事 例

オブジェクト指向モデリング手法「MELON」を用いたシステム開発

System Development using Object-Oriented Modeling Technique "MELON" by Fumio KONDO and Masao OKABE (Corporate Systems Department, Tokyo Electric Power Co.,Inc.).

近藤文夫[1] 岡部雅夫[1]

1 東京電力（株）システム企画部

1. はじめに

当社では，業務アプリケーションの開発にMELON(Multi-layered Epistemic Logical Object Network,多層型認識論理オブジェクトネットワーク)[1]を導入することを検討している．MELONはオブジェクト指向技術を用いた業務モデル化手法である．業務分析・要求仕様のまとめといういわゆる上流工程で用いる．しかし，導入する際には明確にしなければならない事柄がいくつかある．まず，モデル化の手順が不明確では，試行錯誤を繰り返すこととなり，手法の定着は難しい．また，下流工程で必要とする成果物を揃えることは当然だが，さらに下流工程で使いやすい形で提供することが重要である．この点は実際のシステム開発をする立場としては重要な課題である．今回はこの2つを明確にするために1996年4月から9月の約6カ月間に，コンピュータ設備のファイナンスリースおよび設備管理を行う当社関係会社のシステムへMELONの試験適用を行った．ここでは，その適用事例を紹介する．

まずMELONの概要を述べ，次にプロジェクトの実績とプロジェクトの取組み状況を述べる．それからMELONを用いた開発工程について提言する．

今後オブジェクト指向技術を用いた業務系システムの分析・設計を検討されている方々の参考になれば幸いである．

2. MELONについて

2.1 MELONの背景

当社では，業務をモデル化し整理するために，オブジェクト指向技術の適用を検討している．しかし，オブジェクト指向技術を通常のビジネス系業務の業務ドメインに適用しようとすると動的振舞いの記述が困難であることが多い．

たとえばリース管理業務をモデル化する場合に，リース品であるパソコンや利用者をオブジェクトとして捉えたとする．このとき，パソコンや利用者自体はどんな属性を持ち動的振舞いを起こすのか．純粋にパソコンを考えれば，キーボードCPU，ディスプレイなどにより構成され，各部品はメーカ，性能などの属性をもつ．動的振舞いは電源を入れたりコマンドを叩くことにより発生する．しかし，属性についてはよいのだが，動的振舞いについては，リース管理業務においては問題の対象とはいえない．パソコン自体の動的振舞いをモデル化することはほとんど意味をもたない．

それでは，動的振舞いとは何であろうか．あるいは動的振舞いをモデル化する必要はないのだろうか．従来からのE/Rモデルなどのデータモデルは，いわゆる静的構造のみであり，動的振舞いをもたない．データの共有という意味ではよいのだが，これだけでは業務を表現することはできない．ビジネス系業務には，その業務がもつ動的振舞いがある．つまり，前述の例を用いるとパソコンや利用者のような純粋なオブジェクトが動的振舞いをもつのではなく，両者を結びつけた関係が動的振舞いをもつ．そこで，両者の関係をオブジェクトとして捉えた方が自然である．この考え方がMELONの原点である．

