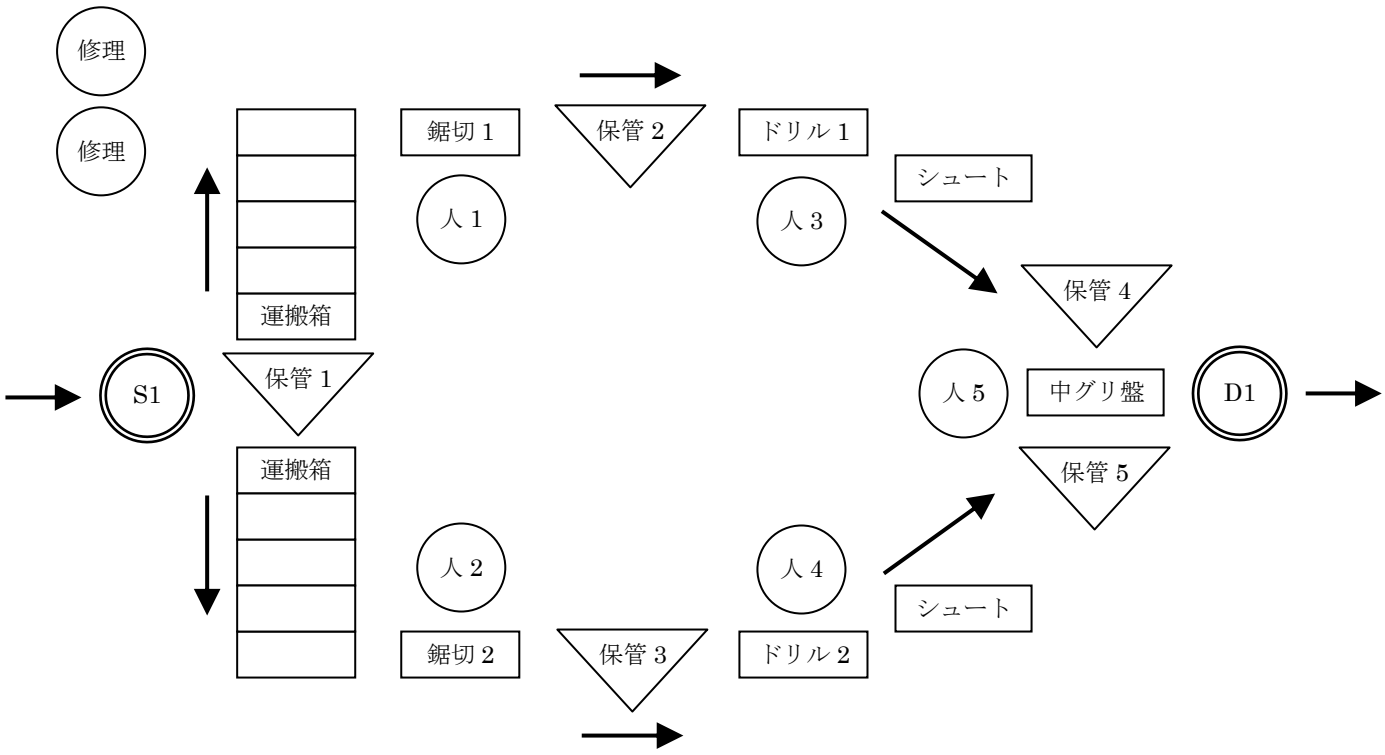


図 A-9　工場配置に使用された施設サブモデル

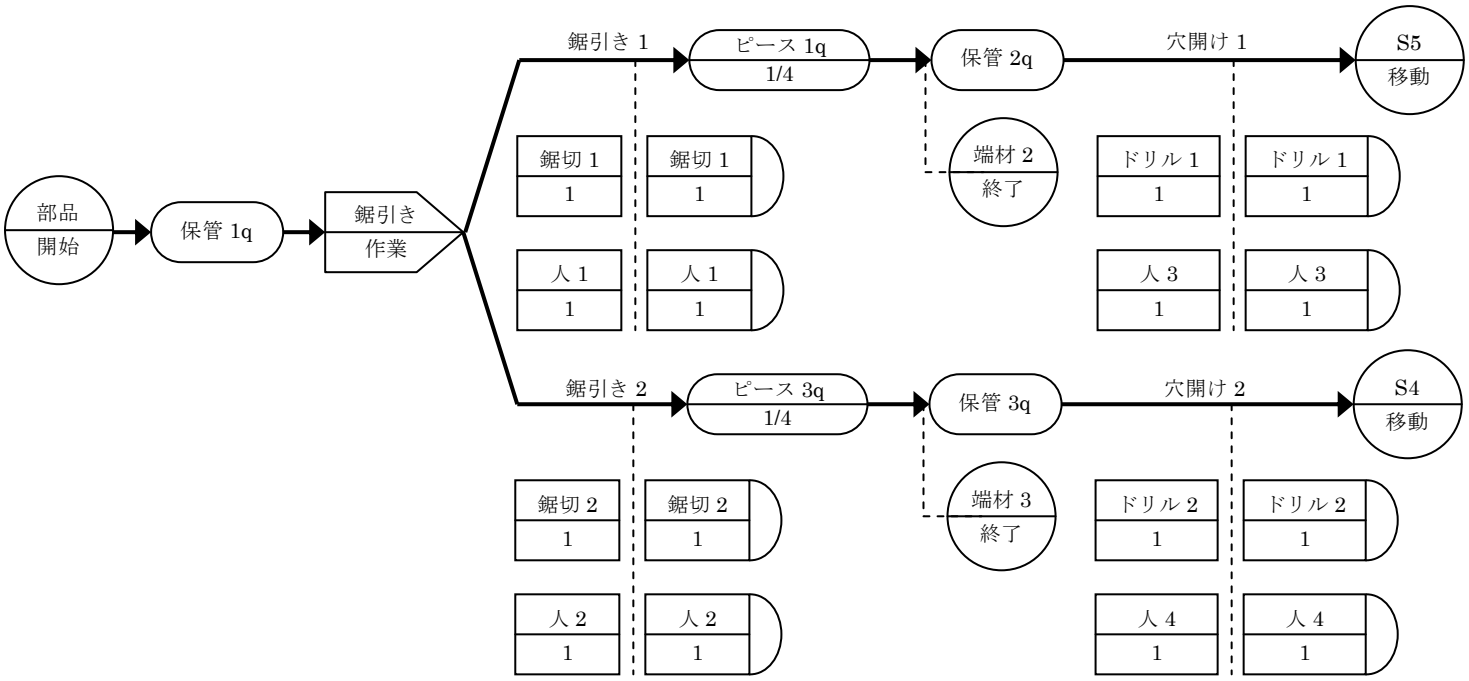


図 A-10　実体フローネットワークの例

IDEF2 モデリングの問題点（論点 issue）

IDEF2 は CIM 導入の開始時と、工場近代化作業において、広く用いられている。Pritsker & Associates 社により開発された VAX（DEC 社の 1970 年代の画期的なコンピューターシステム）ベースのシミュレーション・決断支援システムは、こうしたモデルを分析する為に使用されていた。しかし現時点では、IDEF2 手法は相対的に休止状態にある。仕様化能力（specification capabilities）とグラフィックが画期的に進化し、MAP や SLAM、SIMAN 等のシミュレーション言語が市販化されるようになった。それにより、IDEF2 の持っていた、シミュレーションモデルの設計とシミュレーションプログラムの実行との間に存在した語義的隔たり（semantic gap）を埋める能力が、広く知られるようになった。この能力は、シミュレーションモデル作成者にとっては生産性を向上させる重大な進歩であったが、シミュレーションの訓練を受けていない経営者（decision maker）にとっては、利点は少なかった。この事は、その昔、CAD システムが登場した事により、生産設計者が機械部品の仕様を簡単に作成できるようになったものの、そうした仕様作成作業の背後にある実際の設計決断作業に対しては、役に立たなかった事と似ている。

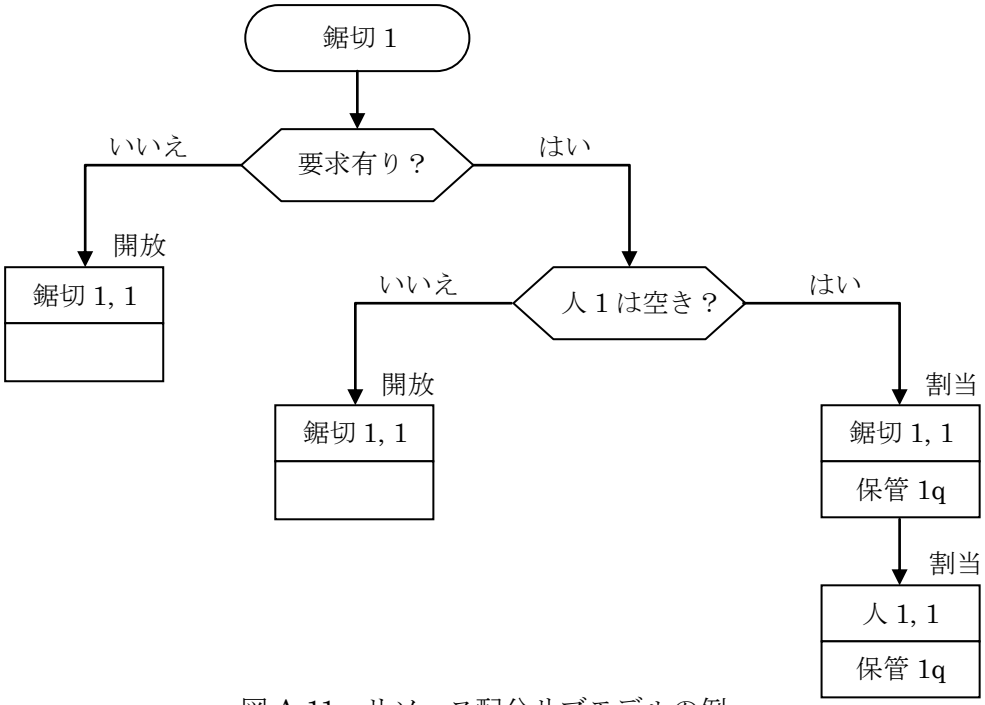


図 A-11　リソース配分サブモデルの例

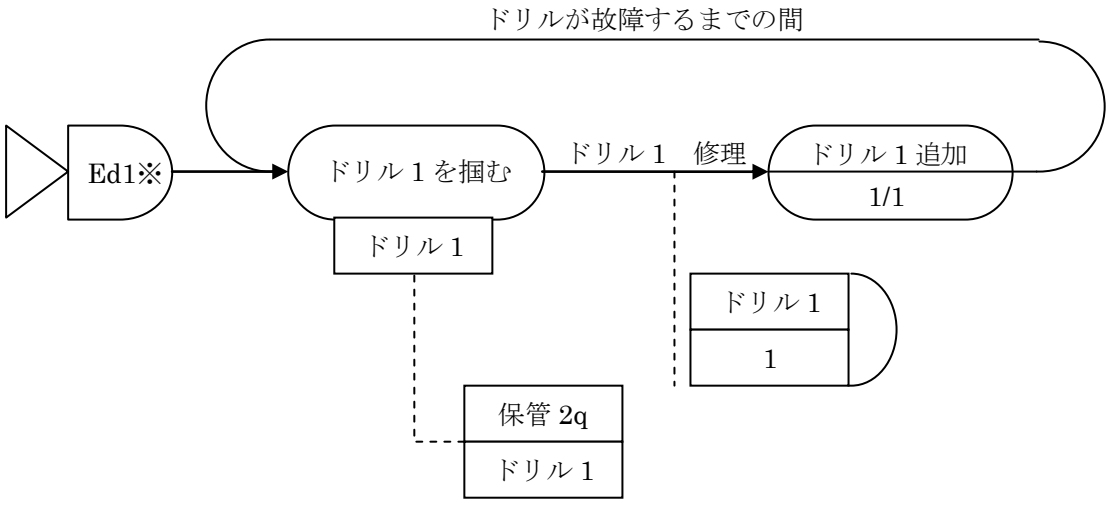


図 A-12　システム制御サブモデルの例（※→意味不明）

次の章で説明するIDEF3手法は、その分野の専門家が、自分たちの環境下でのプロセスフローと対象物の状況変化に関する知識の記録を、補助する事を目的としている。現在行われている研究では、知識ベースシステムを構築することにより、このようなIDEF3プロセス記述と分析での設問への回答を基にした、シミュレーションモデルの自動設計の実現を目指している。このシステムでは、経営者（decision maker）がシステムの何を知っているのかについて入力補助をしてくれる。(彼の受け持つシステムがどのような挙動をしているのかという記述と、彼が回答を望むシステムについての質問)。こうした環境下（IDEF3によるプロセスフローと対象物状況の記述）では、IDEF2は、シミュレーション分析者による批評の為に作成されたモデル設計の、中間表現を提供している。今後、最近出始めたオブジェクトベース(オブジェクト指向で無い場合)での仕様構造の実装に合わせたIDEF2の変更が、期待されている。

次世代の IDEF　（訳注：IDEF2 までの時点で）

手法（method）とは、意図して作られたシステム（designed system）である。他の CIM 技術とは異なり、手法は人の頭の中（on the human mind）で実行されるものである（複数の領域に跨る複数の人の頭脳の集合体により実行されることの方が多いが)。そして、他のシステムと同様に、作られた際に予め設けられていた限界を越えてしまうと、手法は機能しなくなる。

IDEF 開発の目標は、システムを統合している情報の定義（もしくはリバースエンジニアリング)、設計、開発、そして運用管理を行う手法を相互に連結したフレームワークを提供する事である。この「フレームワーク」[Zachman 87, IUG 90, Mayer 90]という用語は、システムの開発と進化を支援する為の、手法、ルール、手続、ツールの集合体を構造化したもの、という意味で使われている。

