平成１５年度

ビジネスオブジェクト検討プロジェクト

「ビジネスオブジェクト研究報告書」

～これからのＩＴ投資を考えるユーザのために～

平成１６年４月

社団法人　日本情報システム・ユーザー協会

目次

1 はじめに

ビジネスオブジェクト研究プロジェクトは、本年度で２年目となる。初年度はオブジェクト指向技術の全体像をつかむ為の基礎研究とオブジェクト指向に積極的な取り組みをおこなっているオブジェクト指向先進企業の調査が主な活動内容であった。

本年度は、昨年度の活動を礎としてオブジェクト指向の応用技術の研究と、IT先進国といわれている韓国の訪問を行った。さらに、昨年と同様に最新の技術動向に関して調査を継続した。

プロジェクトのメンバも多くの人が昨年より引き続き参加してくれ、また有能な人材にも今年度新たに参画して頂いた。おかげで有意義な報告書が出来上がったと自負している。

よく知られているように、オブジェクト指向技術はプログラム言語から発達している。これまでオブジェクト指向技術は、コンピュータ上でのパフォーマンスが悪く実用に適さなかったが、インターネットの普及による分散技術の発展と、コンピュータの高性能化、低価格化によって、これまでの古い言語を凌駕しつつある。

さらにプログラム言語への適用を超えて、システムの分析と設計への適用が盛んに行われるようになり、今では企業のプロセス分析にまで適用されようとしている。

その具体的な技術例として、モデル駆動型のMDAやコンポーネント技術を駆使したCBDなどが盛んに研究され、あるいは実際のビジネスに適用されつつある。

このような時代にあって、ユーザ企業もベンダーも、このオブジェクト指向技術から無縁でいられるわけはないであろう。

わが国の産業は、1970年代と80年代のIT投資によって高度成長が実現できたことは諸外国が高く評価する事実である。しかしながら1990年代は、10年以上に及ぶ景気の低迷から、ほとんどの企業が大きなIT投資を控えてきた。このことは、新技術に対する研究投資にも及んでいる。米国では日本とは逆に1990年代にITへの積極的な投資を行い、見事なまでの産業構造の変革を実現している。オブジェクト指向技術もこの時代の寵児と言える。

今年、わが国はようやく長い不景気のトンネルを抜けようとしている。何よりも新しい市場に新しいサービスが求められ、それを支える新しいシステムが必要となる。オブジェクト指向技術が一気に開花する時代がすぐ目前にある。「守・破・離」の教えによれば、本年度は「破」の年であった。ユーザの立場から考えるオブジェクト指向に一歩でも近づいたと、この報告書を読んで感じてもらえれば幸甚である。

ビジネスオブジェクト研究プロジェクト
委員長　福　田　修

2 クラスの切り出しの標準化

工学的なアプローチとは、同じ方法で何かを作れば、誰がやっても同じ機能で同じ品質のものができることを言う。オブジェクト指向技術は、これまでのソフトウェア工学が培った技術の綜合に位置している。しかしながら、現在のところ世に広まっているオブジェクト指向技術を使ってクラス設計を行うと、誰がやっても違うクラス図ができあがってしまう。これは一体どうしたことだろうか。

2.1　分析者の視点が違う

物理的には同じもの、すなわちオブジェクトを観察していても、観察の視点・座標が異なれば違ったものに見える。例えば、円錐形の物体があって、これをぶら下げ、横から光を当ててその影を見れば三角形に見える。また、上から光を当てて底に映った影を見れば円形に見える。

客観的にひとつの物体を観察しても、観察の視点によっては違うものに見えてしまう。この問題を解決するためには、幾つかの方法がある。

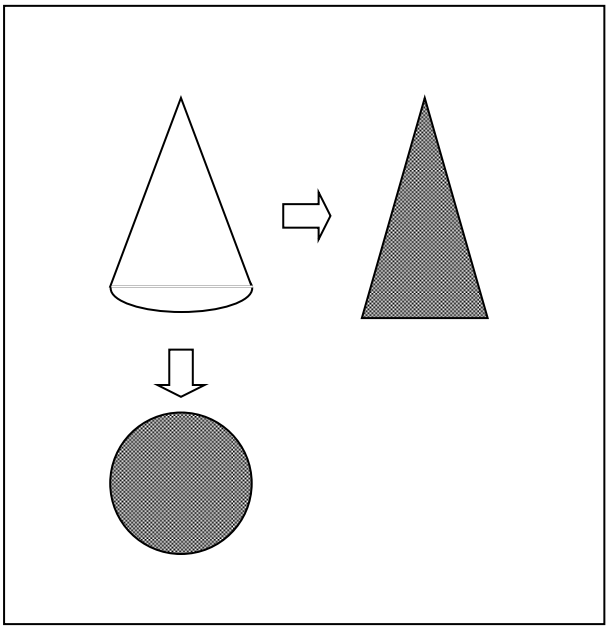


図 2.1-1

ひとつには、視点を交換する方法である。三角に見える視点と円に見える視点が発生したら、それぞれの観察点を交換すれば、三角から円に、円から三角に、物体の形状が変わって見える。そこから、二つの観察データを比較すれば、そのように見える物体とはどのような形状をしているのかが発見できることになる。

オブジェクト指向分析において、この問題は、ユーザの立場から見た視点とシステムの立場から見た視点の違いに相当する。業務の分析を行う場合、ユーザの立場をとれば、その業務はどのような手順で実施されるのかと言った視点を持ち易いし、システムの立場をとれば、その業務はどのような機能を持っているのかといった視点を持ちやすい。例えば、物やサービスを売るという営業業務についてこの視点の問題を考えてみる。

企業には概ね営業担当者がいる。これが経営者そのものでも良いし、特定の営業員でも構わない。物やサービスを売って収益を得る機能を持った企業内の存在を営業担当者とする。

営業担当者は、電話なり文章なり面会なりで引き合いを得る。これを「案件」と呼ぶことにする。案件が発生したら営業担当者は案件リストを作成する。案件台帳を作る目的は、案件の発生日時や内容によって、商談成立の確度を高めるためと行動予定を明確にするためである。この案件にかかわる商談は、「商談メモ」として記録され案件に紐付けされる。このような作業によって、営業担当者は案件台帳をみれば、案件の進捗状況や、商談成立の確度を把握することができ、営業成績の向上を図ることができる。また、企業内でこの情報を共有することで、売上見通しや営業戦略の策定ができるようになる。

まずこれだけの要件でシステム担当者がクラス図を書くと次のようなものになることが多い。(図 2.1-2)

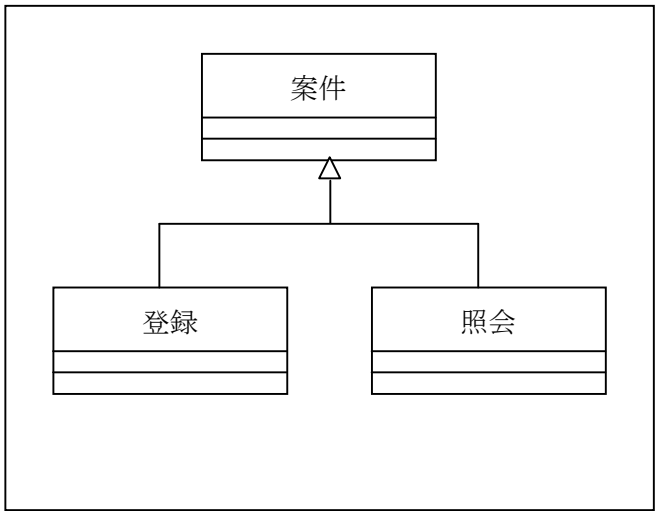


図 2.1-2

これは「案件登録」という機能と「案件照会」という機能に視点を置き、両機能は同じ「案件」という属性をもっているので、汎化の関係を使って案件クラスを導き出したと考えられる。