2.2 MELONの概要

MELONでは，モデル化の対象領域を開発しようとしている情報システムと人間系を含めた業務システム全体として，「役割場協調モデル」，

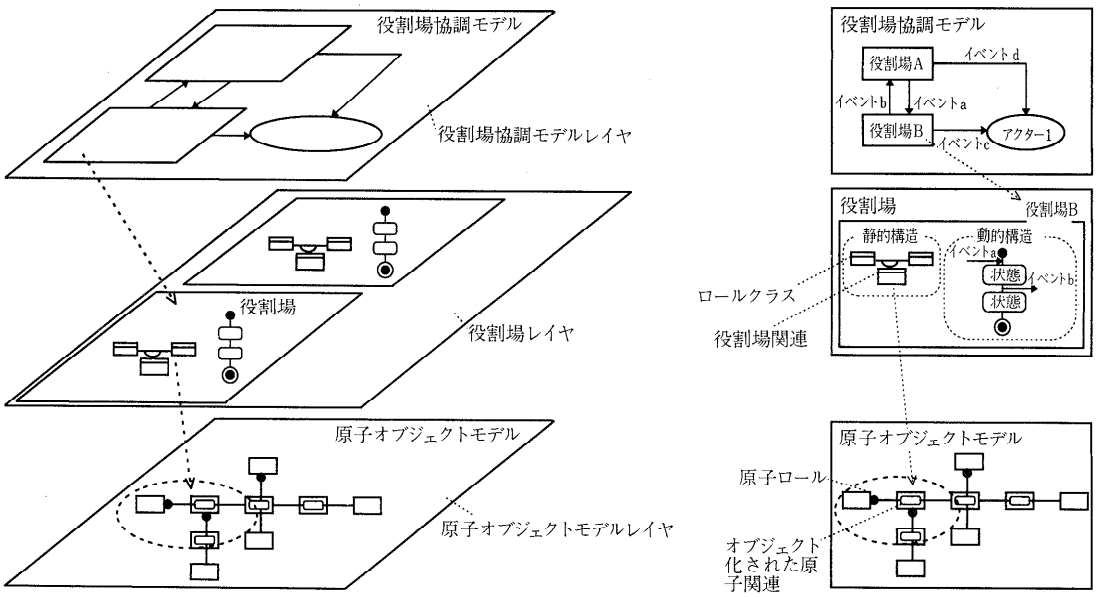

図-1 モデルの構造

「役割場」,「原子オブジェクトモデル」という3層構造のモデルで表現する(図-1参照).

まず,業務の観点に応じて役割場というオブジェクトを導入する.役割場はモデル化対象領域の内部に存在し,一般的には,ほかの役割場からのイベントにより起動するものである.したがって業務システムの全体像は,各役割場間のイベントのやりとりとして表現される.これが次に述べる役割場協調モデルである.また,役割場では,オブジェクトの静的構造と動的振舞いを定義する.静的構造は,後述する原子オブジェクトモデルという業務の観点に依存しないオブジェクトの関連を用いて定義する.動的振舞いは,役割場のインスタンスが生成してから消滅するまでを,状態遷移により定義する.役割場では,イベントを出力するまで,あるいはイベントを受けとった後の動的振舞いを定義する.つまり,役割場間の関係は定義しない.これについては役割場協調モデルで定義する.

役割場協調モデルでは,各役割場の入力イベントと出力イベントの関係を定義し,業務システムの全体像を表す.

動的振舞いは担っている役割により異なるにしても,ビジネス系業務において重要な情報の共有化という観点からは,業務の観点を排除したモデルが必要である.MELONでは,すべて対等な原子オブジェクトクラスとそれらの間の関連としてモデル化している.それを原子オブジェクトモデルと呼ぶ.

MELONの最大の特徴は,動的振舞いの単位としてのオブジェクトである「役割場」と,情報の共有化のための「原子オブジェクトモデル」とを有機的に対応づけていることにある.

3. MELONによるシステム開発事例

3.1 対象業務

リース契約,設備購買,設備管理,資産管理,保守契約,請求,収入管理というリース会社の社内業務部分を対象とした.具体的には,図-2の役割場協調モデルで示された範囲が開発した範囲である.

今回の対象業務はビジネス系業務であり,また対象業務がほぼ社内で完結している.MELONの検証という意味で妥当な業務規模である.

3.2 システム概要

現行システムではプログラム言語としてCOBOLを用いており,約80kstepである.

新システムでは,プログラム言語としてVisualBasic4.0,データベースとしてOracle7を用いた.ハードウェアは1台のデータベースサーバマシン,複数のクライアントマシンという環境を想定した.