IDEF 手法を適用した最初の 7 年間の経験から、以下に挙げるものを含む、幾つかの追加手法が必要であることが認識された[Mayer 87]。

（1）論理的なイベント順序、もしくは時間的なイベント順序による、シナリオの獲得
（2）効率的なオブジェクト指向アプリケーションとデータベースの設計
（3）現実世界領域に記述された対象物の、参照記述の把握
（4）CIM システムそのものの TQM（Total Quality Management、総合品質管理）を行う為に必要となる、CIM 設計決定の理論的根拠（rationale）の記録

これらについては、次節から説明する次世代の IDEF 手法により、順次対処されている。

IDEF3 による、プロセスフローと対象物状態の記述

（Process Flow & Object State Description With IDEF3）

状況もしくはプロセスを記述する為の最も一般的な情報伝達機構の一つは、イベントもしくは活動を順番に並べる事で表現された物語（story）である。IDEF3 は、このような叙述的な活動の為に特別に作られた、シナリオによって表現されるプロセスフローモデリング手法である。IDEF3 は、その分野の専門家（expert）にとって自然な形式で記述された、状況とイベントとの間に存在する順序関係と因果関係を、直接的に把握すると言うコンセプトを基にして作られている。IDEF3 の目標は、対象となる分野の専門家が持つ、あるシステムもしくは組織的作業についての知識を表現する為の、構造化された手法を提供する事である。IDEF3 の記述は、以下に挙げるものを含めた、数多くの目的の為のデータ提供に使用する事が可能である。

（1）システム分析プロジェクトにおける事実探索インタビュー（fact-finding interview）から得られた生のデータを、記録、分析する為の、システム的手法の提供

（2）経営における主要な運用シナリオでの、組織の情報リソースの影響力（impact）の決定

（3）重要な共有データ（特に製造、エンジニアリング、維持管理そして生産を定義しているデータ）の状態とその時系列での変遷（states and life-cycle）に影響を及ぼす決定手続きを、ドキュメント化する為の機構（mechanism）の提供

（4）データ構成管理（data configuration management）の定義と、管制ポリシー定義（control policy definition）の変更

（5）システム設計と設計トレードオフ分析の支援

（6）シミュレーションモデルの作成を支援する強力な機構の提供

（7）機能モデル（IDEF0 モデル）作成に便利な情報の提供

（8）事実と決定点（decision point）、作業分類（job classification）の定義を明確にするための機構を提供する事による、リアルタイム管制（realtime control）を実現するソフトウェアの設計での、プロセスマッピングの促進

（９）ユーザー視点からの需要と要求の作成に必要なデータを、明確に定義する手法の、
分析者への提供

（１０）専門的なシステムの開発に必要となる、その分野での専門家の視点の、
収集と表現（express）

IDEF3 プロセス記述獲得手法（Process Description Capture Method）は、既存のシステムもしくは作成予定のシステムの「行動（behaviral）」面に関しての、その分野の専門家の持つ知識を獲得する為に、システム開発者によって使用されている。

IDEF3 によって獲得されたプロセス知識は、シナリオという枠内（with in the context of a scenario）で構造化されており、その意味で IDEF3 はシステム記述での直観的に知識を獲得する機械である。一般化と単純化を促進する為、システムの 1 つの側面のみを適用し、時系列論理（tempral logic）を全て削除する IDEF0 モデルとは異なり、IDEF3 は、経営プロセスに関連し、時間と共に変化する（temporal）順序と因果関係についての異なったユーザーによる記述を、構造化する為に用いられている。その結果、IDEF3 の記述は、分析モデルと設計モデルを構築可能な、構造化された知識ベースとなっている。

システムに対して要求されている挙動についての記述と、システムの挙動を予測する数学的理想化もしくは数学的モデルとを区別する必要性から、IDEF3 が開発された。IDEF2 シミュレーションモデリング手法と、他のシミュレーション言語（QGERT や SLAM 等）の一群は、後者（数学的理想化もしくは数学的モデル）を満足させることを目的としたものである。そうしたシミュレーション言語は、時間とともに変化するシステムリソースの挙動を表現し、シミュレーションの基盤となる数学的モデルの仕様（specification）の枠組（framework）を提供する。一方の IDEF3 は、異なったユーザーによるシステムに対する視点（view）を構成（organization）し、表現する為の言語として、前者（システムで想定されている挙動）を取扱っている。IDEF3 が基盤としている構造化（organizational）の原理とコンセプトは、（１）Shir Nijssen 博士と（２）Jon Barwise 博士が始めた研究が元になっている[Barwise 83, Devlin 89]。Nijssen 博士は、情報システムとは、組織内のエージェント（agent、行為者、執行者）の間の談話状況（discourse situation）を具体化（embodiment）したものである、という観念を発案した。Barwise 博士は、そうした談話状態がどのようにして生じているのか、そしてそうした談話状態により、どのような情報の流れが支援されているのかということを再認識（new understanding）する為の理論的基礎を提供する事になる、状況理論（situation theory）と状況意味論（situation semantics）

という、全く新しい分野を創出した。この手法の理論的背景について詳細を知りたい読者は、論文[Mayer 89a, Mensel 90]を参照するとよい。

IDEF3 の観点からの、システムの記述
（Describing System From The IDEF3 Perspective）

IDEF3 には、プロセスフロー記述と、対象物状態変遷記述（object state transition description）の、2 つのモデリングモードがある。プロセスフロー記述は、組織の中で「物がどのように動いているのか」という知識を獲得するためのものである。また対象物状態変遷記述は、特定のプロセス下で対象物が許容されている変遷を要約したものである。プロセスフロー記述と対象物状態変遷記述は、どちらも記述を構成する情報ユニット（unit）を含んでおり、この 2 つのモデルの実体（entity）により、IDEF3 記述の基本ユニットが構成されている。その結果、他の IDEF 手法のダイヤグラムと説明文（text）が「モデル」を作り出しているのに対して、IDEF3 のダイヤグラムと説明部は「記述（description）」を構成している。この特性の違いは重要である。

前にも説明したように、モデルとは、対象となる現実世界システムの重要な面を模倣する為に作られた、対象物と特性、そして関係性で構成される、完全に理想的なシステムである。実際のところ、モデルはそれ自身、現実世界の挙動の予測に用いるモデルが設計状態を完全に保持し続ける（hold over）という推定の下で、現実世界のシステムを仮定、単純化したものに囲まれて構築されたシステムなのである。その為、モデルの利便性を保障する為に、モデルは自己完結し、内部的に整合性が取れたものでなければならない。

しかし記述（description）は、一般的に不完全なものである。例えば、ある人が、自分の専門範囲外のプロセスやイベントについて部分的に知ることは出来ても、全てを知ることはできない。また、不適切な記述がされたり、単純に書き忘れてしまうことで、記述から事実が省かれてしまうこともある。記述とは、我々の周囲にある世界について正しいものと仮定された事実と信念とを単純に記録した物であり、不完全である。その為、モデルの構築には、その分野の専門家による記述の蓄積が必須となるのである。

IDEF3 の基本的なコンセプト

IDEF3 のプロセスフロー記述は、特定のシナリオの背景（context）内の活動（action）間に存在する、関係性のネットワークを把握してくれる。この記述は、特定の問題を解決しようとしている（もしくは問題が繰り返されている）状況下で（in the context、文脈、背景）、特定の組織内を物がどのように動いているのかを表わそうとしている。IDEF3 は「シナリオ」を、プロセス記述を行う際の焦点と境界状態とを確立する為の、基本的な組織化構造として使用している。手法のこの特性は、人間というものは、与えられたシナリオもしくは状況の下（within the context of）、これまでに経験、観察した活動の順序に従い、知っている事象を記述する傾向がある、という事実を基にしたものである。この、プロセス記述という流れの中で（within the context of）の思考と表現を組織化しようとする自然な傾向は、確立された流れの中の（within the establish context）組織活動を支援する役割を担う、実現可能なシステム設計の、代替となる「外部からの視野」を提案する為の非公式なフレームワークとして、シナリオを幅広く使用することを推奨（motivate）してきた。このような開発アプローチは、「外部制約駆動設計（External Constraint Driven Design）」として言及されており、新システムの設計の為の効果的な機構として、実際に繰り返し使用されている。図 A-13 は、IDEF3 プロセスフローダイヤグラムの例を示している。

（訳注：全般的に良くわからない。それにしてもこの著者、context が好きだな…）

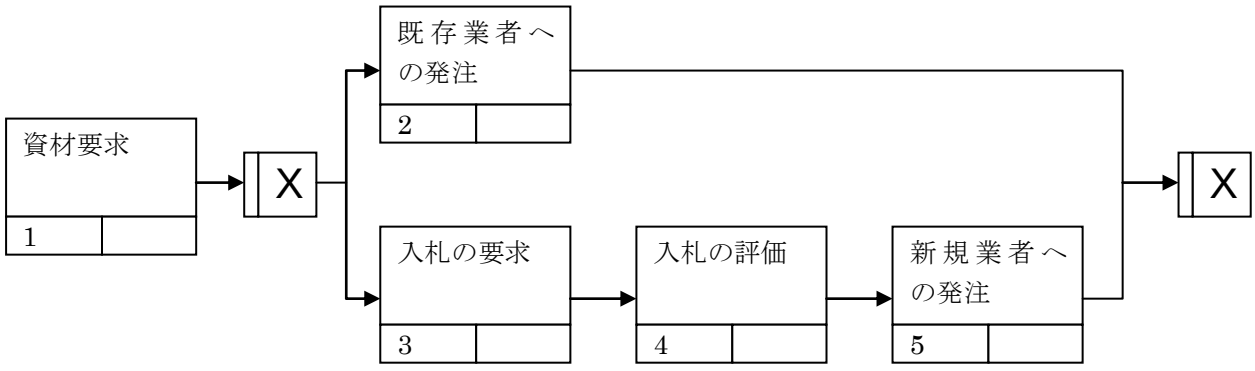


図 A-13　プロセスフローダイヤグラムの例

IDEF3 プロセスフローダイヤグラムは、以下の構造体によって構成されている：

（１）行動ユニット（Unit of Behavior、UOB）
（２）接合点（Junctions）
（３）リンク（Link）
（４）参照先（Referents）
（５）詳細追記（Elaboration）

IDEF3 プロセスフローダイヤグラムの開発は、こうした記述実体（descriptive entity）の作成と操作によって行われる。

対象となるシナリオを背景として（within the context of）使用されている IDEF3 の図形的記述の基本的な構文単位が、行動ユニット（UOB）である。この UOB は周囲の構造に従い、機能（function）、活動（activity）、作用（action）、動作（act）、プロセス、運用（operation）、イベント、シナリオ、決定（decision）、手続（procedure）という種類へと更に分類される。このように分類された個々の UOB は、対象となるシナリオに関係する、知覚された情勢（state of affairs）もしくは変化（state of change）という点での特定の視野（view）を表現している。各 UOB は、分解（decompositions）と呼ばれている「他の UOB からの記述」と、詳細追記（elaborations）と呼ばれている「関係する対象物とその関係性のセットによる記述」の、両方と関係を持つことが可能である。図 A-13（前ページ）から、IDEF0 の活動（activity）を、IDEF3 の UOB として再利用すること（並びに UOB と相互参照させること）が可能であることが判る。

UOB は、接合点とリンクを用いて、互いに結び付けられている。接合点（Junction）は、UOB ネットワーク内に有る同期・非同期行動を表現する意味素性（sematic facility）を提供してくれる。またリンク（Link）には、①時間的順序（temporal precedence）、②対象物フロー（object flow）、そして③関係性（relational）の 3 つの種類がある。この関係性リンクは、時間的順序リンクと対象物フローの標準的な語義では適応できない制約を表現する。

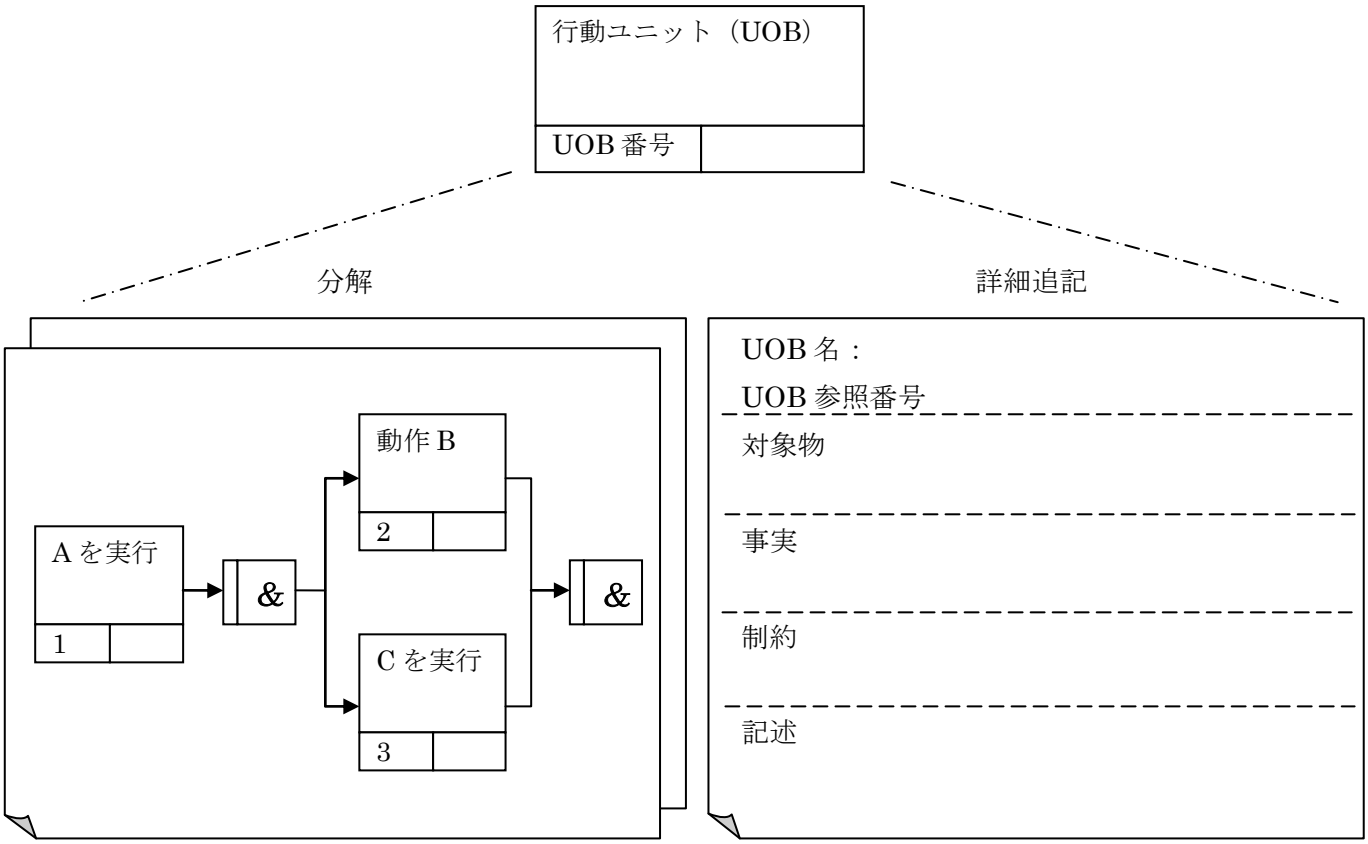
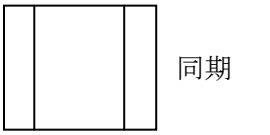