つまり、システム担当者は、営業担当者がシステムに案件と登録し、登録された情報を照会するという「機能」に視点を置いて、クラスの関係を導き出しているのである。

システム担当者がこのように「機能」を中心とした視点を持つ限り、商談についても同様のクラス図を考えるであろう。(図 2.1-3)

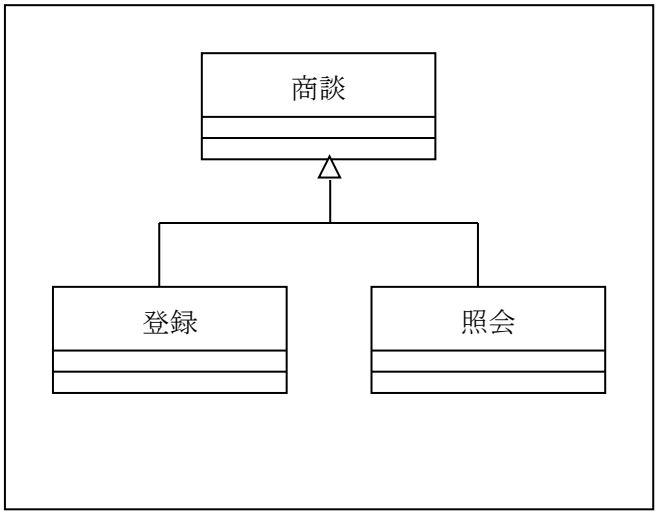


図 2.1-3

次には、案件クラスと商談クラスの関係から、「営業」というクラスを作り出し、図2.1-4のようなクラス図を作ることになる。

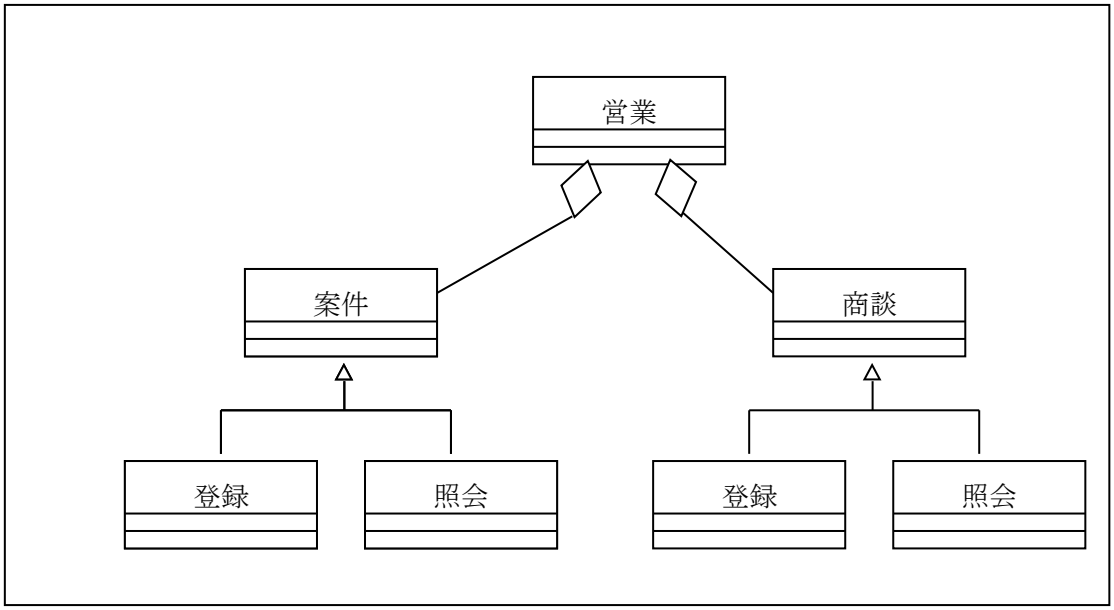


図 2.1-4

一方、ユーザの視点を持っている場合には、最初に考えるクラスは何になるのか。おそらく、クラスとはオブジェクトを抽象化したものであり、オブジェクトとは現実世界であなたが業務上関係している人や会社や伝票などであると条件を与えられれば、最初に考えるのは「顧客」になるだろう。この顧客が案件を出し、案件から商談が生まれると考える。

また、集約の概念が分かっていれば、ひとつの案件に複数(数回)の商談が発生する可能性があるので、商談と案件が集約によって結ばれるようにするだろう。こうして書かれたクラス図(図 2.1-5)は、現実の世界に近いモデルとなっている。この後、残りの「営業担当」や「営業計画」などを付加してゆけば良い。

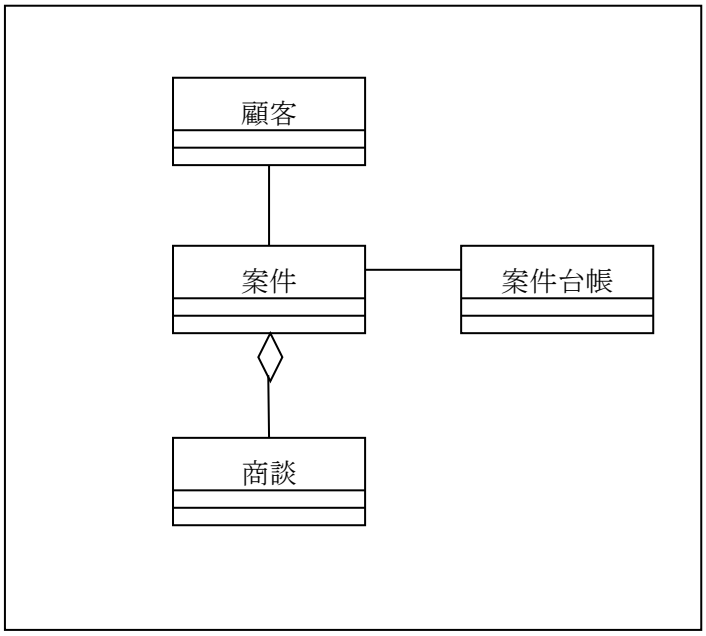


図 2.1-5

さて、機能に視点を置いた図 2.1-4 と業務に視点を置いた図 2.1-5 のクラス図としての優劣を検討しなければならない。なぜならば、これまでにソフトウェアの世界で使われてきた構造化設計に慣れ親しんだソフトウェア技術者には、図4が何故クラス図として悪いのかが理解しづらいからである。

実は、機能中心の図 2.1-4 と業務中心の図 2.1-5 の優劣は、オブジェクト指向技術である前提がなければ優劣ははっきりしている。機能中心であれば図 2.1-4 が正しく、図5は意味をなさない。あくまでもオブジェクト指向技術を前提としているから図 2.1-5 が良いクラス図になる。

オブジェクト指向とはオブジェクトという単位で実世界を捉えていく方法である。システムをオブジェクトの集合と考える。このオブジェクト間のメッセージ送信によってシステムの機能が実現されると言う考え方をとるシステム開発方法論である。

何故オブジェクト指向が注目されるようになったかについては、昨年度の当委員会報告書に詳しく報告しているのでここでは割愛するが、システム開発をコンピュータ寄りから人間寄りにしていることがポイントである。

従来のデータ中心設計手法では、機能分析とデータ分析がまったく別の作業になってしまっており、設計時点での統合が困難になったり、この困難が実装に持ち越されたりしていた。これに対してオブジェクト指向開発では、要求・分析・設計・実装と、一貫して「オブジェクト」という概念に基づいて開発プロセスが進められるため、分析から設計へのギャップ、設計から実装へのギャップが小さくなり、シームレスに開発を行うことができるようになる。