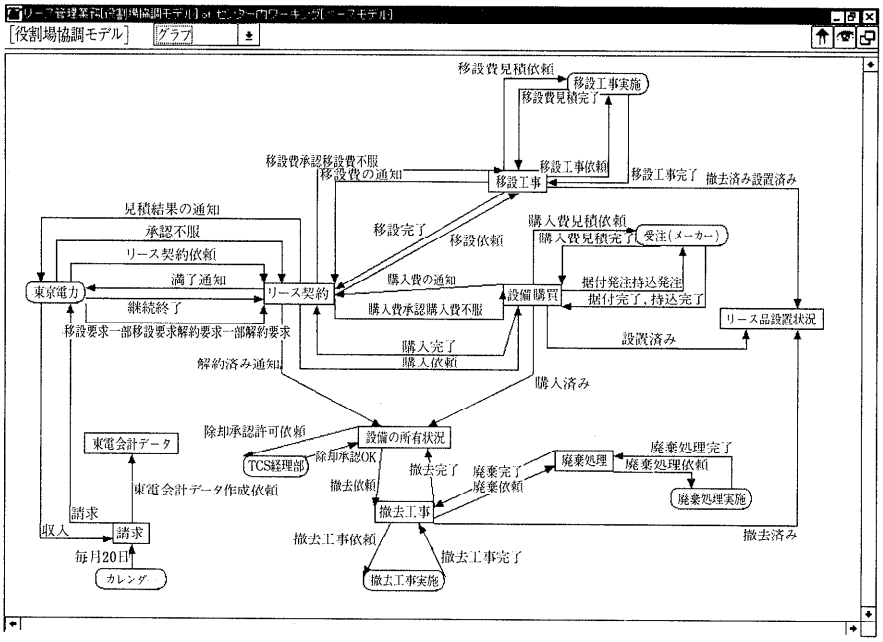

図-2 役割場協調モデル

3.3 開発体制

当プロジェクトはワーキンググループとして13名で取り組んだ．また，MELONに関する知識を有する者は内2名で行われた．全員がプロジェクト専属ではなく，作業時間はのべ900時間程度となった．

3.4 開発工程

MELONの検証という観点から，従来のウォータフォール型ではなく，手戻りを積極的に評価するという意味でスパイラル型で行った．また，期間が短いことから，作成ドキュメントを極力省略した．

(1)業務設計

モデルの作成手順としては，まず役割場協調モデルの案を各自作成し打合せ時に持ち寄りレビューを行い，全員の合意を得た役割場協調モデルを定義した．次に，役割場単位に担当者を決めて各自役割場を作成し，打合せ時に関係する役割場間の整合を図った．そして，役割場で定義した静的構造をまとめて原子オブジェクトモデルを作成した．それから，役割場協調モデルを調整し，役割場を調整するという作業を繰り返した．

a. 役割場協調モデル

最初に各自が作成した役割場協調モデルはセマンティックスの違いはあまりなかったが，表現方法はさまざまだった．現行システムのサブシステムが大体役割場の単位となった．イベントは，実業務における帳票や通知を用いることから始めた(図-2参照)．

b. 役割場

ここで苦心したところは，イベントとイベントの引数の定義である．役割場間には役割場協調モデルを通じて有機的なつながりがあり，イベントの定義のためには，関係する役割場の担当者間の調整が必要となり，臨機応変に検討の場をもつこととなった(**図-3**参照)．

c. 原子オブジェクトモデル

原子オブジェクトモデルについては，役割場で必要とする項目のみを定義すべきであるが，最初は現行システムで使われている項目を片っ端からあげてしまい，不要な項目が多々見受けられた．

(2)システム設計

基本的には業務設計で作成したモデルをそのまま使用することにした．細かい点については，モデルから実装へのノウハウの蓄積のために各自のアイデアに委ねた．その結果，作成したアプリケーション中にもその違いがみられるが，大きく分けて，2つの考え方がある．

a. プロセス中心

1つめはプロセス設計を中心にした考え方であ

る．1役割場を1プログラムとし，1状態を1画面とする．状態の遷移ではプログラムの内部変数の値を引き継いで，画面を切り替える．役割場間のイベントはデータベース経由で受け渡すこととする．つまり，出力イベントはデータベースにイベントの引数を格納することを意味し，入力イベントはデータベースからイベントの引数を取得することを意味する．受け手側での入力イベントの特定は，アプリケーションの画面に入力イベントをリスト表示してユーザが選択する形となる．イベントとその引数は業務を遂行する上での単位となるので，画面でイベントを選択するという行為は業務上違和感はない．この考え方の場合，テーブルには2種類現れた．1つは業務で必要とするデータを管理するテーブルで，もう1つはイベントを管理するテーブルである．イベントの方は前述の通りである．データのテーブルは役割場の静的構造を用いた．この考え方では，モデルからシステム設計がかなり機械的に行われた．テーブルの構造も従来のシステムとほぼ同じになった．しかし，今回は可能だったが，役割場が多数ある場合にテーブル間の整合性を設計時にチェックしきれるかいささか疑問が残った．