図 A-14　UOB と、その分解・詳細追記

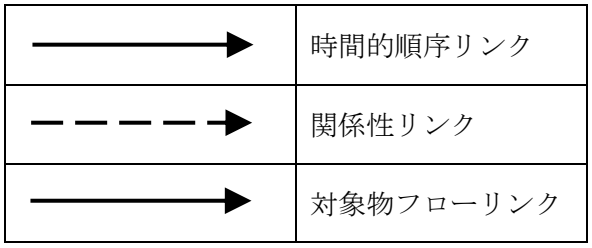
接合点の種類



接合点の種類（分岐・合流）

種類	略号	意味
AND	&	かつ（論理積）
OR	O	もしくは（論理和）
XOR	X	どちらか（排他的論理和）

リンクの種類



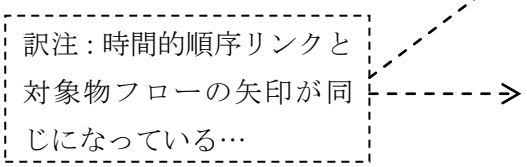


図 A-15　IDEF3 接合点・リンクの種類と意味

対象物状態変遷ダイヤグラム（Object State Transition Diagram）は、プロセスの対象物を中心とした視野（view）を把握する為に使用される。プロセスダイヤグラムから、この視野を切り出すことで、その領域における対象物の変遷の可能性を要約してゆく。対象物状態変遷記述の実体（entity）には、以下の物がある。

（１）対象物状態（Obhect State）

（２）状態変遷円弧（State Transition Arcs、アークは枝の意味もあり）

対象物状態は、特性値（property value）と制約（constraint）によって定義される。特性は、情報システムによって追跡し続けられているが、この特性は IDEF1 モデルの中で属性として定義しておき、対象物状態変遷ダイヤグラムとの間で相互参照させなければならない。更にこの対象物状態は、それに関係した、変遷前（Pre-transition）と変遷後（Post-transition）という 2 つの制約を持つことが可能である。この制約により、①変遷が開始可能となる、もしくは、②変遷が完了可能となる、為に満足さなければならない条件（condition）が明確になる。こうした制約は、特性／値の簡単なリストか、もしくは制約の宣言（statement）によって明確にされる。属性値は、要求が満たされる値になっていなければならない。また、対象物状態ダイヤグラムを描くことで、対象物が、ある状態から次の状態へと変遷する際に必要となるプロセスの、ネットワークが明確になる。

図 A-16 は、対象物状態変遷ネットワーク（OSTN）ダイヤグラム上で、対象物状態がどのように記述されているかを示したものである。実線円は、実際の状態を記述したものである。個々の対象物状態には、それに関係する詳細追記が付属している。対象物状態ダイヤグラム（OSD）のフォームは、OSTN ダイヤグラム上にある全ての対象物状態に対して、別々に作成される。OSD フォームを描くことにより、状態の特性を詳細化するだけでなく、対象物がその状態である為に必要な条件と、その状態から次の状態へと、前の状態からその状態へという発生し得る全ての変遷に必要となる仕様が、判り易くなる。

状態を定義する為に必要となる物には、次の 3 つがある：

①（対象物状態への）進入条件（entry condition）

→　対象物が、前の状態からその状態へと変遷する為に必要なもの

②（対象物状態からの）退出条件（exit condition）

→　対象物が、その状態から次の状態へと変遷する為に必要なもの

③状態記述（state description）

→対象物が、その状態であり続ける為に必要なもの

こうした条件は、属性とその値との組み合わせや、制約によって表現されている。

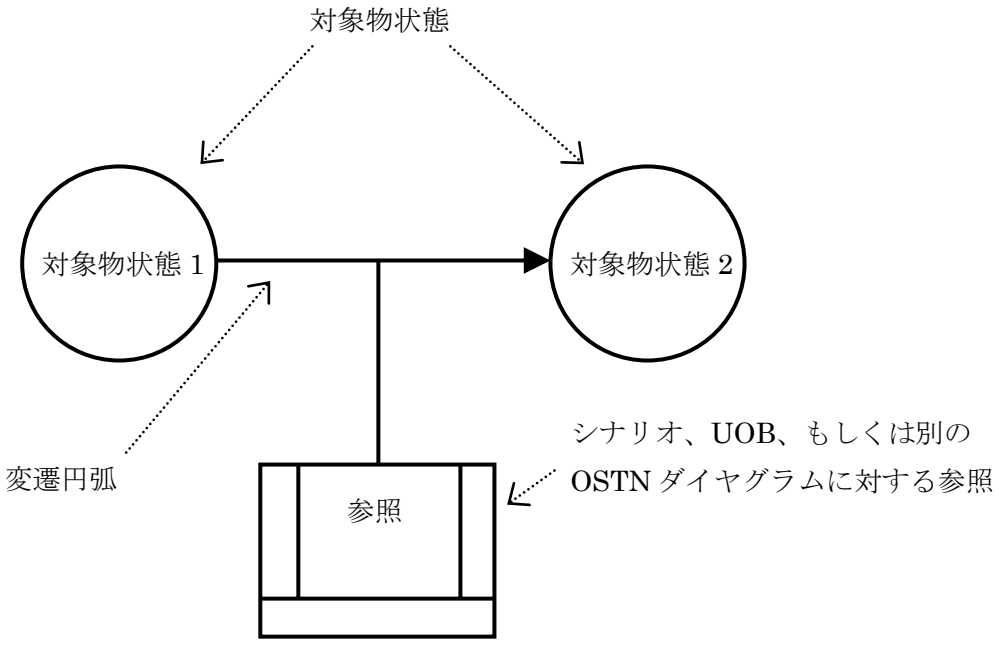


図 A-16　IDEF3　対象物状態記述

IDEF3 を使用する際のガイドライン

IDEF3 は、その分野の専門家の知る彼らのシステムの挙動に関する、生の記述を収集する媒体である。しかし、そうして収集された事実の組織化（organizing）と翻訳（conveying）には、数多くのやり方（mechanism）が存在する。

余りにも複雑になってしまった IDEF3 ダイヤグラムは、多くのシナリオと視点（viewpoint）とを 1 つのダイヤグラムに混在させてしまった可能性がある。

IDEF3 ダイヤグラムを作成するヒューリスティック（heuristic、辞書には「発見的方法」とあるが、「コツ」とか「裏技」という意味の方が自然？）は、もしも表示されている UOB が、あるシナリオに関係する全ての視界（view）から見えていない場合、恐らく、その UOB は分割されるべきである。

また、収集したデータの「空白を埋め」ようとする傾向には注意すべきである。記述獲得手法として、IDEF3 は「不完全さ（partial）」に寛容に作られており、矛盾した記述すら可能である。組織と、その組織を支援する情報システムとを分析する際に、そうした不一致や不完全な範囲が、問題の原因となることが多くある。もしも、それが省略されたものであっても意図的なものであっても、こうした不一致や不完全な範囲が記述から隠されてしまうと、「このモデルは、何も問題は無い（these models don't indicate anything is wrong）」という良く聞かれるコメントが上がるだけで、問題は気づかれないままになるだろう。（訳注：その為、収集したデータの空白を埋め、この不一致や不完全を誤魔化そうとすることは、避けなければならない）。もちろん、モデル作成者は全ての範囲を概略的に（実際には興味のある範囲だが）推敲しているので、問題が気づかれないままになることはない。、

IDEF4 によるオブジェクト指向設計

IDEF4（オブジェクト指向設計手法）は、IDEF1X のように、システムの自動化（コンピューター化）を行う際の設計手法として、使用する事を意図したものである。ただし、IDEF1X がリレーショナル技術であるのに対して、IDEF4 はオブジェクト指向技術の導入での使用を目的としている。

オブジェクト指向原理が登場し、それを用いた開発が行われるようになると、モジュール性や保守性、再利用性といった、ライフサイクル品質として望ましい性質を持つプログラムコードを作成する能力が、オブジェクト指向原理にあることが知られるようになった。それと同様に、オブジェクト指向プログラミング方法論（paradigm）により、ソフトウェアコードの作成そのものも容易なものとなり、より多くの人がコードを作成することが可能になった。しかし皮肉なことに、コード作成が容易となったことで、設計の出来の悪いソフトウェアが作られ易くなってしまった。コード設計の出来が悪いと、モジュール化されておらず、保守が困難であり、そうしたシステムは、再利用が困難となる。IDEF4 の目的は[Edwards 89]、オブジェクト指向プログラミング（Object-oriented Programing、OOP）方法論の正しい適用を支援し、オブジェクト指向技術による長所を安定させる事である。IDEF4 について説明する前に、まずは OOP の背景と原理について見て行くことにする。

製造業の進化に伴って情報コストが上昇して行く中で、注目を集めるようになった技術が OOP である。世界的な製造企業で生産される主要な生産物は、コスト、関係者数、重点、量のいずれの面から見たとしても、情報なのである。そのような状況で、情報の生成コストを削減したいという需要は、ある技術に猛烈な興味を抱くようになったのである。この OOP は、マサチューセッツ工科大学の人工知能研究室において、CAD システムのユーザーインターフェースとして使用され始めたものである。その後はシミュレーショングループによって長い間使用されていたが、人工知能コミュニティの中で、可能性を秘めた知識表現方法論（knowledge representation paradigm）として再び注目されるようになり、遂には主要なソフトウェア生産性方法論（software productivity paradigm）となったのである。この方法論がエンジニアリング分野と製造分野に与える影響と利益は、その方法論を実際にどれだけ理解できたかに依存している。

現在、オブジェクト指向○○を悩ませている問題の一つは、○○の本質（nature）である。ベンダーが、自身が持たない技術の習得を余りに急ぎ過ぎ、それを供給しようと無茶（mad dash）をするため、オブジェクト指向を再定義してしまう傾向にある（「リレーシ

ョナル（relational）」や「スリースキーマ（three schema）」、「ルールベース（rule based）」、といった、それ程古くない決まり文句を暗示させている）。その為、「オブジェクト指向（object-oriented）」だけでなく、「オブジェクトベース（object-based）」や「オブジェクト中心（object-centerd）」という用語すらできてしまった。Ada や IMS といったものですら、「本質的には」オブジェクト指向であると表現されてしまい、疑う事を知らない人（mildest skeptic、最も穏やかな懐疑論者）でも、一体何がどうなっているのかと戸惑うような事態になっている。

訳注：

relational：IDEF1X 参照。この文脈ではオブジェクト指向との対比で用いられている。

three schema：情報システム構築に用いられるアプローチの一種

rule based：知識ベースシステムの問題解決手法の種類の 1 つ

Ada：強力な言語機能を豊富に持ち、高度な型の体系をもつプログラミング言語。
後にオブジェクト指向化され、オブジェクト指向言語の国際標準を取る。（Wiki）

IMS：データマネージメントシステムの 1 つ。

オブジェクト指向が意図していた目標は、恐らく、再利用性（reusability）、修正容易性（modifiability）、信頼性、保守性、冗長なコードの削減、そして、より複雑なシステムの理解・導入が可能な人間能力、といったものを達成することだったのであろう。プログラミング言語としてのオブジェクト指向の重要な特性は、その抽象化のレベルの高さであり、以上に挙げた目標に、大きく寄与している：

再利用性は、次の 2 つの主要な手段で達成されている：

①あるデータ型式（オブジェクトクラス型式）で繰り返し実行される運用のセットを、
そのデータ型式の定義をも構成するソフトウェアモジュールとしてカプセル化する

②プロトコル（データを送受信する際の手順の協定）、もしくは
一般化ルーチンとその意図された使い方の、目的と効果の宣言仕様を通して

修正容易性と保守性は、再利用性と同じ手段によって強化されている。それに加え、継承（inheritance）という考え方を用い、プログラムの代替案（alterations）の作成を継承リンクを加えるだけで簡単にできるようにしている。オブジェクト指向プログラミング言語（もしくは設計手法）で無い場合には、そうした代替案の作成に、広大な範囲でのプログラムコード（もしくは設計）の書き直しが必要となる。

冗長なコードの削減も、継承により実現されている。それにより、実際にコピーをしなくとも、プログラムコードの広い範囲を（1 つのマクロとして定義することで）コピーするのと同じ効果を得る事が可能である。冗長性が削減されることで、信頼性も向上することになる。プログラムコードの冗長なコピーを繰り返すと、データベースで同様な事を行うよりも、多くの不一致を生み出す原因となる。