要求・分析・設計・実装という各開発局面において、一貫してオブジェクトの集まりとしてシステムを捕える事のできるメリットは、トレーサビリティの向上に繋がる。すなわち、

◆　上流工程での要求項目や修正要求などが下流に与える影響範囲がわかりやすい。

◆　下流での修正が上流工程のどの範囲に影響を与えるかがわかりやすい。

といったメリットが生まれる。これは、従来手法に比べて仕様変更の影響度や影響範囲が明確になる事を意味している。

また、オブジェクト指向技術ではオブジェクトの責務を明確にするので、機能追加や仕様変更時にどのようにオブジェクトを追加・修正すればよいかが明確になる。

したがって、人間の実社会を素直にモデリングし、人間側の考え方でシステムを設計開発し、オブジェクトの責務を明確にさせるといった考え方をとれば、図 2.1-4 と図 2.1-5 では明らかに図 2.1-5 のほうがクラス図としては良いと判断できるのである。

また、図 2.1-4 の場合には、「案件登録」としたクラスは、案件登録の機能を述べているのであって、案件に関する責務を述べているのではない。

オブジェクト指向技術では、機能はオブジェクト同士の相互作用によって実現されるのもである。

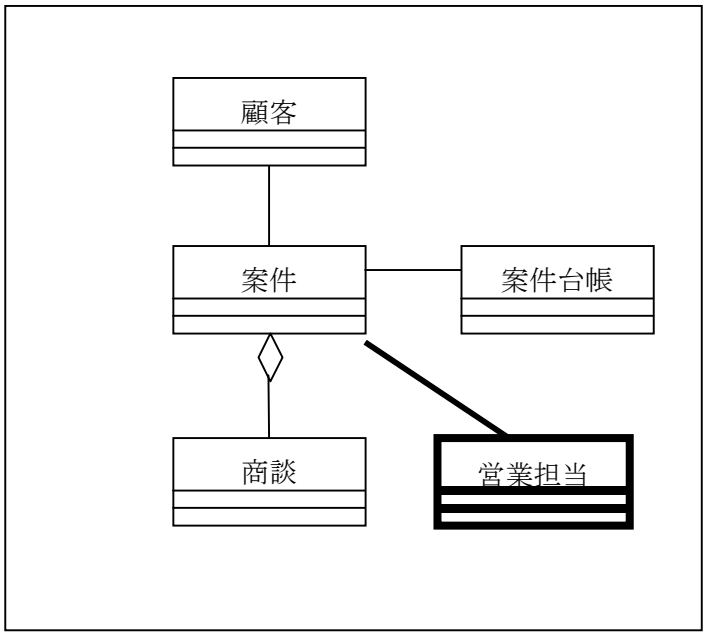


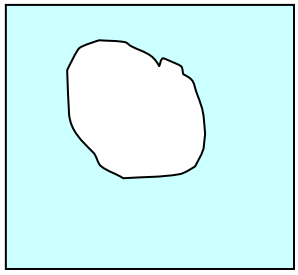
図 2.1-6

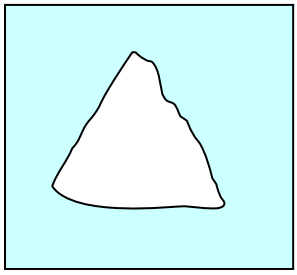
図 2.1-6 は、図 2.1-5 に営業担当のクラスを追加したものだが、案件を登録すると言う機能は、案件オブジェクトと営業担当オブジェクトとの責務の関係において営業担当の責務として実行される。（オブジェクトとはクラスを実体化したものである）

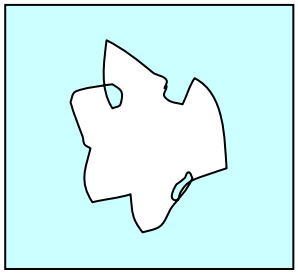
誰が書いても違うクラス図になってしまう原因のひとつが、観察者の視点（座標）がことなっていることにあることをこれまで見てきた。

2.2　指示と道具について

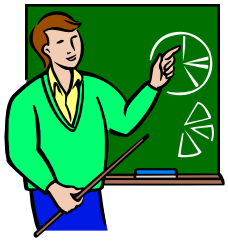
分かりやすい例として、小学校での授業を想定してみる。先生が数人の生徒に白い紙を渡し「絵を描いてごらん」と指示したとする。生徒たちはそれぞれに異なった絵を描くだろう。





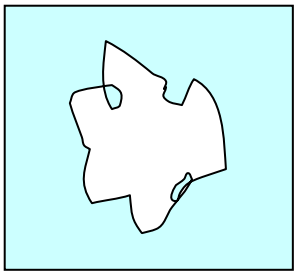


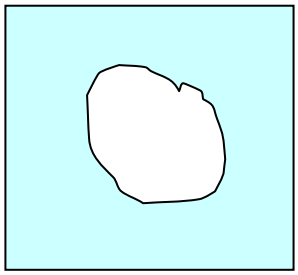
この先生の生徒に対する指示を、次のように変化させたとする。

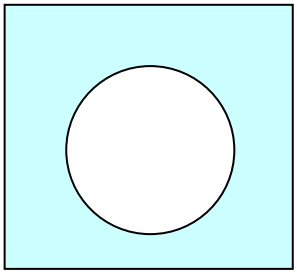


(1)「絵をかいてごらん」
(2)「円を描いてごらん」
(3)「真円(まんまる)を描いてごらん」
(4)「みんな同じ大きさの真円を書いてごらん」

おそらく、それぞれ次のように生徒たちは描くだろう。


(1)


(2)


(3)

では、(4)に対してはどうするだろう。学年によって異なるだろうが、幼年ではお手上げの状態になるに違いない。ここには、コンパスなり、型(テンプレート)という道具がなければ、この指示には対応できない。

オブジェクト指向技術でクラスの設計を行う場合、現状では(3)までの状態である。真円を

描く道具がUMLであり、デザインパターンやフレームワークに相当する。アナリシスパターン、アーキテクチャパターンなどと呼ばれているバターンの利用が(4)を狙った方法論であるが、システムの分析や設計をすべてパターンで実現することは、今のところ不可能である。どうしても分析者や設計者の技術力が必要になる。あるいは想像力が必要となると言っても良いかもしれない。そのためには、訓練と実践の繰り返しが重要で、さらに、良いクラス分析、クラス設計をたくさん参考にすることも必要になる。

オブジェクト指向技術を学習しようとする場合、最初に躓く場所は、オブジェクトやクラスの発見である。自分が抽出したクラスとかオブジェクトに自信がない、あるいは教科書にある正解を見て、自分のモデルと正解との違いに愕然とする。ここから、オブジェクト指向技術は難しいとか、自分はオブジェクト指向には向いてないという考えが出てくる。

しかし、モデリングに 100 点満点の正解がない以上、より 100 点に近い状態に持ってゆこうとする努力と、その結果としての成果を求める姿勢こそが必要なのである。具体的には、モデルが実態を正確に表現していること、クラスの関係が明確に説明できること、つまり辻褄があっていること、他人が理解できることなどである。

では、再度先生が生徒に指示をだす例を見てみよう。

(1)「絵をかいてごらん」
(2)「円を描いてごらん」
(3)「真円(まんまる)を描いてごらん」
(4)「みんな同じ大きさの真円を書いてごらん」

これらの指示は、システム開発においては要件定義に相当する。(1)と(2)は制約条件、あるいは成果に対する期待がゆるく、作る側には楽な作業となる。しかし、真円の条件が加わると困難さは数段高くなる。正確に円を書くためには、道具を思いつかなければならないからである。さらに(4)では、道具に加えて同じ大きさを決める基準が必要となり、道具も共有されなければならない。誰が描いても同じ円が書けるという工学的なアプローチとなっている。