b. テーブル中心

2つめの考え方は，テーブル設計を中心とした考え方である．プロセス設計とテーブル設計は完全に切り離して考える．まず，テーブルについてはオブジェクト化された原子関連を用いて正規化されたテーブルを定義する．さらに，各状態を管理するために役割場のインスタンスを特定するキーと状態を格納するテーブルを定義する．そして，静的構造をもとに1役割場のもつ属性と状態を1つのビューとして定義する．このとき役割場のインスタンスはビューの1レコードである．そして状態のアップデートはイベントにより行う．イベントはすべてストアドプロシジャを用いた．イベント名をストアドプロシジャの名前，イベントの引数をストアドプロシジャの引数とする．こうすることによりデータのライフサイクルを含めたテーブル定義ができる．先ほど，「プロセス設計とテーブル設計を完全に切り離して考える」と述べたが，この考え方においてはここまでをテーブル設計と考える．プロセス設計については役割場の各状態で行われる操作を元にユーザインタフェースの設計を行う．そして，どこかのあるタイミングでストアドプロシジャを呼び出す形となる．ストアドプロシジャはイベントの受け手側，すなわち状態をアップデートされる側が設計を行った（図-4参照）．

(3) プログラミング

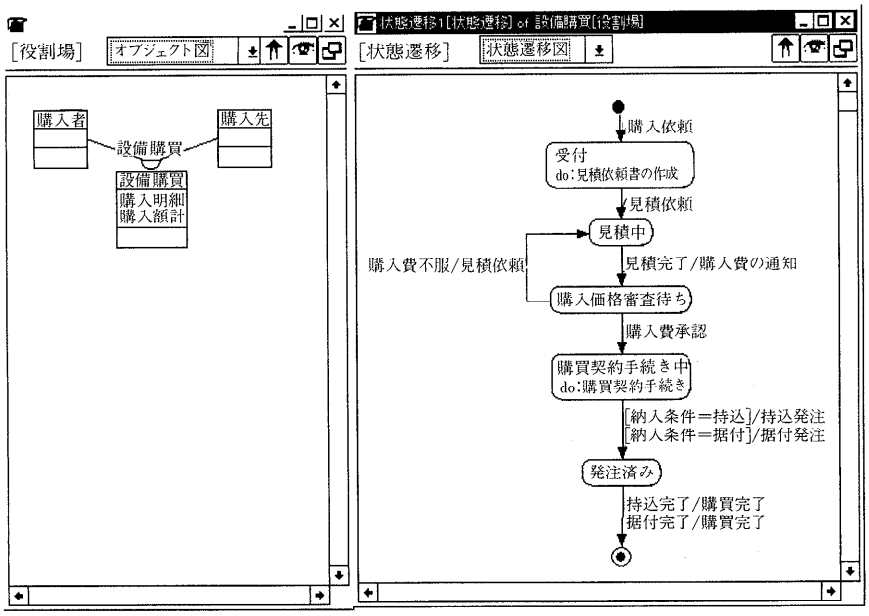

図-3 役割場

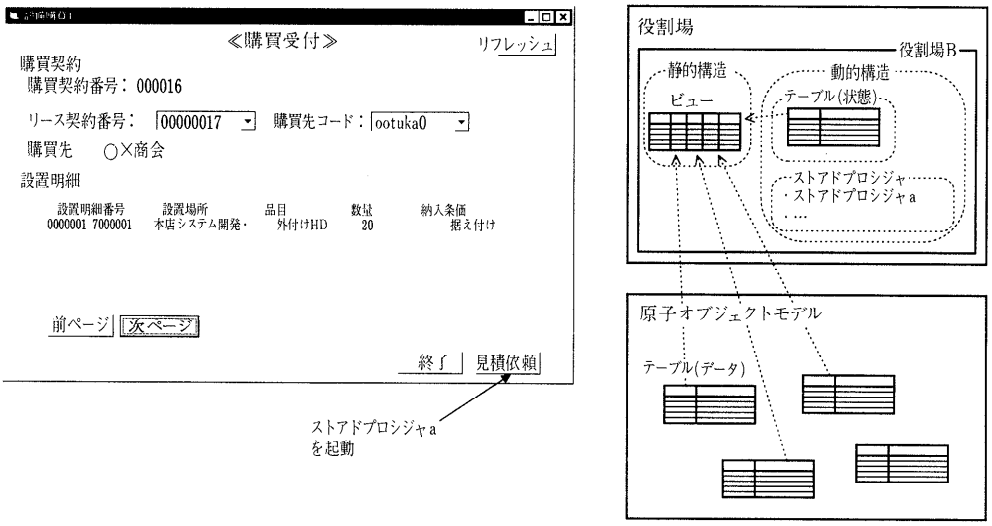

図-4 テーブル中心の考え方

ハードウェアのトラブル，またデータの不整合等に起因するプログラムの例外処理は簡略化し，エラーメッセージを出力して直ちに終了する形をとった．また今回，コーディングの標準化は行わなかった．

(4)テ ス ト

業務全体を通した整合性のテストを行った．手順としては，業務シナリオに基づき役割場協調モデルをウォークスルーして確認した．

3.5 適用についての総括

MELON の理解と業務の理解を同時進行で行ったために，それらを順番にこなす以上に時間をとられたように思う．最初のモデル作成まではかなり時間を費やしたが，システム設計以降は予想以上にスムーズに進んだ．

スムーズに進んだ理由としては，MELON のモデルがほとんどそのままシステム設計として利用できたことが大きいと考えられる．VisualBasic，Oracle を用いたプログラミングの経験者が 1 人もいなかったにもかかわらず，生産性もまずまずの実績を残せたのではないかと思う．今回に限らず，業務分析と設計で同じモデルを利用することにより，システム化の目的に対する設計の信頼性が向上し，実装における手戻りが減少するであろう．生産性を論じるのは早計であるが，結果として生産性は向上するのではないだろうか．