オブジェクト指向は、問題領域（problem domain）の存在論（ontology）についての人間の思考方法の基礎となっている、分類学的な論法（taxonomic reasoning）と密接にリンクしていることから、概念的な強化ツールとなっている。オブジェクトクラス階層と分類学とは異なる物であると強調されているが、この 2 つはオブジェクト指向にとっては十分に似たものであり、問題の概念化とプログラムコード作成との間に存在するギャップを、大幅に狭めてくれる。

オブジェクト（object）という用語は、元々はコンピューター言語の用語であり、（適切なローカルステート情報（relevant local state information）を意味する）ローカルステートデータ（local state data）と、プログラムされた挙動（behavior）によって構成されているプログラム対象物（object）の作成、取扱いを支援する、プログラミング概念を示したものである。ここで重要な言葉は、「適切なローカルステート情報（relevant local state information）」である。実際にオブジェクト指向プログラムの大きなソースを調べてみると、オブジェクト（object）のクラスは、典型的には、何らかの情報概念を表わしていることがわかる（稀に現実世界の対象物（object）を表していることもあるが、一般的ではない）。その為「オブジェクト（object）」という用語は、共通的な用語（common term、ここでは object のこと）を特別な方法で使用する際に現れがちな事象に悩まされている。つまり、一般的な「オブジェクト指向におけるオブジェクト（object）」は、同じ名前を持つ物理的対象物（object）を参照しているわけではない、ということである。また同様に、オブジェクト指向設計でのオブジェクトクラス階層構造も、同じ名前の付いた物理的対象物に関する分類学的階層構造とは一致しないのである。その為、「オブジェクト指向プログラミング」におけるオブジェクトとは、通常はデータそのものを意味しており、それが実際の対象物と一致するかどうかとか、実際の対象物と同じ挙動をするかどうかといったことは、重要ではないのである。一方で、こうしたオブジェクト指向の持つ概念的強化ツールとしての特質を、システム開発の設計、要求、果ては問題分析の各段階へと展開しようとした、OOP 手法論（paradigm）の姉妹的な手法論が登場しつつある。

下に挙げる定義は、オブジェクト指向技術のレベルを理解する助けとなるだろう。

（１）オブジェクトベース（object-based）技術は、現実世界の物理的もしくは概念的な対象物（object）に対する類推法（analogy）を基にした、方法論的分析（methodological analysis）と概念化支援（conceptualization support）とを提供してくれる。オブジェクトベース技術は、人の頭の中で機能している。

（２）オブジェクト中心（object-centered）技術は、ローカルステート（local state、局所状態）の記録域（storage)、プロトコル、手法のそれぞれの定義から構成される、プログラミングオブジェクトを提供してくれる。ただし、プログラマーは普通、オブジェクト状態（object state）とプロトコル手法についての、自分だけの基本的な継承構造を構築しなければならない。

（３）オブジェクト指向（object-oriented）技術は、オブジェクト状態（object state）の記録域、プロトコル、手法のそれぞれの定義と取扱いを行う為の、完全な言語とマシンサポート（machine support）（型のチェックや、多段階継承の支援も含む）とを提供してくれる。

（４）永続オブジェクト指向（persistent object-oriented）技術は、オブジェクト指向構造の定義とインスタンス（instance）を保存する為の、ディスクレジデント（disk regident、ディスク上の記録単位？）支援を提供し、オブジェクトの有効期間（？half-life、半減期）を延長する。

幸いにも、ハードウェアと OS が進化し続けている事から、プログラマーがメモリーの割当てのようなレベルの詳細さにまで頭を悩ます必要が無くなりつつある。OOP ベースのシステムは、抽象化レベルを、アプリケーションプログラマーが保守性を十分に確保した問題に対する解決案を表現可能となり、それによってシステムプログラマーが詳細な問題に専念することが可能な点にまで、引き上げる事が可能になった。その結果、これまでに述べてきたオブジェクト指向の利点を十分に活かすことが出来るようになる。より多くのプログラマーがオブジェクト指向言語を適用できるようになれば、永続オブジェクトを提供する為に、最近登場したオブジェクト指向データベース（OODB）技術を使用し、他のシステムからネットワーク越しにデータベース上のオブジェクトへとアクセスする事が可能となる。OOP 方法論の参照語義（？reference semantics）がより判り易くなることにより（データが表現しているものと、それがどのように使用されるのかが、より明らかになる、等）OODB のシステム効率は、従来のリレーショナルシステムより 100～500%も改善すると見積もられている。

IDEF4 の観点からの設計システム

構造図（structure charts：システムを管理最低階層に至るまで分解し図示したもの）やデータフローダイヤグラムといった伝統的な手法論と、（階層式、リレーショナル、ネットワークといった）データ設計モデルが、既存のシステム開発にいて開発哲学と開発実践（practice）を支援してきたように、IDEF4 は、オブジェクト指向設計決断プロセス（decision making process）の支援を促進しようと模索している。

IDEF4 は、以下の 2 つの主要な設計目標を特に掲げている：

①導入する事により、所要のライフサイクル品質を実現し、また全体の導入開発時間を短縮する、オブジェクト指向設計の作成支援

②オブジェクト指向コードによって作成される製品が、完成した際に設計を満足させ、かつ所要のライフサイクル品質を実現するかどうかの評価を容易にする

訳注：ライフサイクル品質（life-cycle quality）
完成した直後だけでなく運用予定期間を通じ、保守管理から廃棄に関しても
要求された仕様を満足させるために必要となる品質

構造図、データフローダイヤグラム等の、構造化設計（Structured Design：モジュールによって構成する設計手法）が、機能プログラミングの領域において、より高い品質と生産性を促進しているように、IDEF4 手法は、オブジェクトプログラミングの領域において、同様な長所を発揮する事を目指している。そして構造化設計の関数的側面（？functional emphasis、機能的側面？）が、Pascal や C、Modula-2 や Ada といったプログラミング言語の関数的側面と相対しているように、IDEF4 設計のオブジェクト指向的側面（object-oriented emphasis）によって、Smalltalk や CLOS、C++、Eiffel といった言語のオブジェクト指向的側面を提供しようとしている。その為、IDEF4 は「設計システム（Design System）」の為の機構（mechanism）であろうとしている。

IDEF4 は、これまでに開発されてきた概念と表記法の、多くの長所を保つような方法で、オブジェクト指向設計に関する情報を、維持管理している。このように過去の資産（previous work）との一貫性が保たれることで、ソフトウェアエンジニアによる IDEF4 モデリング技術の効果的な学習と使用とが補助されることになる。しかしながら、IDEF4 はオブジェク

ト指向データベースとプログラミング環境の、最良の設計経験をカプセル化しようとも試みている。それを行う為に、IDEF4 は以下に挙げる物の識別（identification）、取扱い（manipulation）、表示（display）、分析（analusis）に、焦点を置いている：

（１）インスタンス変数、クラス変数、一時変数、そしてこうした変数値のデータ型／オブジェクト型を含む、オブジェクト定義

（２）継承的階層構造もしくは格子構造（lattice structure）と、他のオブジェクトと関係する個々のオブジェクトの可視性（individual objects visibility）を含む、オブジェクト構造

（３）インスタンス化（instantiation：初期化）の手法と制限（初期化前、初期化後、初期化中の各状態）、削除の手法と制限、そして挙動の抽象的記述を含む、個々のオブジェクトの挙動（indivisual object behavior）

（４）以下のものを含む、手法のシステムプロトコル
- ・オブジェクト継承格子内における手法の位置
- ・引数（argument）の型・数・順序
- ・各プロトコル項目の挙動の抽象的記述
- ・個々のオブジェクトクラスの抽象的挙動の特殊性（specialization）

設計方法論として、IDEF4 は、システム開発プロセスの設計段階で、設計チームが運営管理（manage）を行う為に重要となる、オブジェクト指向システム設計の内容物が公表されるように構造化されている。

オブジェクト指向設計で主要となる要素は：
- ・宣言（state：プログラム用語、明言する）の維持管理
  （これはクラス構造で定義される）
- ・挙動の共有（これはメソッド（method：ここでは「手法」ではなくプログラム用語としての意味）と継承構造で記述される）

の２つである。

一方で IDEF4 は導入設計にも使用されている事から、オブジェクトよりも、既存のレコード、ストラクチャー、リスト、ストリング（record、structure、list、string：いずれも

プログラムコードの変数型・データ構造）や、他の共通データ構造の方が、適する場合もあるだろう。共通データ構造を使用する事も考慮して、IDEF4 はクラスとメソッドの特殊化により、型を明確にするという手法を準備している。

プロジェクトの規模拡大によって設計モデルが煩雑で使い難くなることを防ぐために、IDEF4 は、モデルを様々なサブモデル、ダイヤグラム、データシートへと分割した、独特の組織構造を採用している。言換えるなら、IDEF4 はオブジェクト指向設計活動を、別々の運用管理可能な大きさの活動へと分解すると共に、分解されたサブ活動毎に、設計の他の側面に影響を与えることになる設計決定を図形的手法により強調することで、設計を支援する。IDEF4 の設計モデルに含まれている全ての情報を、1 つのダイヤグラムだけで表現することは不可能であるが、これにより複雑性が制限され、必要な情報を素早く調査することが可能となるのである。そして異なった種類のダイヤグラムの間をオーバーラップするように注意深く設計することにより、異なったサブモデル間の一貫性を保証してゆく。

IDEF4 は、コンピューターシステムのオブジェクト指向設計を行う為の、図形指向方法論（graphically oriented methodology）であり、オブジェクト指向設計の決定プロセスを支援するために必要なものを提供してくれる。

IDEF4 設計モデルは、概念的に、クラスサブモデル（Class Submodel）とメソッドサブモデル（Method Submodel）の 2 つのサブモデルから構成されている。この 2 つのサブモデルは切換え割当（Dispatch Mapping：ディスパッチマッピング、クラス毎に実行するメソッドを切り替える仕組み。同じ名前のメソッドでも駆動するクラスの違いにより、異なるメソッドが実行される）を通して接続・合体されており、それにより設計モデル内にある全ての情報の取得が可能である。しかし、クラスサブモデルとメソッドサブモデルの大きさの都合から、設計者はそうした全体的な構造を見る事は不可能であり、その代り設計者は、クラスサブモデルとメソッドサブモデルの内部情報を効率的に取得することが可能な小さなダイヤグラムを集めたものを利用する。

IDEF4 では、以下の 5 つの種類のダイヤグラムが使用されている：
（訳注：IDEF4 単体の資料内にあるダイヤグラムの分類とは異なっている）

（1）継承ダイヤグラム（Inheritance Diagram）
クラス間の継承関係を明確にする

（2）型ダイヤグラム（Type Diagram）
他のクラスのインスタンスを値として持つ、あるクラスの属性を通して
定義されたクラス間の関係性を明確にする（？？？？）

訳注：ここの Type は「データ型」（整数、文字列、等）の「型」というイメージ

（３）プロトコルダイヤグラム（Protocol Diagram）
メソッドを呼び出す為のプロトコル（手続、形式）を明確にする

（４）メソッド分類ダイヤグラム（Method Taxonomy Diagram）
クラスの特性とメソッドの種類との間にある、挙動の類似性とリンクから、
メソッドの種類を分類する（？？？）

（５）クライアントダイヤグラム（Client Diagram）
メソッドの利用者（client）と供給者（supplier）を説明する

また、以上のようなダイヤグラムと合わせて使用される、2 つの特別なデータシートがある。

（１）クラス不変データシート（Class Invariant Data Sheet）
オブジェクトの、ある特定のクラスの各インスタンスに対して適用されている
制約をそれぞれ明確にする

（２）契約データシート（Contract Data Sheet）
あるメソッドセット内のメソッドが満足しなければならない契約を明確にする

このようなダイヤグラムを理解するには、IDEF4 の基本コンセプトの説明が必要である。
このコンセプトについては次の節で説明する。

IDEF4　基本コンセプト

IDEF4 の中にあるコンセプトは、オブジェクト指向の経験のある人にとって、親しみのある物である。殆どのオブジェクト指向言語に存在している物と同一の構造が、IDEF4 の中にも存在している。IDEF4 の中の注目すべきコンセプトには、以下の物がある：

（１）クラス（Clasees）

（２）特性（Features、機能、機構）

（３）継承リンク（Inheritance Link）

（４）型リンク（Type Link）

（５）メソッドセット（Method Set）

クラスは、IDEF4 設計モデルでの、文法の基本単位である。

そしてこのクラスの特徴は、特性（feature）の集合によって表現される。それぞれの特性（feature）は、パブリック（public）かプライベート（private）のどちらかにすることが可能である。パブリック特性は、全てのクラスからアクセス可能であるが、プライベート特性はクラスとそのサブクラスからしかアクセスできない。

オブジェクト指向的方法論（paradigm）の威力は、クラス継承によってもたらされる。継承関係が明確な場合、親クラス（スーパークラス）の全ての特性（feature）は、子クラス（サブクラス）へと受け渡される。その時、サブクラスへと継承された特性（feature）は、その特性が明確に再定義されていない限り、スーパークラスにある物と同じ特徴を維持する。この継承により、簡単なクラスから、複雑な構造を持つクラスに至るまで、構築する事が可能である。図 A-17 は、クラス継承ダイヤグラムの一例であり、図形オブジェクトの簡単なセットにおける、単継承と複継承の効果を示している。