誰が作っても似たようなものになるための技術と標準化はオブジェクト指向技術の重要な課題となっている。オブジェクト指向技術には分析・設計者の創造活動が必要であることを述べたが、逆にそれ以外の部分は限りなく標準化できるし、しなければならない。そのためには、要求レベルを高くしておくことが必要である。

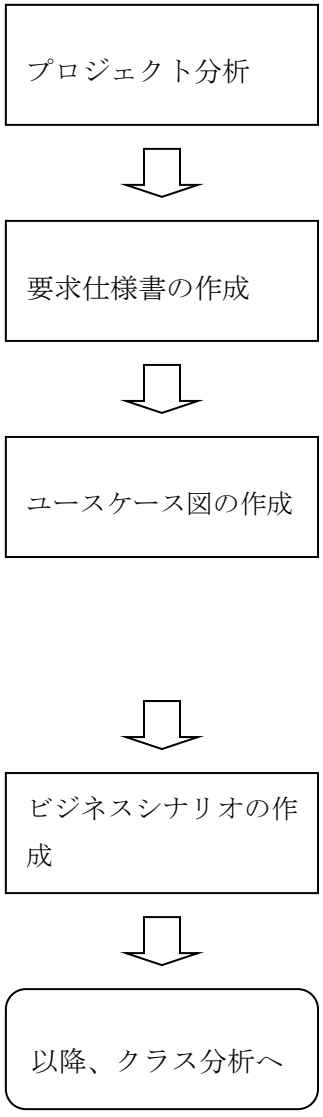
「技術者は高い要求が出されるとそれに応えようと最大の努力をするし、要求が低ければそれなりのものしか作らない」と喝破したのは、G. M. ワインバーグである。(要求仕様の探検学)

オブジェクト指向技術に取り組む場合にも、常に同じ物ができるようにするためには、どうしたらよいのかを考えながら設計分析を行いたいものである。

2.3　ビジネスシナリオの書き方

開発対象となるシステムに関するクラスを抽出するには、事前に幾つかの作業が必要になる。何も材料を持たずに、クラス図を書き始めることが不可能ではないが、第三者が理解できることと、結果に至るまでの考えたプロセスを明確にしておくことが、より正確にクラス設計を行うコツになる。

これらの大まかな流れは、次のようになる。



現状の問題点や顧客要求、および市場要求について分析を行う。さらにプロジェクトを遂行する上で、リスクとして考えられる項目を分析する。

プロジェクトの分析結果から、要求仕様書を作成する。この要求仕様書は、システム開発の原点にあたる文書となる。さらに、提案要求書（RFP）にも用いられる。

システムの境界を明確にした後、アクターの識別を行う。アクターとは、当該プロジェクトが開発するシステムに関連した人物やシステムである。日本語で「登場人物」と呼ぶこともある。アクターの識別が完了すれば、ユースケースの識別を行う。ユースケースとは、アクターがシステムに希望することを述べたものである。

ユースケースとは、業務シナリオのことである。ユースケースは、

- どのようなアクターが
- どのような場面において
- どのような目的を達成するために
- どのような活動を行うのか

ということを記述したもので、ユーザの視点から記述される。

ユースケースは、ユースケースドキュメントとか業務シナリオ、あるいはビジネスシナリオなどと呼ばれるが、「ユースケース」とすると「ユースケース図（ユースケースチャート）」と混同しやすいので、ここでは「ビジネスシナリオ」と呼ぶことにする。かつ、ユースケース図とビジネスシナリオとの二つを総称して「ユースケース」呼ぶことにする。

2.3.1 プロジェクト分析と要求仕様

統一プロセス(UP:Unified Process)では、最初の「方向付け」局面においてプロジェクトの概要とプロジェクトに影響を及ぼす市場要因、プロジェクトのリスク要因、および仮定事項を定義することになっている。

ここではビジネスシナリオの書き方に焦点を当てるので、プロジェクト分析と要求仕様はすでに完了している前提とする。要求仕様の概要は次のとおりである。尚、文中の「BBショップ」は、当委員会で想定した仮の会社名である。

A．要求仕様

（1） ＢＢショップは通信販売を主な事業としている会社である。この会社は、複数の仕入先から購入した商品をカタログ化して顧客へ配布し、商品の注文に応じて販売する。今回のプロジェクトでは、ＢＢショップの注文管理システムを開発する。

（2） ＢＢショップでは 3 ヶ月ごとに商品カタログを作成し、顧客および見込み顧客（商品に関心のある人々）に郵送する。

（3） 商品の注文は、電話、ＦＡＸ，電子メール、ＢＢショップのＷｅｂサイトから行う。

（4） 開発予定の注文管理システムは、注文発生から発送完了までの注文状況を管理する。

（5） 国内への商品発送は、離島を除き 1 週間以内に商品が注文者へ届けられる。海外へはこの保証を行わない。

（6） 返品は商品到着から 10 日以内とし、送料は顧客負担とする。

（7） 今回の開発はオブジェクト指向で行う。文書はＵＭＬ２．０を採用する。

（8） 支払いは、クレジットカード、代引（代金引換取引）、銀行振込、電信振替とする。

B．仮定事項

（1） 郵送や運送会社は複数利用することが予想される。

C．リスク要因

（1） システム障害発生時の注文情報の紛失が懸念される。

（2） ＧＵＩはＩＴに弱い人でも使える必要がある。

（3） 大量の注文データが発生した場合の対策をどうするか。

（4） オブジェクト指向を理解した技術者が少ない。

D．市場要因

（1） 他の通販会社は、商品の到着期間を 2 週間としている。

（2） 共稼ぎの家庭が多く、通販の人気は年々高くなっている。

（3） 各通販会社は、独自の商品戦略を持っており、選ばれた商品がそれぞれの通販会社を特徴付

2.3.2 ユースケース図を作成する

ユースケースは、開発対象となるシステムの全体像を明確化にする。これにより、システム全体の概要を把握することができるようになる。また、ユースケースを書くことでシステム開発に対する一貫性を与える。

作成したユースケース全てに対して、システムの顧客と開発者が合意することで、ユースケースはシステムの仕様書になる。

さらに、ユースケースは次のように開発全ての基準ともなる。

- クラス図は、ユースケースに表されている情報を表現しているか
- シーケンス図は、ユースケースの通りに進むか
- 完成したシステムは、ユースケースの通りに動くか

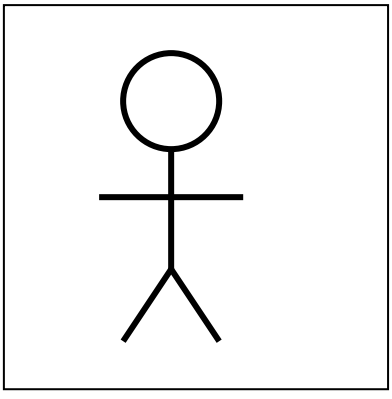
このように、ユースケースはシステム開発の全ての局面において利用され。こうしたシステム開発を、「ユースケースドリブン」の開発と言う。

2.3.2.1　　アクターの識別を行う。

アクターは、開発対象のシステムに関わる人間や組織、会社、システム、データストアおよび機器などのハードウェアである。アクターは、何らかの役割を持つ。また、要件定義書に表現されていないアクターもあり、非圧用に応じて新たなアクターを作り出す事もある。アクターを識別するヒントとして次のようなものがある。