また，MELON で提唱している「業務の役割をオブジェクトとする」ことのメリットを実感できた．ここでいうオブジェクトは役割場であり，モデル化においては役割場をいかに切り出すかが課題なのだが，これは違和感なく切り出すことができた．業務の観点を排除したオブジェクトを中心にするモデルよりはモデル化のしやすさを実感できた．オブジェクト指向技術をビジネス系業務システムの分析・設計に導入する際の参考にしていただけるのではないだろうか．

4. 実例を踏まえた開発工程の検証

4.1 全 体

今回の適用を踏まえ，以降では MELON を用いた開発工程について提言する．

まず全体として，純粋な業務分析・設計については MELON でほぼカバーできる．しかし，システム開発を行うときには，ハードウェア，ソフトウェアの選定，開発に要する費用と期待される効果の見合いなど検討すべき事柄がいくつかあり，これらの作業はそのまま残る．

一方，MELON では業務の中心となるデータはデータ間の関係を含めて定義しており，下流工程でそのまま扱える情報となっているので下流工程の工数は若干減少するだろう．

4.2 業 務 設 計

今回の経験から推察すると，1 役割場の大きさは COBOL 換算で 10kstep くらいまでが妥当である．それを超える場合には役割場の切り出し方

を見直した方がよい．というのは，設計者が全体を見きれなくなってしまい作業効率が低下する．また，モデルの不整合が生じる確率も高くなると予想される．役割場協調モデルについては役割場の数で10個くらいまでがよいのではないだろうか．それを超える場合には，役割場協調モデルを分割して作成してから統合すべきである．

開発体制としては，今回の業務規模程度であれば役割場協調モデルと原子オブジェクトモデルは1～2名，役割場は1役割場あたり1名がよい．そして，少なくとも1名はMELONを熟知した人員が必要である．そしてその人員は原子オブジェクトモデルの作成に加わることが望ましい．また，モデル化の過程で生じる業務の目的，手順に対する疑問について逐次確認できる環境が望ましい．これはMELONを適用する場合に限った話ではないが，業務に精通した人員がモデル作成に加わるか，あるいはユーザ部門の参画が望ましい．現行業務のマニュアルからある程度詳細までモデル化できるが，システムの移行を考えるとユーザの現行業務に対する認識を理解することは重要である．