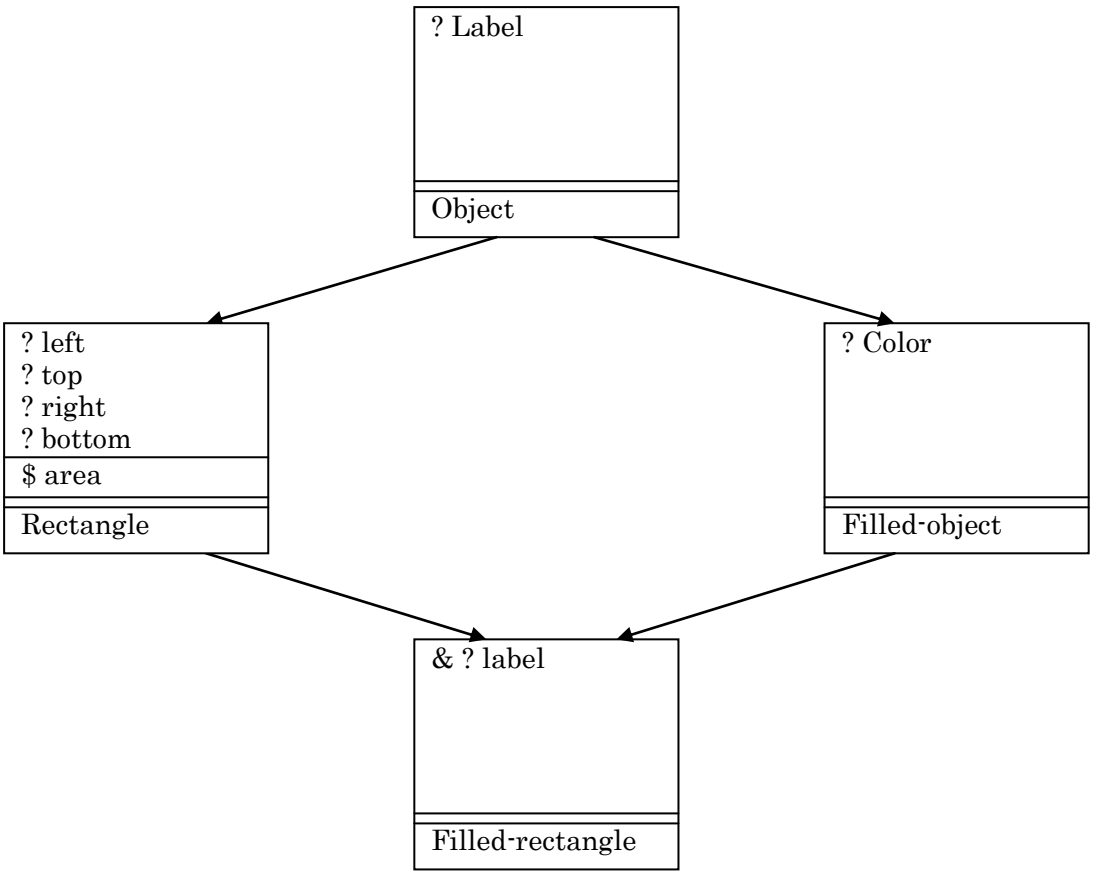


図 A-17　IDEF4 でのクラス継承ダイヤグラムの例

図 A-18（次ページ）は、IDEF4 のクラス継承ダイヤグラム上で、クラスがどのように表現されているかを示している。クラスは、四角形の箱によって表現されており、箱の二重線の下にクラス名が書かれている。IDEF4 では、クラス名の頭文字を大文字にしなければならない。クラスの特性（feature）もまたクラスの箱内に書かれており、プライベート特性は出力線（? export line）の下に、パブリック特性は出力線の上に配置されている。特性シンボル（「?」とか「$」とか「&」といった記号）は、それぞれの特性の果たす役割についての追加情報を与えてくれる。

定義された全てのクラスに対して、設計者はクラス不変データシート（Class Invariant Data Sheet）を添付する。このシートはクラス内のオブジェクトに関する追加情報を提供してくれる。このデータシート上の情報は、いかなる時であっても、クラス内の全てのオブジェクトについての正しいものでなければならない。このクラス不変データシートの面白い特徴は、サブクラスは、スーパークラスのデータシートの制限も継承すると言う事で

ある。

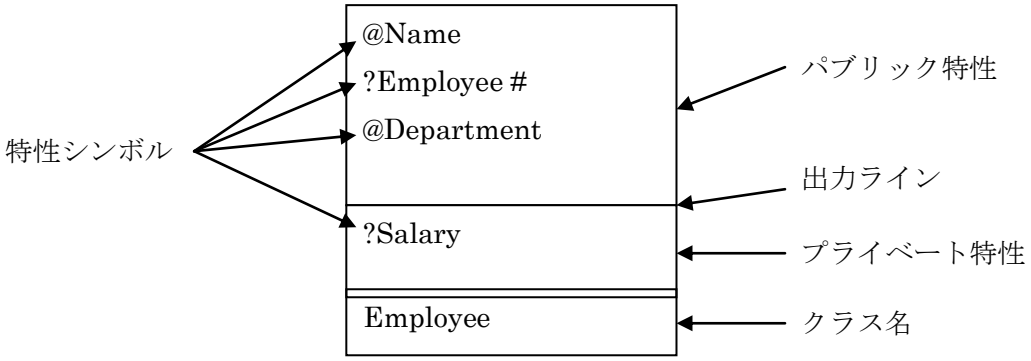


図 A-18　IDEF4 のクラス箱

特性（feature）は、あるクラスの特定の特徴を、名前を付けて表現したものであり、そのクラスのインスタンスの挙動を把握する為に使用されている。そして特性を定義する際には、定義する特性の型（type：データ型の「型」というイメージ）を明確にしなければならない。この時、特性の値の型と、特性の型とを区別する事が重要である。特性の値の型とは、特性が取得する・返す、正当な値（legal value、訳注：整数型、文字列型、浮動小数点型等の、プログラム言語によって予め決められた型であるかどうか、ということか）に関するものを示している。一方の特性の型は、クラスの文脈（context、背景）の範囲内で特性が果たす役割（訳注：１つ、もしくは複数の値によって表現される特性の意味のようなもの？）に関するものである。IDEF4 設計モデルの特性は、以下の 6 種類の型の中のいずれかのものでなければならない。

（１）未定義特性（Non-Specific Feature）
　　設計の初期段階での場所取り（place-holder、場所の予約）

（２）ルーチン（Routine）
　　プログラムとして導入される特性を定義

（３）属性（Attribute）
　　値を固定（hold）するか、値を計算する特性を定義

（４）関数（Function）
　　ルーチンと属性の両方の特徴を持った特性
　　（訳注：一般のプログラムにおける関数（Function）は、返り値を持つルーチン）

（５）手続（Procedure：プロシージャ）
　　返り値の無い、その側のみで実行されるルーチン

（６）スロット（Slot）
　　値を「固定（hold）」した属性

継承リンク（Inheritance Link）は、2 つのクラス間の親子関係を定義しただけのものである。そして継承リンクが定義されると、親クラスの全ての特徴が、子クラスへと継承される。この時、特性が補助特性シンボルを使用して再定義されていない限り、継承された特性は子クラスの中で、親クラスと同じ働きを示す。

クラスはデータ型として考慮することが可能であり、その為、伝統的なプログラミングデータの型をクラスとすることが可能である。属性として分類されるか、もしくはより明確にスロットやルーチンとして分類されたクラスの特性は、値がクラス内部に取り込まれている為、クラスをデータ型として扱う事は、その特性の値の型を指定するのに便利である。このような方法による型の宣言は、型リンク（Type Link）によって行われる。型リンクは、型としての特性（feature being typed）と、属性が持つ正当な値を表現したクラスとを定義している。

図 A-19 は、IDEF4 でサポートされている 4 種類の型リンクを示したものである。

1 つ目のリンクは単純に「A の属性 f は B 型の値を返す」というものである。

2 つ目のリンクは「A の f は B 型の値を返し、B の g は A 型の値を返す」という二重リンクである以外は、1 つ目と同じである。この二重リンクは、型ダイヤグラム上にあるリンクの数を削減し、より単純なダイヤグラムにしてくれる。

3 つ目のリンクは「A の f は B によって構築された値を返す」というものである。これは、B のインスタンスのリストか、もしくは B のインスタンスの構成物である。

そして最後の 4 つ目のリンクは、3 つ目のリンク型の半二重形で、「A の f は B のインスタンスから構築された値を返すが、B の g は A 型の値を返す」というものである。

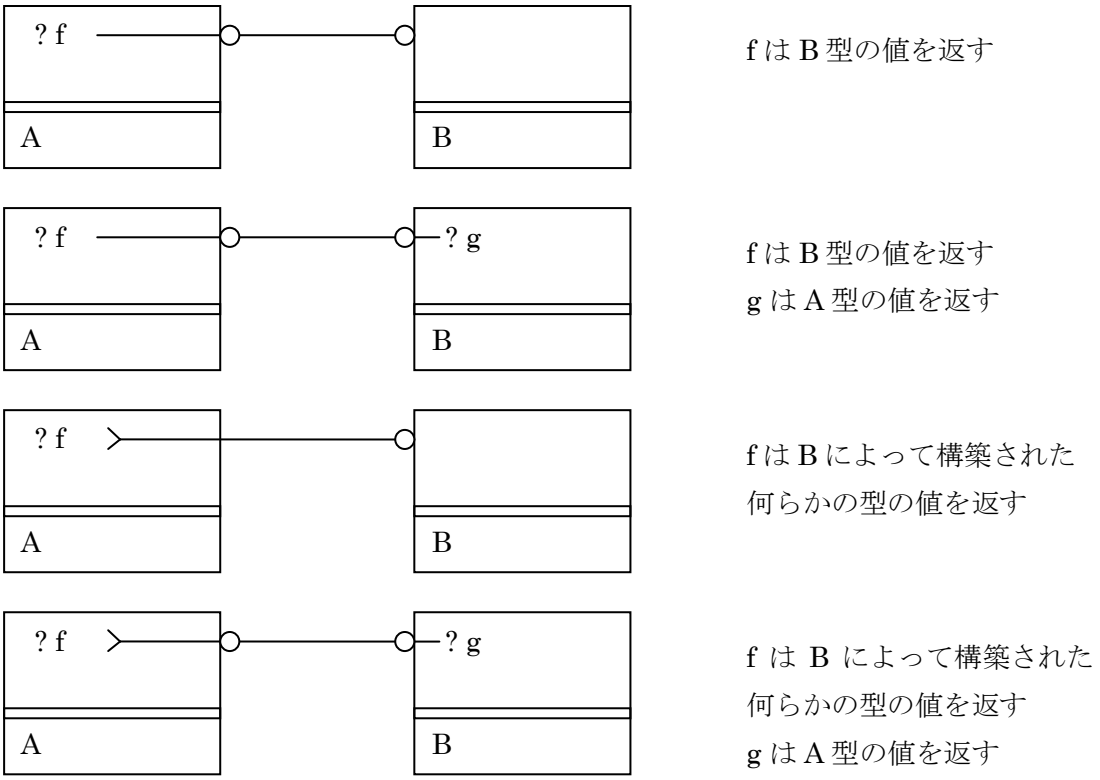


図 A-19　IDEF4 での型リンク（Type Link）のリンクの種類

図 A-20 は、図形的オブジェクトの型ダイヤグラムの例を示している。

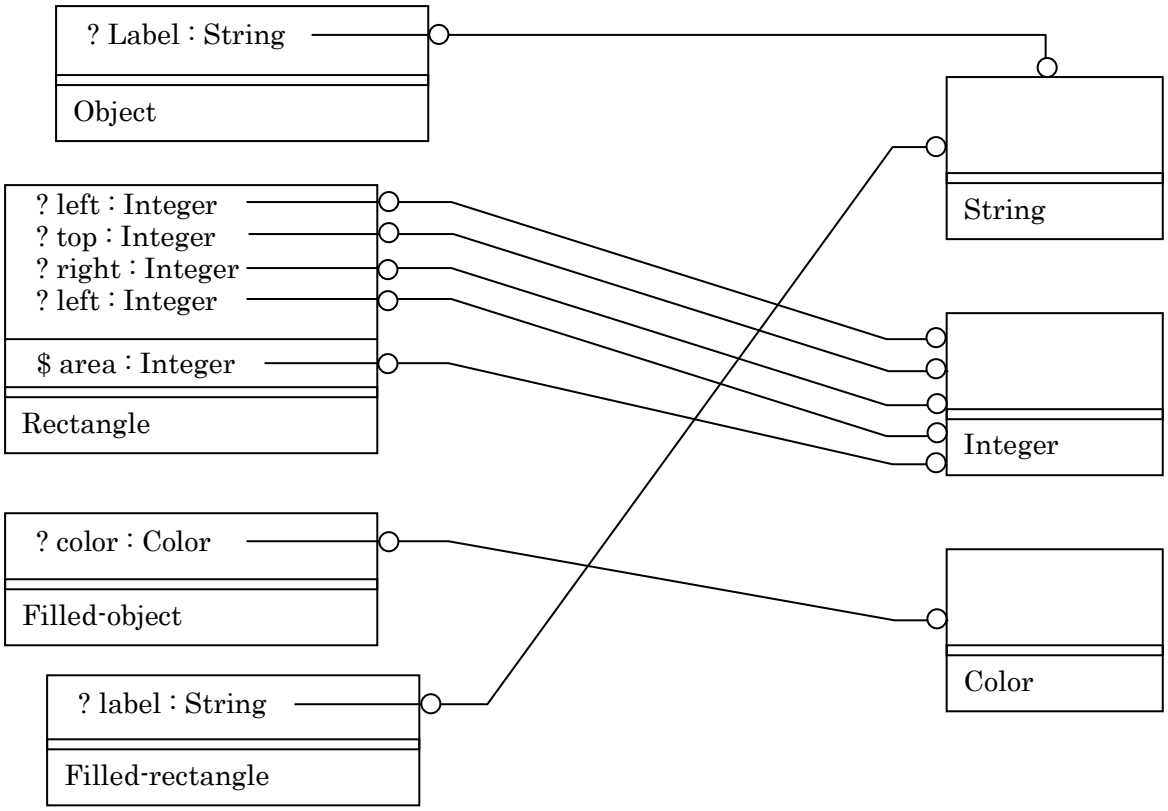


図 A-20　IDEF4 の型ダイヤグラム

IDEF4 では、メソッドを独立したものとして表現しない。その理由は、１つのメソッドが 2 つ以上のクラスのインスタンスをパラメータとして受ける事が可能である為である。その結果、同一のメソッドを、各クラス毎に定義しなければならない。この繰り返し作業と煩わしさを回避する為に、こうしたメソッドの情報を 1 組のメソッドセットとして維持管理している。クラスと、そのクラスの特性とをメソッドセットへと結びつけているが、この結び付けは切換え割当（Dispatch Mapping、ディスパッチマッピング）により行われている。