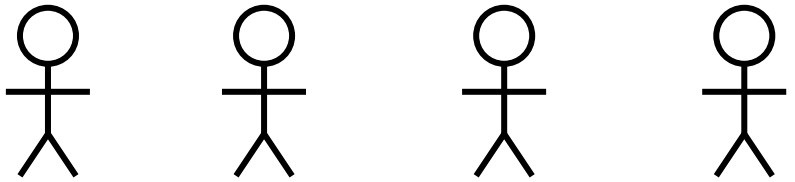
- 誰がシステムを使うのか
- 誰がシステムをインストール、起動、停止、保守を行うのか
- このシステムを使う他のシステムは何か
- 誰がこのシステムから情報を取得するのか
- 誰がシステムに情報を提供するのか
- 何が、あらかじめ設定された日時に自動的に行われるか

アクターは、下図のような記号で表される。



※この人物を模った記号を「スティックマン」と言う。また開発の現場では、このスティックマンに対して、愛情を込めて「呪いの藁（わら）人形」とも呼んでいる。

今回の要件仕様において、先に述べた条件に合致するアクターは、「顧客」と「運送会社」だが、BBショップ内での注文管理に関わるアクターも必要となる。これは、「顧客窓口」と「発送係」の二つである。



顧客　　運送会社　　顧客窓口　　発送係

さらに分析を進めてゆけば、在庫管理と会計のシステムのアクターとして作り出さなければならないだろう。

アクターが決まれば、それぞれのアクターの役割を定義する。

アクター	役割
顧客	BBショップに商品を注文する。
顧客窓口	BBショップの従業員で、顧客の注文に関する以下の処理を行う。 ・ 注文を受け付ける ・ 注文の状況を調べる。 ・ 商品カタログを送付する。 ・ 注文のキャンセルを受け付ける。 ・ 返品を受け付ける。 ・ 苦情処理を行う。
発送係	BBショップの従業員で、注文品の梱包、荷札貼り、及び発送を行う。
運送会社	商品を顧客へ届ける。〇〇運輸、××運送など
在庫管理システム	商品在庫を管理するシステム
会計システム	会計処理を行うシステム

2.3.2.2　ユースケース図を作成する

アクターの役割が決まれば、次にユースケースを検討する。先に定義したアクターの役割を検証してみると、顧客の役割と顧客窓口との役割にちぐはぐな点がある。つまり、顧客窓口が幾つかの役割を持っているのに、顧客はひとつの役割しかない。ユースケースを考える場合には、このように矛盾した役割や不足した役割があるので、何度も検証を繰り返す必要がある。

次にユースケースを洗い出す。この作業は、アクター毎にそのアクターが入力するものと出力

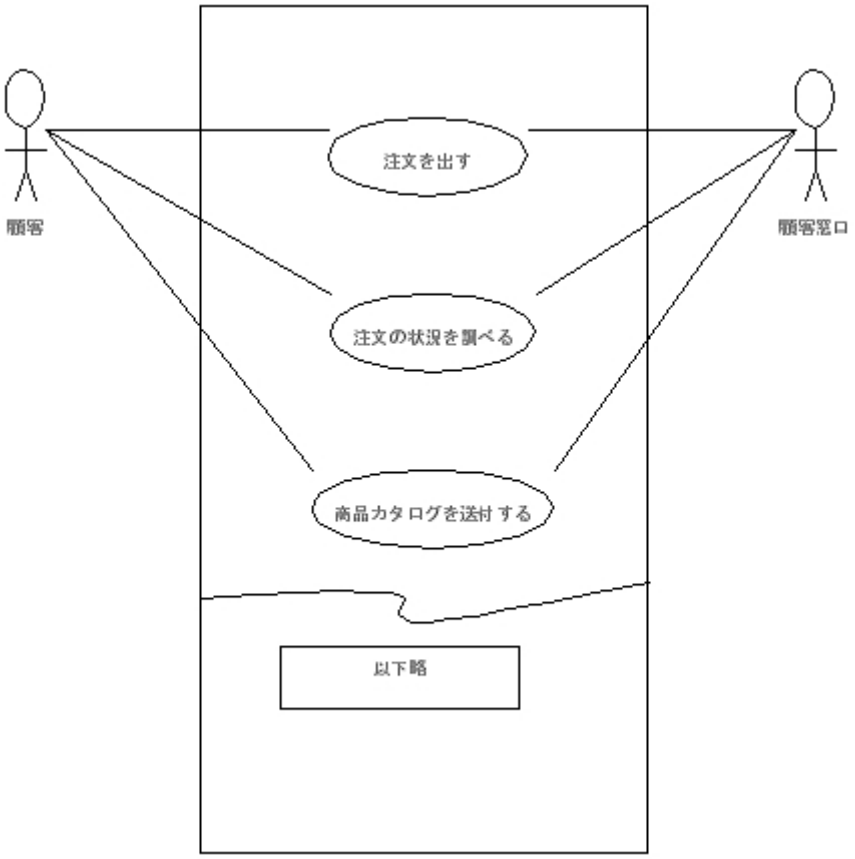
するものとを定義する。入力と出力は、どちらかが無い場合もある。また、記述方法は、「何々を何々する」と書く。

アクター	入力	出力
顧客から	注文を出す 商品カタログを受け取る 注文状況を照会する 商品を返品する 注文をキャンセルする 苦情を登録する	
顧客窓口から	注文を受け付ける 商品カタログを送付する 注文の状況を調べる 商品の返品を受け付ける 注文のキャンセルを受け付ける 苦情を処理する	
発送係から	宛名ラベルを印刷する 送料を計算する	
運送会社への		配送荷物を梱包する
在庫管理システムへ	バックオーダ品を入荷する	商品情報を提供する 在庫数を更新する
会計システムへ		顧客に請求する 仕入れ先に振り込む

初回のスケッチは概ねこのようなものになるだろう。これより、さらにシステムの概要、市場要因、リスク要因、仮決定事項、および検討中に発生した要求事項を見直し、全てのアクターが表現できているか、また機能はすべてユースケースとして表現できているかを検証する必要がある。

出来上がったアクター毎のユースケースを元に、ユースケース図を作成する。
最初にシステムの境界を矩形で表現し、矩形の中がシステムで実現する機能をユースケースとして記述し、矩形の外部にはアクターを配置する。その後、アクターとユースケースの関係を実線で結び、両者の関係を表現する。このユースケースとアクターの関係、及びアクターとアクターの関係には、「集約条件」や「拡張」などがあるが、ここでは「ビジネスシナリオの書き方」の説明という趣旨とは違うので省略する。また、全てのユースケースは書かず、最初の 3 つのユースケースのみを書き、ビジネスシナリオの書き方へつないでゆく。

ユースケース図の例（一部分）



2.3.3 ビジネスシナリオを書く

UMLではビジネスシナリオの書き方については何も規定されていない。従って、文章で記述しても良いわけだが、書き方が統一できないため、テンプレートを用いるのが一般的である。このテンプレートもどれがよいと言うものはなく、幾つかの書き方がある。

ここでは、3つの例を比較してみることで重要と考えられるユースケース項目を洗い出してみる。

例 1	例 2	例 3
ユースケース名	ユースケース名	ユースケース名
ユースケース概要		
アクター	主アクター	
事前条件	事前条件	事前条件
事後条件	事後条件	事後条件
イベントフロー		イベントフロー
基本フロー		基本パス
代替フロー		代替パス
シナリオ	主成功シナリオ	主シナリオ
	拡張シナリオ	副シナリオ
備考	関連情報	

これらの記述スタイルを比較した結果、次のような観点からビジネスシナリオの記述項目を定義した。

すなわち、ビジネスシナリオはどのように書いても良いが、読み手に分かりやすい内容とすべきである。読み手の候補としては、顧客、エンドユーザ、システム分析者、システム設計者、プログラマ、テスト担当者、管理者、及び経営者などが上げられる。これらの読み手に正確に伝えるためには、以下のものが書かれていれば十分であろう。