次に，工程内での作業手順について簡単に述べる．大きくは以下の5つのステップに分けられる．

ステップ1：役割場協調モデルの概略定義
ステップ2：役割場の静的構造・原子オブジェクトモデルの骨格定義
ステップ3：各役割場の動的振舞いの概略定義
ステップ4：各役割場の詳細定義
ステップ5：役割場協調モデルと役割場の関連づけ

MELONでは，役割場協調モデル，役割場，原子オブジェクトモデルという3つの層が有機的に関連しているので，それぞれの層を1つずつ完成させる手順には無理がある．また，当然ながらそれらを完全に並行して作成するのも難しい．そこで，前述のステップ1～5のように，3つのモデルを少しずつ作っていく．

言い換えれば，MELONでは，業務の理解の過程そのものが，モデル化の過程である．業務の理解の過程は，通常，ある程度理解して，そこから，不十分な点が洗い出され，それをもとに，さらに理解を深めていくといった過程を繰り返すものである．

各モデルを作る際の注意点を以下に述べる．

(1)役割場協調モデル

無用な議論をさけるために，ノーテーションの会得，そしてネーミングルールなどの標準化の適用をすすめる必要がある．

役割場は現行システムのサブシステムを用いることから始めるのが，担当者間の意識のずれが少なくてよい．イベントは，実業務における帳票や通知を用いることから始めるとよい．もちろん最終的には帳票や通知とは異なるべきであることをつけ加えておく．

(2)役 割 場

イベントの定義のためには，関係する役割場の担当者間の調整が必要であり，臨機応変に検討の場をもつ必要がある．

(3)原子オブジェクトモデル

原子オブジェクトモデルについては，役割場協調モデルを業務シナリオに基づいてウォークスルーをし，必要となる項目のみを定義すべきである．

4.3 システム設計

開発体制については従来となんら変わらない．システム開発工程の成果物も従来とそれほど変わるものではない．しかし，設計へのインプット情報はMELONで作成したモデルであり，手順は当然ながらMELONに依存する．今回の適用をもとに，モデルの活用方法について述べる．

(1)インタフェースの切り出し

役割場協調モデルから，情報システムと人間系の切り分けを行う．詳細なモデルが定義されているなら，アクタを人間系，役割場を情報システムと考えればよい．

(2)テーブル設計とプロセス設計

3.4節の(2)で述べたテーブル中心の考え方を用いるべきである．こうすることにより，ユーザインタフェースを自由に設計できる．また，役割場単位の作業の並行度が高くなる．後々のシステムの拡張性に富んでいる．かつ保守性に優れている．

たとえば，帳票の設計段階で，モデルに未定義のデータ項目を帳票に表示させたいという要求が発生した場合には，必要に応じてそれらの項目のテーブルを追加作成することになる．しかし，それらの項目は該当業務においては従属な項目であ

り，単に追加すればよく，追加することによるほかのテーブルへの影響はほとんどないはずである．万一，従属なデータでなければ，業務モデル自体に修正の必要がある．言い換えると業務自体の仕様が変更になっている．この場合には，どんな手法であれ大きな手戻りが生じるのは当然であり，手法の適用により回避することはできないであろう．

ところで，メリットばかりとはいいきれない．原子オブジェクトモデルのデータベース構造へのマッピング次第では，RDB を用いた場合には実行性能に差が出るため，正規化するレベルを考慮すべきである．

4.4 プログラミング以降

開発体制については，従来となんら変わらない．

作業手順についても従来と変わらない．システム設計を受けてプログラム設計を行い，コーディング，テストと順に行う．

5. おわりに

今後の課題としては，生産性の明確化があげられる．これについては，大規模システムへの適用実績を積み重ねて検討するしかない．

また，MELON をサポートするツール LEMON(Logical Environment for Multi-layerd Object Network，多層型オブジェクトネットワーク向き論理環境)を今回用いた．LEMON は原子オブジェクトモデルを除いて，図-2，図-3 のようなグラフィック表現をサポートし，グラフィックエディタを備えている．原子オブジェクトモデルについては，その性格上グラフィック表現はかなり複雑になり実質的に意味をなさないため，リスト形式の表示とエディタで対応している．今後は，生産性向上のためにも，LEMON に改良を加えて MELON のノウハウまでもサポートすることを検討中である．