メソッドセットは、関係契約（associated contract）によって定義される。実際、メソッドセットというものは、契約データシート（contract data sheet）の名前に過ぎない。契約データシートは、メソッドセット内にあるメソッドの意図される効果を定義する宣言（declarative statement）を維持管理している。関数（function）の場合、この契約は関数の引数リスト（argument list）と、それに対する戻り値（corresponding return value）との関係を宣言している。また手続（procedure、プロシージャ）の場合、この契約によって、引数リストと現在の環境が渡された際に、メソッドセットによって環境がどのように変更されるのかについて、定義する必要がある。

IDEF4 を使用する際のガイドライン（訳注：良くわからない）

OOP（オブジェクト指向プログラミング、Object-oriented Programing）ユーザーの多くは、それによって要求分析と要求設計とが不要になるという一方的な仮説（assumption）の下で、OOP 技術を適用している。その結果、伝統的な構造化設計の初期段階において経験したものと同じ種類の文化的問題が、IDEF4 でも発生している（例：経験のある OOP プログラマーは理由を見ようとしない、初心者は概念を理解しようとしない、管理者はライフサイクルプロセスの 1 ステップを省くことを望むことだけを気にする、等)。こうした文化的問題があるものの、IDEF4 手法は行動抽象化（behavioral abstruction）に焦点を置いた支援手法を提供してくれる。しかし新規ユーザーは、型ダイヤグラムを定式化（formulation）する際に、分類学的抽象化（taxonomic abstruction）を使用してしまうことが多い（？)。当然の結果として、型ダイヤグラムとメソッドセットダイヤグラムの間でのメソッドの割当（method mapping）は、非常に複雑なものとなってしまう。ただ、こうした複雑性が出てきた場合に、IDEF4 ではクラス構造を更新する必要があると、設計者に指示されるようになっている。最終的に、IDEF4 は設計プロセスを支援するように作られている。IDEF4 の仮設は、不活発な設計決定プロセスを支援することである。しかし IDEF4 ユーザーにとっては、メソッドを単に、設計をドキュメント化する為の手段として扱うことは、極めて当たり前の事であり、既にコード化されていることも多い。

IDEF5 における、概念／存在論記述（Concept / Ontology Description）

（訳注：IDEF5 に関しては、全体的に良くわからない。IDEF5 の本文を別途読む必要有）

情報の維持管理という意味合いにおける「存在論（ontology）」の役割は、対象となるエンジニアリング、製造、営業、流通等の分野に存在する本質と構造（nature and structure）を抽出し、それを利用しやすい表現媒体（representational medium）の形で保存することである。中でも、現実世界と、人や場所、出来事等の間に存在する関係性を、効率的に獲得可能な手法が、特に必要とされている。実態学的な情報獲得の重要性は、大規模なシステムの場合に決定的なものとなる。大規模な CIM プロジェクトでは、幾つもの法人組織の、多くの異なった集団（cluster）からもたらされるリソースを協調させなければならない。各集団が自分自身の役割に貢献するのは当然であるが、しかしプロジェクト全体で成功を収めるには、こうした異なる集団を、開発プロセスを通じて上手く統合して行かなければならない。この効率的な統合のカギとなるのは、集団間でアクセス・更新し、異なった集団がそれぞれ持つ目標に関係したシステム全体の共通の特性（feature）の獲得を可能とする、システム存在論である。この「共通な」フレームワークは、①システム内の様々な発生源からもたらされる情報の共有を促進し、②情報基盤の維持管理を容易にし、③一度収集された情報の再利用性を強化してくれる。

存在論（ontology）に最も期待されているのは、信頼性の高いシステムを簡単に入手することである。情報の運用管理における最も重大な問題は、システム内で既に記録済み、開発済みの情報の、収集と再作成に消費されている冗長的な作業の存在である。しかも、この冗長的努力の殆どが報われないのである。いやむしろ、開発チームが、状況間の類似性や同一性の認識に失敗した結果が、こうした冗長性なのである。

IDEF5 の目的は、そうした類似性を手動・自動の双方で認識する為の基盤（basis）として使用可能な、参照モデルの構築である。IDEF5 はまた、経営情報分析（enterprise information analysis）のたたき台（precursor、先駆者）としても使用されている。IDEF5 は、物理的・概念的対象物についての生の知識を、その対象物の本来の関係（natural association）と共に、記録、構成する為の構造も提供してくれる。このファクトベース（fact base：ルール状態が顕かにされた事実を収集したもの。事実を収集しただけのものではない）は、こうした概念についての情報含有物（information implication）の測定（determination）も含めた、様々な分析目的の為の基盤として利用する事が可能である。

そして最後に、存在論分析は、重厚な知識ベースシステムの構築における、効率的な第一歩であることを証明している[Hobbs 87]。CIM 導入（implementation）の最新バージョン（current generation）は、知識ベースシステム技術と専門家システム技術（expert system

technology : 専門家の知識や手法をコンピューターに移植したシステム）の利点を活かしたものになる。IDEF5 はこうしたシステムの為の、初期段階における知識獲得の為の手法と、いかなる実行シェル（implementation shell : コンピューターのシェル？）からも独立した、獲得知識の表現媒体を提供している。

存在論の研究（inquiry、調査、追及）は、情報科学における広範な研究作業の主題であった。IDEF5 の開発は、セマンティックネット（Semantic Nets : 意味ネットワーク）や状況理論（Situation Theory）、NIAM オブジェクト役割モデル（Object Role Model）、IDEF1、セット理論（set theory）、一階述語論理（first-order prediate logic）、モデル論理（model logic）等の、基礎的研究作業が基になっている（訳注 : ほぼわからん）。そうすることにより、IDEF5 は他の既存手法（IDEF0～IDEF4 の事）の対象範囲から漏れた手法論を補完しようとしている。IDEF1 と IDEF1X は主に構造化された情報を、IDEF0 と IDEF3 は様々な種類のプロセス情報を、それぞれ獲得している。もちろん、構造化情報もプロセス情報も、どちらもシステム内の対象物が関係しているものの、既存の手法論では存在論的表現が制限されていた。しかしこれから述べるように、既存の手法論では表現できなかった幾つかの重要な種類の存在論的情報が有るのである。更に、そうした既存の手法論には、システム存在論を引き出し、獲得する為の特別な技術は含まれていなかった事から、その為の別の手法が必要であると提案されているのである。

IDEF5 は、ある領域の、概念と概念的関係性をモデル化する。概念的モデルは、問題対象となる領域もしくはシステムを記述する為の表現の抽象化レベルを提供してくれるが、この抽象化レベルは、現実世界の対象物や、属性、機能（function : 関数？）といったものの表現も含む、その領域における人間の概念化（human conceptualization）を密接に反映している。存在論は、その領域に何が存在しているのか、という理論（theory）を表現しているのである。

IDEF5 の観点からのシステム記述

存在論（ontology）を、それぞれが互いに関係し合える状態にある異なる面から知覚された世界についての知識を、表現する為の構造体として考える事が可能である。また、存在

論は、概念、対象物、そして関係（associations）の認識と分類並びに、そうしたものの種類や関係の識別を行う本質的な特徴と、関連している。

ある分野に対する存在論の定義は、以下の 4 つの主要な活動から行われている [Menzel 91]：

（1）対象となる分野に関する最良の情報源（その分野の専門家等）に従った、
その分野に存在する対象物の種類についての、棚卸リスト（inventory）の提供

（2）対象物の種類毎の、全体で共通な特性と、その種類のインスタンスのみが持つ
特性の記述の提供

（3）あるシステム内で実際に種類のインスタンスを作成している特定の対象物の、
特徴づけ（characterizing）

（4）対象となる分野内の対象物の種類の間（と種類の内部）に存在する
関係（association）の、棚卸リストの提供

例えば、半導体の製造について考えてみよう。最初の 2 つの作業は、ウエハースと試薬を含む、対象物の種類の識別である。この試薬は、更に液体試薬とエッチング液を含む幾つかの種類へと細分化される。そして、細分化された個々の種類は、その種類のメンバーに含まれる為に必要となる特性の関係セットを、それぞれ持っている。

3 番目と 4 番目の存在論に関する作業は、ある個別の対象物の特徴づけをしたい、対象物やその特性、関係性について特に表現したいと、我々が望んでいる場合には、より身近なものとなる。

IDEF5 に備えられている基本的な区別（distinction）は、本質的な特性と、非本質的な特性との区別である。ある対象物の本質的特性は、単純に、その対象物が欠くことのできない特性である。次節では、この IDEF5 の特質が、天然物と人工物の両方を含む膨大な組み合わせを取り扱わざるを得ない製造分野で、使いやすい存在論を素早く開発することに対して、どのような支援を行っているかについて述べてゆく。

IDEF5 の基本概念

「種類（kind)」という用語は（クラスや型（type）とは区別されたものとして)、IDEF5 の中心的な概念となっている。その為、通常用いられる意味としての「種類」と IDEF5 における「種類」との区別を理解する事が、まず重要である。

自然発生的なシステム（naturally occuring systems）においては、同じ種類に含まれる全ての対象物が、ある 1 つの特徴立った（distinguishing)、その特性が無ければその種類内に留まれないという特性のセットを、持っている場合が多い。これはつまり、その種類のメンバーである為の特性は、メンバーの本質的な特性であるということである。そのため、通常用いられる「種類」という用語は対象物の集合体であり、その集合体に含まれている全ての対象物は、種類に属している全ての、かつ種類に属している対象物のみが本質的に持っている特性のセットのような、共通の性質（nature）を持っている。

しかし製造システムでは、ある種類の一部として参加する為に、対象物は何らかの明確な特性セットが必要とされるものの、その種類に属し続ける為に、その特性セットは必要とされない、という場合が多い。

前で述べた半導体の製造の例で考えてみると、薬品（chemical）は、エッチング液であると認識する為の明確な幾つかの特性を持っており、そして全てのエッチング液は、それらの特性を持っている。これは、本質的な種類についての、伝統的な考え方である。この伝統的な考え方に対し、対象物の種類とは、製造の「再加工（rework)」品（item）が表現するところのものである。

4 つ以上の傷を持つ全てのウエハースは再加工品であるとする。この時、4 つの傷を持つウエハースは再加工品となる。しかし、このウエハース上の 1 つ以上の傷を修正しても、このウエハースは再加工品のままなのである。実際、監査官により、受領可能なウエハースとして、もしくは廃棄品として再分類されるまで、このウエハースは再加工品であり続けるのである。これは、人工的なシステムで典型的に発生する「種類」の一例である。
IDEF5 は、この両方の「種類」の識別を対象としている。

言換えるなら、より広範な観念として「種類」を捉えている理由は、人工的なシステムの存在論を作り上げる際に、システム内にある世界をあるがままに発見（discover)、分類（classify）するだけでなく、むしろシステム内の対象物を、使いやすく情報価値が高くなるように分配（divide up)、分類（categorize）しているからである（訳注：classify と categorize の違いは、次ページの表を参照)。存在論の分類スキーム（categorization scheme）は、そのように分類する事により、システム内にある知識や情報の組織化、運用管理、表現が容易になるという点においてのみ正当化されている。仮に K という種類に属する対象

物が、システム上で便利な役回りを演じているのであれば、K の特性の定義が K のメンバーにとって本質的か否かとは無関係に、その事が、K という種類をそのシステムの存在論に編入する理由の全てとなるのである。(考えてみるといい。あなたのマネージャーが、マネージメントのやり方を知っている必要があるかどうかを（訳注：マネージメントのやり方を知らなくてもマネージャーはマネージャーだ、ということ？))

訳注：categorization と classification の違い

どちらも日本語は「分類」だが、以下のように意味合いが異なる

	Categorization	Classification
プロセス	状況（context）もしくは知覚された類似性を基にした、実体の独創的合成体	必要十分な特徴の分析を基にした、実体のシステム的配列
境界	どのグループも資格が縛られていない為、境界は「曖昧」	クラスは完全に排他的で重なりが無い為、境界は固定されている
資格	柔軟： カテゴリー資格は、一般化された知識や直接的状況（immediate context）を基にしている	厳密： ある実体が特定のクラスに属するか、属さないかは、クラスに内包（intension）されているかどうか、を基にしている。
登録基準	状況（context）への従属性・独立性の両方を基準とする	予め定義されたガイドラインや原理を基準とする
典型性	同一カテゴリー内のメンバーを、典型性によって順位づけすることが可能	あるクラス内の全てのメンバーは平等
構造	実体のグループは、階層構造を形成することもある	固定されたクラスによる階層構造

https://pdfs.semanticscholar.org/774e/ab27b22aa92dfaa9aeeeafbe845058e85f58.pdf

「Classification and Categorization: A Difference that Makes a Difference」ELIN K.JACOB 著　より

しかしながら、システム内の対象物の特性をリスト化するよりも、対象物に特徴づけする事の方が多い（？)。対象となるシステムによっては、システム内の対象物が互いに結ぶことが可能な、もしくは結んでいる関係（association）を詳細化する事も、同様に重要となる。そして特性と同様に、システムに本質的な関係と、システムに非本質的な関係とは区別されなければならない。これは関係が特性と同様に発生していること、また関係が、種類の特性となるかもしれないこと等が理由である（例えば、結婚関係（marriage association）と、「結婚している（married)」という種類)。K1 と K2 の 2 つ（かそれ以上）