記述項目	記述内容
ユースケース名	当該ユースケース名でユースケース図に書かれたもの
ユースケース概要	ユースケースの役割と目的を 1 段落程度で書く。
アクター	このユースケースに関わるアクターを書く。
事前条件	このユースケースが実行される前に、システムが置かれている状態を書く。
事後条件	このユースケースが実行された後に、にシステムが置かれているべき状態を書く。
イベントフロー（パス）	
基本フロー（パス）	正常処理の流れ。（別名ハッピーパスとも言う）
代替フロー（パス）	基本フローのバリエーションとなるパス。すなわち、基本フローの代替手順を提供する。
主シナリオ	ユースケースの基本機能を記述する。
副シナリオ	ユースケースの代替手段とエラー条件を記述する。
関連情報	上記以外で読み手に伝えたい情報を書く。

では、これらのユースケース記述項目を用いて、BBショップの「注文を出す」というユースケースのビジネスシナリオを書いてみる。

ビジネスシナリオ	
ユースケース名	注文を出す
ユースケース概要	顧客が商品カタログから商品を選び、BBショップに注文をだす。
アクター	顧客。顧客窓口。
事前条件	ユーザ認証が完了し、システムにログインしていること。
事後条件	注文はシステムに保存され、注文確定のフラッグが付いていること。
イベントフロー（パス）	
基本フロー（パス）	① 顧客が画面から「注文する」を選択することでこのユースケースが始まる。 ② 顧客は自分の個人情報（名前、住所など）を入力する。 ③ 顧客は希望する商品の商品コードを入力する。 ④ システムは商品コードにより該当の商品価格と商品説明を画面に表示する。 ⑤ システムは商品品目の入力毎に現在の合計金額を記録する。 ⑥ 顧客は代金の支払方法を入力する。（クレジットカード、代引、銀行振込、電信振替） ⑦ 顧客は、購入確認の選択する。 ⑧ システムは入力された情報から、注文を「未決済」としてマークをつけて保存時、支払い情報を会計システムに転送する。 ⑨ 支払が確認されると、注文には「決済完了」のマークが付けられ、注文番号が顧客に返され、このユースケースが終わる。
代替フロー（パス）	① 注文のイベントが発生しない。 ② 基本フローの③で存在しない商品コードが入力された場合はエラーメッセージを顧客に返す。 ②基本フローの⑧で顧客が入力した情報に間違いがあれば、システムは顧客に対して訂正のメッセージを表示する。
主シナリオ	① Web 画面上の「注文」ボタンをクリックする。 ② 郵便番号＝「101-0001」、都道府県＝「東京都」、市町村＝「千代田区大手町」、番地＝「1 丁目 1 番 12 号」、名前

	＝「千代田三郎」、生年月日＝「1971 年 2 月 6 日」、メールアドレス＝「JP4001@juas.ne.jp」 ③ 商品番号＝「DX100294」」 ④ 価格＝「2，800 円」、商品説明＝「選び抜かれたスキンケアクリームです。毎日使って約 2 ヶ月間使えます」 ⑤ 合計金額＝「4，700 円」、内訳＝「スキンクリーム＝2，800 円、洗顔石鹸＝1，900 円」 ⑥ 代金の支払方法＝「クレジットカード」 ⑦ Web 画面上の「購入する」ボタンをクリックする。 ⑧ 支払い情報＝「入力された顧客情報、商品番号、金額、合計金額、未決済マーク（＝“0”）」 ⑨ 文番号＝「AX100230」を Web 画面に表示する。同時に「JP4001@juas.ne.jp」に注文確認メールを自動送付する。
副シナリオ	① イベント待ちの待機時間は 1 分とする。 ② 商品コードエラー＝「商品コードが正しくありません」 ③ 訂正のメッセージ＝「メールアドレスが間違っています」
関連情報	

今回の例では、「注文を出す」というユースケースを取り扱った。このユースケースでは、事前条件として、顧客がログインを完了している事のみである。これは、全体のユースケースの中で最初のものであるから、このような単純な条件となるが、それ以外のユースケースでは、事前条件が入力情報であり、事後条件が出力情報に相当し、ビジネス/シナリオが入力情報から出力情報を生成する関数に相当する。

この意味から、ユースケースでは、事後条件と事前条件がビジネスシナリオよりも、むしろ重要な位置付けにある。

2.3.4 ビジネスシナリオの次工程

ビジネスシナリオが完成すると、次は名詞と名詞句からクラス候補を抽出してクラス分析に着手できる。ここで重要なことは、ビジネスシナリオに用いられる用語を正確に定義しておくことである。これがクラスの分析を行う場合に効率を大幅に向上させる結果となる。

特に関係者が全員理解しているような用語でも、改めて定義を行うと違った解釈・理解をしている場合がある。プロジェクトの初期の段階から、こまめに用語集の編纂を行っておくことを薦める。

2.4　オブジェクトの発見と関係の定義

2.4.1 要求定義書からクラスを取り出す手法

一般的に、オブジェクト指向による分析や設計を行う場合にはクラスの抽出が最も難しいと言われている。また、クラスの抽出はこれを行う人によっても随分違ったものになる可能性がある。当委員会では、どのようにしたらオブジェクト指向での分析・設計を行う人に結果として差の少ない、すなわち共通のクラス抽出ができるか、これまで一般論を含めて考えてきた。ここでは、具体的な例題をもとに、これらの目的に近づくための方法論を考察する。

2.4.1.1　　クラスとオブジェクト

具体的にクラスの実験的な抽出作業を行う前に、クラスとオブジェクトの関係と違いについて整理しておく。

2.4.1.1.1 クラスとは

我々人間が認識できる現実の世界では、「モノ」として認識可能なオブジェクトが多数存在する。A さんが持っているノートPC、B さんが持っているノートPC、C さんが持っているノートPCなど、どれも同じノートPCと総称されるものだが、それぞれには他のノートPCと違う属性、例えば製造番号であったり、CPUの種類が異なったりする。オブジェクト指向の技術用語では、Aさん、Bさん、Cさんの持っているノートPCというオブジェクトは、ノートPCとして知られているオブジェクトのクラスのインスタンスであると言う。すべてのノートPCは共通に「状態」と「振る舞い」を持つ。しかし、それぞれノートPCの状態は互いに独立しており、他のノートPCとは異なる。

さて、ノートPCの製造メーカは、生産するノートPC毎に設計図をつくるのは大変効率が悪い。そこで、ノートPCの特徴が共有されている事実を利用して、同じ設計図からノートPCを大量に生産する。

同様に、ソフトウェアの世界でも同じ種類のオブジェクトが類似しているという事実を利用して、これらのオブジェクトの設計図を作成することができる。 これをクラスと呼んでいる