謝辞 本稿は，著者と次の 11 名による半年間の研究成果である．また，多忙の中 MELON の試験適用を快く引き受け，ヒアリングに協力していただいた東京計算サービス(株)をはじめ，試験適用に協力いただいた東電ソフトウェア(株)，そしてこのような場を与えてくださった当社システム開発センター，当社コンピュータセンター，当社システム技術課の各位に感謝の意を表します．

東電ソフトウェア(株)堀禎威，吉澤賢一，当社システム開発センター阿孫親悟，茨木久美，小熊康弘，片柳俊雄，熊倉宏，沼田孝一，吉川善樹，渡辺香里，和田幸子 (順不同，敬称略)

参考文献

1)岡部，小熊，渡辺，羽生田，皆川，佐藤：オブジェクト指向モデリング手法「MELON」－ビジネスドメインに特化した手法，オブジェクト指向最前線情報処理学会 OO'96 シンポジウム，pp.1-7，朝倉書店(1996).

2)Rumbagh J.et al.:Object-Oriented Modeling and Design, Prentice Hall, New Jersey (1991).(羽生田栄一監訳:オブジェクト指向方法論 OMT，トッパン(1992).)

(平成 9 年 4 月 7 日受付)

近藤 文夫

1989 年中央大学理工学部数学科卒業．同年，東京電力（株）入社．現在，同社システム企画部システム開発センター．業務のモデル化による業務革新に興味をもつ．

岡部 雅夫（正会員）

1979 年東京大学理学部数学科卒業．同年，東京電力（株）入社．1987 年スタンフォード大大学院・文理・統計学科，工・OR 学科修士課程修了．現在，同社システム企画部システム開発センター．情報処理学会情報規格調査会 SC21/WG3/CSMF SG 委員．日本規格協会情報技術標準化研究センター情報資源スキーマ・業務モデル標準化委員会委員．

用語解説

オブジェクト指向 object-oriented

ソフトウェアをデータ構造と振舞いを両方含むオブジェクトの集まりとして組織化するソフトウェア開発戦略．

オブジェクト object

問題においてはっきりした意味や境界をもつ概念，抽象，もの．クラスのインスタンス．

モデル model

あるものを構築する前にそれを理解する目的で抽象

したもの.

動的振舞い dynamic behavior

時間に依存した振舞い.

属性 attribute

あるクラスの各オブジェクトのデータ値を記述するための，そのクラスの名前つき性質.

E/R モデル entity-relationship model

実体とそれらの間の関連による構造記述.

イベント(事象) event

時間軸上のある点で発生するできごとのことで，瞬間的に起こり継続時間をもたない．あるオブジェクトから別のオブジェクトへの刺激となる.

関連 association

複数個のクラスのインスタンス間での関係であり，共通の構造と意味論をもつリンクの集合を記述する．(リンクとは，関連のインスタンスのこと．オブジェクト間の物理的あるいは概念的な結合．)

インスタンス instance

クラスにより記述されるオブジェクト.

状態 state

ある特定の時点におけるオブジェクトの属性およびリンクの値.

遷移 transition

事象によって引き起こされる状態の変化.

クラス class

似た性質，共通の振舞い，共通の関係，共通の意味論を持つオブジェクトの集団に対する記述.

ウォータフォール型 Waterfall modeling

滝が流れ落ちるように，上流工程のドキュメントを作成しレビューし，それを基につぎの工程のドキュメントを書いていき，最後にプログラムに至る開発手法.

スパイラル型 Spiral modeling

設計フェーズにおいて設計モデルの作成，妥当性の確認，モデルの修正，再確認といった作業サイクルを繰り返しながら開発していく手法.

正規化 normalization

関係データベースの表の更新に際して，一貫性の問題を減らすための規則.

ビュー view

関係データベースにおいていくつかの基本的な表から派生する仮想的な表.

ノーテーション notation

表記法．表現するための記号の集合およびその使い方の規則.

RDB relational database

関係(型)データベース．関係 DBMS が管理するデータベース.

ロール role

関連の一方の端.