の種類の間のシステムに本質的な関係とは、K1 と K2 のインスタンス（instance）が存在している間中、保持し続けなければならない関係である。それに対し、K1 と K2 の間のシステムに非本質的な関係は、2 つの種類で発生し得る全てのインスタンスが存在する間中、保持し続ける必要がない関係である。種類の特性の定義が、その種類のインスタンスに対して本質的である必要が無いのと同様に、システムに本質的な関係の上にある対象物は、本質的に、そうした（システムに本質的な）関係の上に必ずしも立っていないのである。そしてこれは、人工的なシステムにおいて、特に顕著である。

訳注：
種類の間に結ばれた関係（association）の本質性について記述され、それが重要であることはわかるのだが、肝心の内容を理解できない。具体的な例が幾つかあれば理解しやすくなるのだが、それには IDEF5 の本文を読むしかない。

IDEF5 では、存在論（ontology）を可視化する為に、3 種類のダイヤグラムを提供している。これらのダイヤグラムは、存在論の構築と確認のどちらにも便利である。
（訳注：以下に挙げてある 3 つのダイヤグラムは、この論文が書かれた時点では存在したのかもしれないが、その後の資料では、別の物に置き換わってしまっている）

1 つ目の「is-a ダイヤグラム」は、IDEF5 モデル内の種類間にある「is-a（～は～である）」関係を示す為に使用されている。IDEF5 には、次の 3 種類の is-a 連結（link）がある。
（この連結は、論文[Brachman 83]のものを用いている）

（1）一般化／特殊化（generalization / specialization）
（2）あるものの一種（AKO、a kind of）
（3）包摂（subsumption、ある事柄をより一般的な範疇に組み入れる事）の記述

一般化／特殊化（上位集合／部分集合（superset/subset）とも呼ばれている）連結は、ある種類の、他の種類からの特殊化を表わしている。例えば、六角ボルトという種類は、留め金具(fastener)という種類の中から、六角形をしたボルトとして特化したものである。AKO 連結は、自然的な（natural）種類の表現に便利である。例えば、犬という種類は、哺乳類の一種である。包摂記述連結は、抽象的な種類の表現に便利である。「正方形は、4 辺が等しい長方形である」という事実は、包摂記述連結によって捉えられている。

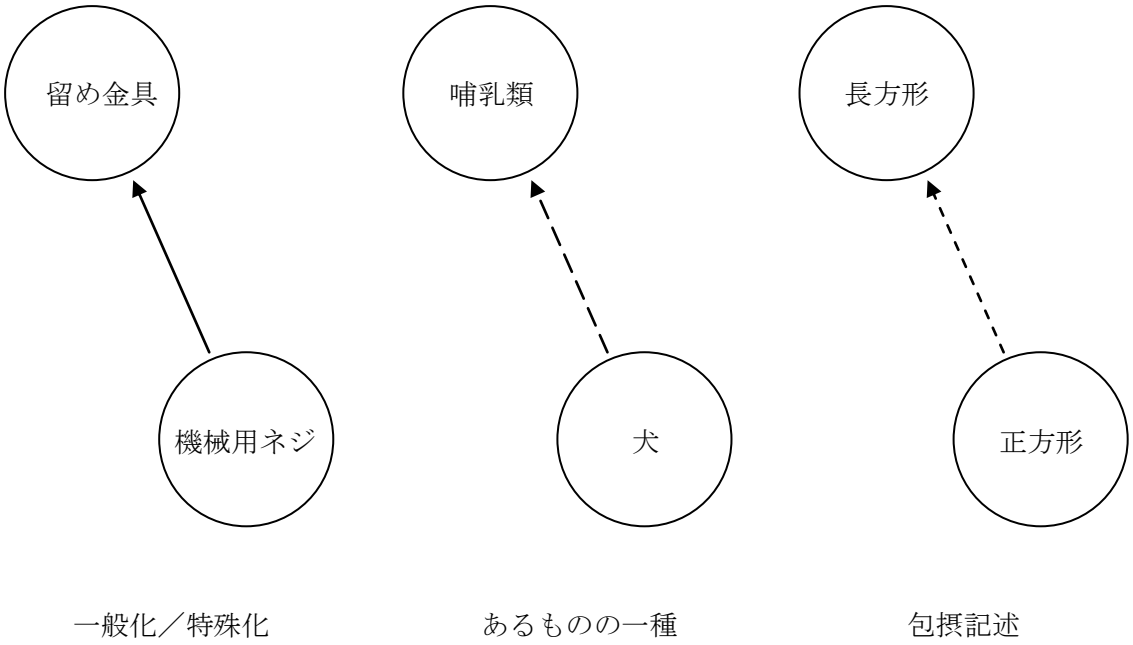


図 A-21　is-a 結合の形式

2 つ目の「システム種類ダイヤグラム（system kind diagram)」は、システムを構成している種類と関係とを表現する。図 A-22 は、「ウエハー切断システム」のシステム種類ダイヤグラムの例である。システム内の種類は円で、システム種類は二重円で、それぞれ表現されている。円の間の線は、矢印方向への関係であり、システムの本質的な関係は二重線で、非本質的な関係は単線で表されている。(訳注：直訳してみたものの、図 A-22 の意味が分からない)

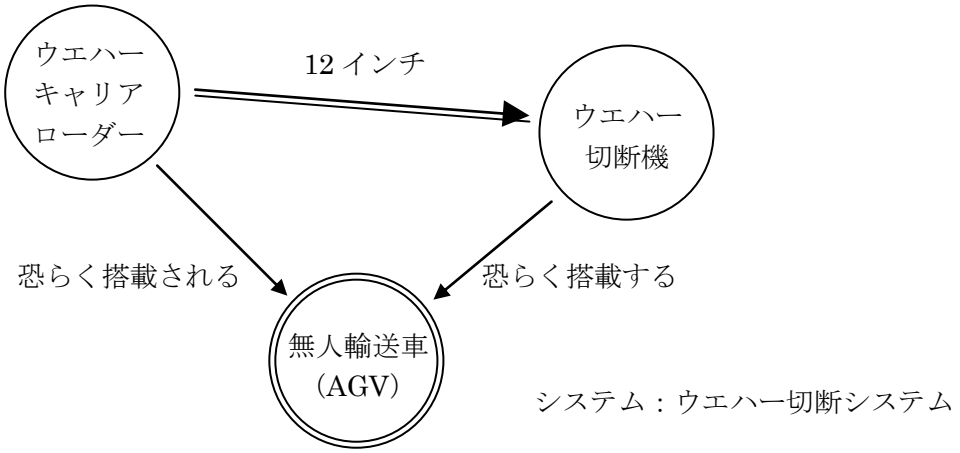


図 A-22　システム種類ダイヤグラム

3 つ目の「関係型式ダイヤグラム（relation type diagram）」は、モデル内の関係を公理化（axiomatization）したものを表現している。ある関係を公理化することにより、システム内の他の関係の立ち位置から、その意味を記述してゆくのである。これは伝統的に、存在論の開発の中で最も難しい部分である。関係型式ダイヤグラムの背後にある考え方は、関係の定義の中心的なセットを、最大限再利用するということである。図 A-23 は「シーラント有り（has_sealant）」の関係を表わしたものである。（訳注：当然に訳が分からない。シーラントはホームセンターで良く見かける、シリコン補填材のこと）

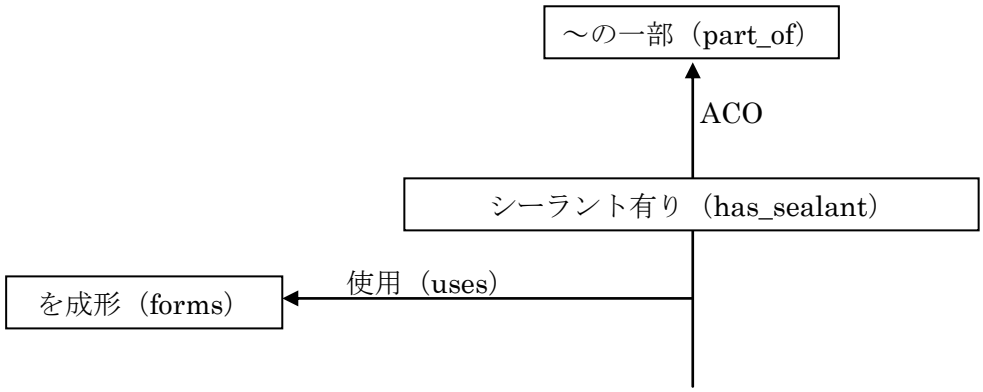


図 A-23　関係型式ダイヤグラムの例

「シーラント有り（has_sealant）」関係は、「留め金具（fastener）」種類と「シーラント（sealant）」種類の 2 つの種類の間に発生する。この関係は、自動車分野における、用途次第でシーラントを要求する留め金具が存在する、という事実を反映している。2 つの関係形式（relation type）の指示子（spesifier）は、図にあるように、「ACO（abides by the contract of、契約に従って留まる？）」と「使用（uses）」である。ACO 連結は、存在論開発者に対して、他の記述内にある 1 つの関係を特徴付けしている原理（axioms）の再利用を可能にしてくれる。また使用（uses）連結は、関係の特徴（characterization）間の相互関係の決定を容易にしてくれる。

（訳注：この部分は、全く意味が分からない。IDEF5 の本文を参照のこと）

3 つの型式のダイヤグラムに加え、IDEF5 には、手法の構築と適用の為の、10 段階のプロセスも含まれている。手法の論理的基盤についてのより詳細な説明は、[Menzel 91]に書かれている。また、実際の手法についての徹底的な記述は、[KBSI 91]に書かれている。

IDEF5 を使用する際に考慮すべき点

IDEF5 は、ある分野における存在論開発の、プロセス機構の基本概念を提供する。このプロセスによって得られる最初の経験から（？)、存在論開発作業が、以下の活動で構成される反復作業になる事がわかる：

（1）事実データの収集と分析
（2）「最初期種類（proto-kinds)」（最初に推測される種類）の発見
（3）最初期種類とその関係物の改良
（4）元々の事実に対して、出来上がった存在論が合っているかの確認

これまでの教訓から、ステップ 2 と 3 は、適量に分割されたデータを 1 人だけで処理するのが最適であり、ステップ 1 と 4 はモデル作成者のチームとして作業する方が容易である。

IDEF6 による、情報システム設計の理論的根拠の獲得

Information System Design Rationale Capture With IDEF6

技術や製造手法、素材が進歩したことにより、使用期間が数十年、数百年という製品が出現するようになった。情報システムもまた同様に、製品生命が比較的短期間で、対象も限られていた、システムに依存したスタンドアロンアプリケーションから、広範囲を対象とし、幾つものシステムに跨ったユーザーに対して、長期間に渡ってサービスを提供するものへと進化している。これまでの製品とは異なり、何世代にも渡る長期間の製品生命が要求されている情報システムの維持管理には、システム設計において、論理的根拠（rationale）の明確な獲得と蓄積が必要である。

明確に獲得された設計の論理的根拠（rationale）は、一般的に、構造化されていないテキストコメントの形で存在している。更に問題を難しくしているのは、設計の論理的根拠の獲得に必要となる完成した評価基準を、体系化し提供する、構造化された手法が存在していない為、たとえ要求された関連情報を発見できたとしてもドキュメント化が出来ないのである。

IDEF1X や IDEF2、IDEF4 といった、「何（WHAT)」を設計するのか、をドキュメント化する設計手法とは異なり、「何故（WHY)」設計がそのようになってしまったのか、もしくは「何故（WHY)」他の形式にならなかったのか、そして「どのように（HOW)」して最終的な設計要目へと至ったのか、を把握する新しい手法が必要となるのである。

この論文においては：
「設計仕様（Design Specification)」は、「何」を設計するのかの把握を、
「設計の理論的根拠（Design Rationale)」は、設計が最終形態に、「何故」至ったのか、
　「何故」至らなかったのか、「どのように」至ったのかを、そして
「設計履歴（Design History)」は、設計が現実化されてゆく過程での、
　　時系列で並べられた作業手順を、
それぞれ示している。

IDEF6 は、情報システム設計の理論的根拠を把握し、その理論的根拠をエンドシステムの為の設計モデルとドキュメンテーションへと関連付ける、具体的能力を伴った手法となるように意図して作られている。その為、IDEF6 は、最終的な設計へと寄与した、もしくはそこへと至る原因となった決定の、背景にある論理を把握することを試みている。設計の理論的根拠を明確に把握する事により、過去の間違いの繰り返しを回避し、設計変更の影響を判断する為の直接的な手段を提供し、目標と仮設を常に明確なものとし、最終的な

システム仕様に関する情報交換を補助するのである。何故、設計者は、経営レベルの情報システムに対して、その設計戦略、もしくはシステム特性（system feature）を選択、採用したのかについての動機を明確に把握する事は、そのシステムを製品生涯を通して維持管理する為には欠かせないのである。

IDEF6 の視点からの、設計の理論的根拠

IDEF6 の目的は、経営レベルの情報システム開発で使用される、設計の理論的根拠の獲得、表現、取扱いを促進することである。この「理論的根拠（rationale）」という用語は、設計者が、その戦略もしくはシステム仕様を選択、適用するように向かわせた「理由、正当化、潜在的動機（underlying motivation）、もしくは言い訳（excute）」として言換える事ができる。更に簡単に言うならな、「理論的根拠」は「何故このような設計になったのか？」という疑問に対する回答の本質（nature of the answer）である、とすることも可能である。IDEF6 は、システム設計の理論的根拠を構成し、その理論的根拠をシステムの設計仕様、モデル、ドキュメンテーションへと関連付けている、「状況、関係、対象物、出来事の状態（states of affairs）、もしくは出来事の推移（course of events）」に関する情報を表現する為に必要な、概念と言語能力を、備えた手法となることを意図している。