UMLでは、クラスとオブジェクトを次のような図で表す。

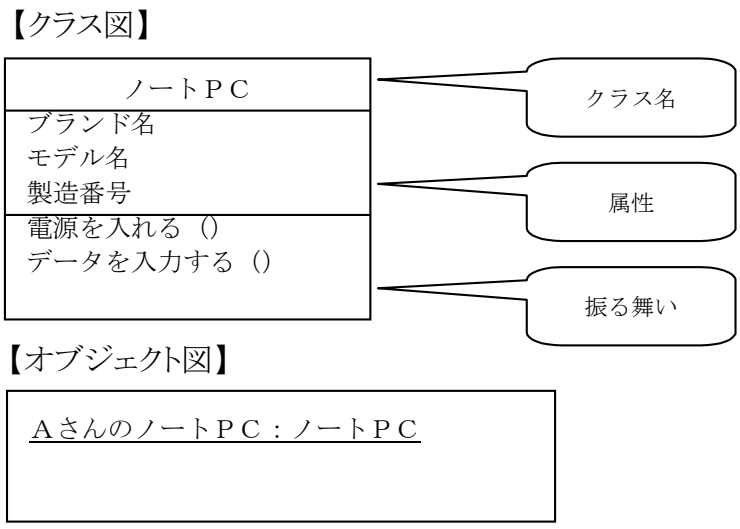


図 2.4-1

オブジェクト図は、長方形のアイコンで表し、中には「オブジェクトの実際の名前:クラス名」を書き、下線を引く。

2.4.1.1.2 オブジェクトとは

先に「クラスとは」で述べたように、オブジェクトとクラスの図は、互いによく似ている。 実際に、クラスとオブジェクトの違いは、しばしば混乱の原因になっている。現実世界では、ノートPCの設計図がAさんのノートPCそのものではないように、クラスはそのクラスが記述しているオブジェクトそのものではないことは明らかである。ソフトウェアでクラスとオブジェクトとの混乱が起きているのは、ソフトウェアで扱うオブジェクトが、現実世界のオブジェクトの電子モデルであり、電子モデルそのものが抽象的概念であることに起因していると思われる。この混乱が原因で、十分にクラスとオブジェクトを区別しないまま、システムの分析や設計局面で言葉の誤用が行われてきたため、さらに混乱を増幅したのであろう。

2.4.1.2 クラス抽出のための情報源

クラスを抽出するための情報源としては、次のようなものがある。

- システム化要件定義書
- システム化要件定義書に関係した議事録
- ユーザからのアプリケーション要求文書
- 関係部署からの要求文書
- 現在運用されている画面と帳票
- システム化対象領域に精通した業務の専門家

これの全てを例示してクラスの抽出例を示すのは、膨大な紙面を要する。ここでは、システム

化要件定義書の簡単な例示を行い、クラスを抽出してみる。

クラスを抽出する方法には次のようなものがある。

① 名詞・動詞分析法

ユースケースや用語集から、名詞、名詞句を探し出して、クラスの候補とし、動詞、動詞句を探し出して、メソッドの候補として作成していく方法。

② CRC 分析法

1～2週間の短期間でブレーンストーミング、あるいはそれに近いやり方で、クラス(Class)、債務(Responsibilities)、協調(Collaborations)に分けられたCRCカードを用いて分析していく方法である。(注1)

③ 「瞬間～時間間隔」、「役割」、「パーティ／場所／物」、および「説明」といった 4 つのアーキタイプに分類・色分けして、パターンに当てはめて分析していく方法。

最初のモデルを作成する場合の強力な手法となっている。

今回は、UML で推奨されている名詞、名詞句からクラスを抽出する方法を実証する。名詞や名詞句に着目する理由は、それが業務領域において興味をもたれていると認識でき、クラスの候補になる可能性が高いからである。

さて、クラスを抽出する担当者やチームにとって、最初の情報源になるのはシステム化要件を記述した文章となる。この文章については、「要件定義書」や「システム化要件書」、「要件メモ」、「システム機能要件定義書」など様々な呼び方がされており、一意に決まったものはない。重要な点は、それらの要件を定義した文章が公式なもの、すなわちシステムを保有する計画の企業が、自ら「我々が求めるシステムの機能は、ここに記載されていることである」と正式に承認した文章でなければならない。

システムを分析する場合に、最初に扱う情報源はシステム化要件を定義した文章であろう。実際に使われるシステム化要件はここでは掲載できないので、当委員会で次のようなシステム要件を例題として定義した。

ＪＵＡＳ銀行は、２００５年の４月にインターネット銀行を設立する予定である。インターネットを用いた銀行窓口サービスを提供する。

Web を利用したインターネット銀行サービスは、入金処理と出金処理を除き、銀行窓口に設置された ATM とほぼ同様な金融取引の機能を持っている。すなわち、残高照会、取引明細照会、電信振込、口座間振替、および各種照会情報の印刷機能である。このサービスの取り扱う口座の種類は、当座預金と普通預金の２種類である。さらに、預金の実績に応じて、無担保のローンサービスも行う。それ以外の付帯サービスとしては、午前０時に行うセンターカットによって、預金口座からの自動引き落としができなかった場合には、その旨電子メールで本人当てに通知を行い、当日１５：００までに入金されれば、引き落とし処理を行うことができるようにする。

銀行カードに相当する本人確認処理は、インターネット取引のための契約番号と、暗証番号によって行う。また、送受信についてはすべて暗号化処理を行う。さらに、成りすましを防止するために、セッション管理を行い１分以上何らかのメッセージやボタン操作などのイベントが入力されない場合には、セッションを強制的に停止する。

2.4.1.3　　クラス候補の選択

このシステム化要件定義書から、名詞、名詞句に注目し、下線を引く。これは機械的な作業である。

ＪＵＡＳ銀行は、２００５年の４月にインターネット銀行を設立する予定である。インターネットを用いた銀行窓口サービスを提供する。

Web を利用したインターネット銀行サービスは、入金処理と出金処理を除き、銀行窓口に設置された ATM とほぼ同様な金融取引の機能を持っている。すなわち、残高照会、取引明細照会、電信振込、口座間振替、および各種照会情報の印刷機能である。このサービスの取り扱う口座の種類は、当座預金と普通預金の２種類である。さらに、預金の実績に応じて、無担保のローンサービスも行う。それ以外の付帯サービスとしては、午前０時に行うセンターカットによって、預金口座から残高不足で自動引き落としができなかった場合には、その旨電子メールで本人当てに通知を行い、当日１５：００までに入金されれば、引き落とし処理を行うことができるようにする。

銀行カードに相当する本人確認処理は、インターネット取引のための契約番号と、暗証番号によって行う。また、送受信についてはすべて暗号化処理を行う。さらに、成りすましを防止するために、セッション管理を行い１分以上何らかのメッセージやボタン操作などのイベントが入力されない場合には、セッションを強制的に停止する。

さて、名詞や名詞句を日本語の文章から選び出す作業は意外と難しいことに気づくはずである。例えば、「入金処理」は、名詞かどうかは「処理」だから機能、すなわち動詞ではないかと

考えることもできる。しかし、「入金を処理する事」という意味を簡略化して「入金処理」としていることから、本来の名詞は「入金」にあると考え、後で整理することを全手として、クラスの候補として取り上げるべきである。同様な事は、「金融取引」や「残高照会」、「電信振込」という言葉についても言うことができる。

2.4.1.4 クラス候補の一覧作成

抽出したクラス候補を一覧に整理する。

JUAS銀行	インターネット銀行	インターネット
銀行窓口サービス	インターネット銀行サービス	入金処理
出金処理	銀行窓口	Web
ATM	金融取引	残高照会
取引明細照会	電信振込	口座間振替
各種照会情報	印刷機能	口座
当座預金	普通預金	預金
ローンサービス	付帯サービス	センターカット
預金口座	自動引き落とし	電子メール
ローンサービス	本人	入金
引き落とし処理	銀行カード	本人確認処理
インターネット取引	契約番号	暗証番号
送受信	暗号化処理	成りすまし
セッション管理	メッセージ	ボタン操作
イベント	セッション	無担保
残高不足		

この一覧から重要なクラスを抽出するのが次の作業となる。この作業の目的は、

① クラス図に書かれる重要なクラスの識別

② システム範囲に入らない無関係なクラス候補の破棄

③ 残ったクラス候補をレビューし、残すか破棄かを決める

という 3 点にある。

重要なクラス候補かどうかの判断は、

① 主要なオブジェクト（実体）に直接関連するクラス候補

② 繰り返し現れるクラス候補

③ 状態の変化が伴うクラス候補

といった観点から判断する。

また、破棄の対象となるクラス候補は、

① システムの対象外と考えられるクラス候補
② 同じ意味を表し違う言葉で表現されているもの
③ 日付や時間など属性にあたるもの
④ 同じクラスの範囲だが抽象のレベルが異なるもの
といった観点から判断する。