IDEF6 を構成する技法の適用可能範囲は、情報システム開発プロセスにおける概念設計、初期設計、そして詳細設計である。ソフトウェアシステムの詳細設計決定が、コーディング段階（coding phase）へと委託されているのであれば、ソフトウエア構築プロセスにおいても、IDEF6 技法を使用可能なものにすべきである。

IDEF6 の適用範囲に関係する仮説（assumption）には、以下の物が含まれている：
（訳注：以下もわけわからん）

（１）IDEF6 は、システム設計から、実行データ構造（implementation data structure）・アルゴリズム・ユーザーインターフェース・プロセスといった詳細設計に至る、経営レベルでの情報システム設計の理論的根拠の獲得を促進することを目的としている。

（２）必要となるまでの間に、設計者が設計の理論的根拠を別途「モデル」してくれることを期待するのは非合理である。理論的根拠が発生した時点で（決定がなされた時点で）たちまち獲得しなければならない。

（３）紙上（write down）で設計仮説（design assumption）や理論的根拠を作り出すことは、ほぼ不可能である。出来る限り、IDEF6 は判り易い手法と合体させ、幅広い各種の設計手法の一部としておくべきである（公式にも非公式にも）

（４）設計の理論的根拠は、あくまでも開発決定の理論的根拠の一部分でしかない。その為、「何が重要な兆候（symptoms）なのか」という決定と、そうした兆候の基になる問題が何なのかを定義している決定を、設計の理論的根拠が参照するようにしておかなければならない。

この、設計の理論的根拠の一般的な特徴は、

「人間が設計の約束（commitment）を作る為に、また、その約束を普及させる為に
　使用する、信念と事実（beliefs and facts）、並びにそれらの組織的構成物（organization）」

であるとする事が可能である。

IDEF6 は、設計の理論的根拠の「型式（type）」と、こうした型式を表現する為の「機構（mechanism）」の両方を、特徴付けしている。IDEF6 による獲得の際に識別される、設計の理論的根拠の型式には、以下の物がある：

（１）次の物を含む設計哲学（philosophy of a design）：
　　A）意図されたシステム運用におけるプロセスの記述
　　B）対象物もしくは関係する型式の立ち位置からの、設計テーマ

（２）システムパラメータもしくは環境要因（factor）による制限範囲（range restrictions）として表された、設計限界（design limitation）

（３）トレードオフ決定で考慮されている要因（factor）

（４）以下の立ち位置から表現されている設計目標：

A）特定の構成物（component）の使用、不使用
B）問題要求における優先度
C）製品のライフサイクルに関する特徴（廃棄か維持か、等）
D）問題点や解決方法の空間分割、テストデータ／モデルデータの変換、もしくはシステム構築、等における設計ルール

（５）実行可能性の優先度（prefedence）もしくはその歴史的証明（？historical proof）

（６）立法上（legislative）、社会、専門社会、経済、もしくは個人的な点における、評価要因（factor）もしくは制約（constraint）

恐らく、ありきたりな持越し戦略（the commonness of the carry-over strategy）が原因なのか、もしくは設計の理論的根拠表現の複雑性が原因なのか、設計に与えられる最も一般的な理論的根拠は、「去年は機能していた設計だから（the design that worked last year）」というものになっている。この状態が良いか悪いかは別として、設計知識の理論的根拠を獲得する為に最低限必要な物は、現在の有る部分の設計状態が前の設計の状態と同一である理由についての確信（belief）と正当性（rationalization）の宣言だけでなく、そこに至った時系列（historical precedence）をも、記録する能力である。それ以外の重要な設計の理論的根拠は、「良さそうだから」とか「バランスや対称性がより高い」といった物でしかない。そこには、ソフトウェア設計に対する重要な審美的（aesthetic）側面がある。

ソフトウェア設計の理論的根拠には、開発プロセスそのものを通して、設計がどのように進化して行くのか、ということに関する期待が含まれている。例えば、プログラム構造は恐らくこのように変化するだろうという期待である。ただし、そうした期待は、我々が機械的ハードウェア設計で見てきたような期待と同程度に良く定義されているようには見えない、ということに注意しなければならない（相当に意訳）。

IDEF6 の基本概念

このように、IDEF6 は、設計の理論的根拠の宣言の種類毎（設計哲学、範囲制限・制約、トレードオフ要因、設計目標、等）の定型化（formulation）の為の、構造化された技法（structured technique）として見る事が可能である。また、他のライフサイクル製品や対象物に簡単な参照付けをするだけで、そうした宣言の定型化を行う為の支援も可能である。つまり、ある設計要素の理由が、ある要求制約を満足させていれば、IDEF6 では、要求制約への参照による、こうした関係性のみを宣言することが許容されている（IDEF6 言語では、要求制約を再作成する必要が排除されている)。IDEF6 は、未だ正式化作業の途中にある。現時点において、以下のような目的の為の言語を構成しようとしている：

（1）理論的根拠の宣言
（2）設計要素と関係する理論的根拠の宣言
（3）設計要素と他のライフサイクル対象物との間に張られた「理論的根拠」の
　　作成と分類

IDEF6 は紙の書類上（purely manual form）でも適用可能であるが、最も適しているのは、ライフサイクル製品の設計データベース（repository）を含む、デジタル化環境下（automated environment）での適用である（IBM の AD サイクルや、DEC の設計データベース、空軍の統合開発支援環境（IDSE、Integrated Development Support Environment）等)。

IDEF6 言語は、設計の理論的根拠の存在論（ontology）を基にしている。その為、IDEF6 言語には、理論的根拠の要素の表現で一般に使用されている用語もしくは句（phrase）のセットが（キーワードとして）含まれている。「満足する」という用語と「～によって満足させられている」という句の場合、以下のような構造で使用される：

（1）設計特性 A は、要求 B を「満足する（satisfies)」。
（2）要求 B は、設計特性 A「によって満足させられている（is satisfied by)」。

その他の、設計の理論的根拠の存在論（ontology of design rationale）で考慮しなければならない用語／句には、以下のようなものがある：

システム（System）
サブシステム（Subsystem）
構成物（Component）
要求している／によって要求される（Requires / Is Required By）
制約している／によって制約される（Constrains / Is Constrained By）
囲んでいる／によって囲まれる（Bounds / Is Bounded By）
支援している／によって支援される（Supports / Is Supported By）
作成している／によって作成される（Creates / Is Created By）
変換する／によって変換される（Translates / Is Translated By）

IDEF6 言語は、英語に似た簡単な構文を用いていることから、上に挙げたような「理論的根拠が成形する」構造を、特定の CIM システムの設計に関係した宣言へと織り込むことが可能である。

設計の理論的根拠を獲得する上での問題点

IDEF6 は、未だ正式化作業の途中であることから、ここでは、開発プロセスの中で発見された、適用に関係する要因について説明するに止めておく。

例えば、設計者が設計プロセスの中で、対象となる設計決定を基に、仮説や理論的根拠をドキュメント化、もしくはモデリングを、別のステップによって実行する事を期待するのは、不合理な事である。その為、こうした情報の獲得は殆どが、設計者本人ではなく、バックグラウンドプロセスもしくは対面での聞き取り作業（interactice questioning）を通した、設計支援環境によって行われているのである。このことを考慮し、IDEF6 手法の開発は、幾つもの異なったシステム設計手法と同時に使用されることになるという仮定の下で、行われている。

表面化している、もう一つの重大な問題は、設計の理論的根拠は、最初に行われる開発決定の動機となった理論的根拠の一部でしかない、ということである。その為、設計の理論的根拠は、そもそものシステム開発決定の動機となった兆候（symptom）と、その兆候の原因と推測される物の両方を、参照しておかなければならない。

まとめとして（Closing Remarks）

図 A-2（P.8）のように、この論文では、CIM システムの分析と開発に使用されている手法と、それに向けて開発中の手法について説明してきた。下の図 A-24 は、現在開発中の IDEF 手法の一覧である。これを見て読者は、その数の多さと、そして既存のシステムもしくは開発予定のシステムについて必要な全ての事象を表現可能な包括的な手法が欠けている事に、疑問を持たれるかもしれない。直観的に、システムの全てを 1 つで表現可能な手法を持つことが良いように思えるかもしれない。しかし、僅かに異なった面それぞれから、モデルと記述とを作成しなければならないのか、という理由を考慮すれば、それが誤った考えであることが判るだろう。

IDEF0	機能モデリング	Function Modeling
IDEF1	情報モデリング	Information Modeling
IDEF1X	データモデリング	Data Modeling
IDEF3	プロセス記述獲得	Process Description Capture
IDEF4	オブジェクト指向設計	Object Oriented Design
IDEF5	存在論記述獲得	Ontology Description Capture
IDEF6	設計理論的根拠獲得	Design Rationale Capture
IDEF8	ユーザーインターフェース モデリング	User Inteface Modeling
IDEF9	シナリオ駆動 IS 設計	Scenario-driven IS Design
IDEF10	適用技法モデリング	Implementation Architecture Modeling
IDEF11	情報人工物モデリング	Information Artifact Modeling
IDEF12	組織モデリング	Organization Modeling
IDEF13	スリースキーママッピング設計	Three Schema Mapping Design
IDEF14	ネットワーク設計	Network Design

図 A-24　開発中の IICE 手法を含んだ、IDEF 手法一覧

一般的に、モデルと記述は、決定を補助する為のものである。それぞれの形式のモデルや記述は、システム全体の中のある特定の視点や見方を構成する、関係性とシステム特性の比較的限定されたセットに焦点を絞ったものになっている。例えば、分析モデルは、既

存設計もしくは準備中の設計の要求を決定する為に使用される。設計モデルは、システム要求の制限されたセットに対する、要望可能な設計特性の最適化を促進する。シミュレーションモデルは、運用状態という制限されたセットの下にある、特定の能力特性を調査する為の、システム能力に関する各種の計測方法と統計を作成する事が可能となる見方を、提供してくれる。こうした各種のモデルと、そのモデルの構築を通して成された決定は、システム全体レベルでの決定に対して、相対的な重みが与えられている。そしてモデル内とモデル型式間で強調された設計決定の、競合したものが次第に明らかになり、トレードオフ解決が必要になってくるのである。

このプロセスの目標は、対象となるシステムの最適設計（optimal design）である。もちろん、現在の値のセットが、トレードオフ決定を何らかの形で満たすことをもって、その設計やシステムが最適化されたものとして考慮される。これは最適設計が、全ての望ましいライフサイクル特性を、もしくは全ての能力特性を、必ずしも満たす必要はないということを、そしてまた殆どの場合は、満たすことが無いということを、意味している。

その結果、モデルと記述とは、制限されたシステム特性のセットに焦点を置き、手近な決定に直接関連しない特性を、明確に無視している。モデルと記述とは、システムで発生し得る状態や挙動の全てを表現することを、決して意図していない。仮に、そのような目標が達成可能なものだとすると、試作品を作成すること自体が、実際のシステムを構築してしまうことになり、モデリングの利点そのもの（低コスト、予想される能力の短期間での評価、等）が否定されることになる。モデリングの境界を越え、現実世界での実際のシステム構造へと拡張してしまうと、シミュレーションは設計決定プロセスというよりも、むしろ統計の実行になってしまう。

全ての関係するシステムライフサイクル特性とシステム挙動特性を表現できる単一モデルを求めようとすれば、設計プロセスを完全に飛ばしてしまう必要がある。同様に、単一の手法もしくは単一のモデリング言語によって概念化やシステム分析、システム設計を促進しようとすると、失敗を招くことになる。典型的に、狭い範囲しか対応していない特殊な目的の手法を、あまりにも多用し過ぎてしまうと、仮に他の手法との統合の為のメカニズムを持っていたとしても、同様に失敗の原因となる。IDEF ファミリーの手法は、特定の問題の種類に対してのみ効率的に適用することが可能な特殊目的手法と、必要なすべての個所に対応した「スーパー手法」との間で、最も好ましいバランスを取ることを意図して作られている。IDEF ファミリーの手法は、個々の手法を適用して得られた結果を統合する為の明確なメカニズムが提供されていることで、このバランスが維持管理されている。

恐らく、IDEF ファミリー手法に関して最も強調したい論点は、複雑性に対応する為に、分析や設計、開発を行う必要のあるシステムを、バラバラで、かつ運用管理可能な塊へと分割するという手法の必要性が、次第に増大しているということである。対象となる分析や設計、製造活動のより良い実践法についての知識を、強化可能な手法でなければならない。適切に設計された手法は、モデル作成者の注意を重要な決定へと集中させると共に、無関係な情報や不必要な複雑さを覆い隠すことで、非熟練者の能力レベルを熟練者に並ぶレベルにまで引き上げてくれるのである。

顧客にとっては、間取り図（floor plan）と完成想像図（artists rendering）も、最終計画図（final blueprint）と同等に重要なものである。その為、手法の開発者は、モデル作成者と顧客の双方のこうした需要を、個々の手法がどれだけ満足させているかについて、常に評価し続けなければならない。また手法の使用者（practitioner、実践者）は、手法を適切に選択、使用可能なように、手法の背後にある基本理論を十分に理解しておかなければならない。

IDEF 手法は元々、生産環境における大規模な統合を目的とした進化に伴って現れた複雑性への対応を目標としていたが、初期の未来像（visoin）を広げ、既存の境界線を越えた統合を実現しようとする、新しい試みへと引き継がれている。エンジニアリングや生産、それらの支援活動に跨る大規模な統合は、特に手法エンジニアにとっては、非常に面白く、かつ挑戦的なものである。手法エンジニアの今後の課題は、分析と設計、そして将来の統合環境を簡単に使用可能な形で実現することに関する経験についての、基本的理論と本体（？body）を要約（encapsulate、カプセル化？）することである（？）。

以上

（参考図書は省略）