最後に重要なクラス候補でもなく、破棄の対象にならないクラス候補が残る。ここが非常に重要な局面である。その理由は、重要か、整理可能か、あるいは破棄かが明確にならないという状況は、システム化の対象範囲が不明確であるから起きる。さらに、クラス候補を選択する上で、クラス候補の名前が明確に定義されていないため、それが何を意味しているのか分からないために生まれてくる現象である。

システム化対象範囲を決めるためには、このシステムが動くための入力は何なのか、出力は何なのか、システム化要件定義書で述べられている一連のプロセスはどこから開始されどこで終わるのか、このシステムの責務は何かを検討して決めることができる。

これは、オブジェクト指向技術に関わらず、一般的なシステム開発方法でも同じ事を行う。

今回の例題で示されたシステム化の範囲は次のとおりと考えられる。

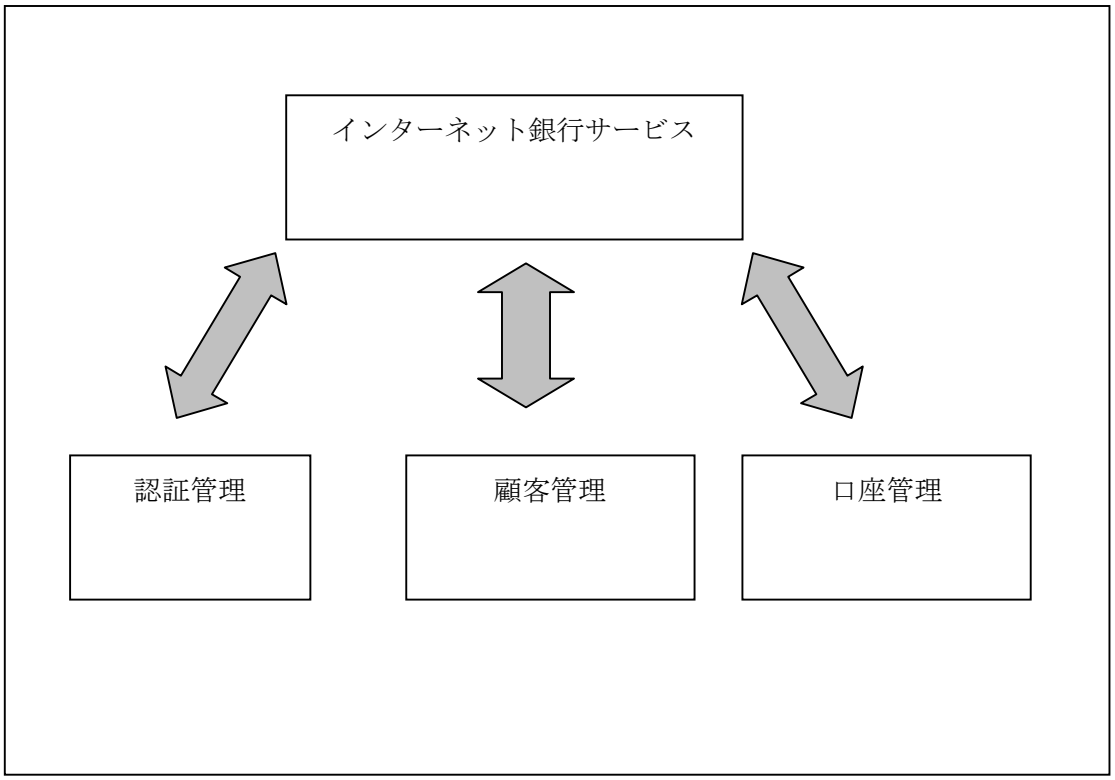


図 2.4-2

2.4.1.5　　アーキテクチャによる整理

クラス候補を整理するために次に行う方法は、アーキテクチャを加味することである。システムに登場するクラスは、大きく 4 つのカテゴリーに分類される。これはMVCモデル（Model, View, Control）に外部インタフェースを加えたものである。

① プレゼンテーション
　GUI（ユーザとの相互作用）
② データ-ソース
　データベースなど
③ ドメイン
　データソースとプレゼンテーションをつなぐもの。アプリケーション・ドメインとも言う。
④ 外部インタフェース
　当該システムと外部システムとの相互作用

2.4.1.6　　属性による識別

最後のクラス候補選択の観点は、クラス候補の性質について調べることである。クラスの性質とはクラスの属性のことで、クラスか属性かの違いを見極める方法は、クラスが自分自身の状態を変化させられるのに対して、属性は状態を変更できないといった視点を導入することで可能となる。

例えば、銀行口座は預金残高という属性を持ち、口座はその残高という属性が変化するもので、残高は刻々と変化をするその時点での数値属性であるという考え方をする。別の言い方をすれば、クラスは振る舞いを持つが属性は振る舞いを持たないということである。
逆にあるクラス候補が属性であると判断できたならば、ではその属性はどのクラスのものかを分析することで、重要なクラスの発見につながる。

2.4.1.7　　クラス候補の整理

これまで述べてきた判断基準を元に、リストアップしたクラス候補を絞り込む。分かりやすいようにマトリクス図にして示す。

これはあくまでも実験的に分類したもので、実際のクラス抽出に際しては、実務面からさらに検討を加える必要がある。その結果、クラス候補から復活するものもあるし、逆に破棄されるものもあり得る。

この作業を反復して行うことにより、より現実に即したクラスが抽出できるようになる。

クラス候補	判断基準								
	重要クラス	システム対象外	未決定	属性	同義語	プレゼンテーション	データソース	ドメイン	外部インタフェース
JUAS銀行		○							
銀行窓口サービス		○							
出金処理	○							○	
ATM		○							
取引明細照会	○					○			
各種照会情報				○					
当座預金				○					
預金口座				○					
ローンサービス									○
引き落とし処理		○							
インターネット取引		○							
送受信		○							
セッション管理			○						
イベント				○					
残高不足				○					
インターネット									○
入金処理		○							
Web		○							
残高照会	○					○			
口座間振替	○						○		
口座	○								
預金				○					
センターカット									○
電子メール									○
入金		○							
本人確認処理					○				
暗証番号				○					
成りすまし		○							

ボタン操作				○					
無担保		○							
インターネット銀行		○							
インターネット銀行サービス		○							
銀行窓口		○							
金融取引	○								
電信振込					○				
印刷機能	○					○			
普通預金				○					
付帯サービス		○							
自動引き落とし		○							
本人	○								
銀行カード		○							
契約番号				○					
暗号化処理				○					
メッセージ				○					
セッション					○				

2.4.1.8　　クラスに関する専門的な考察

論理学の本としてアリストテレスの著「オルガノン」がある。はオルガノンの構成は、

- ◆ カテゴリアイ（範疇論）
  カテゴリアイ（範疇論）は実体（オブジェクト）をベースとしたカテゴリー論
- ◆ 命題論（言葉によるものごとの明示について）
  「命題論」名詞と動詞，および肯定と否定についての定義
- ◆ 分析論前書
  三段論法
- ◆ 分析論後書
  科学的論証（証明）」とは何かということを定義
- ◆ 論拠集（トピカ
- ◆ ソフィストの論法について

という著作でなされている。

アリストテレスは人間が認識する対象物を「実体」と呼んだ。この概念を使って世界をモデル化しようと試みた訳である。アリストテレスは、「存在」には「実体」と「付帯的存在」があるとし、「付帯的存在」は分析する意味のない偶然的な存在とした。

「実体」には、「第一実体＝個物（基体）」または「第二実体＝形相」があり、「第二実体」には「普遍者」=「カテゴリーの形式＝（実体，質，量，能動，受動，場所，時間，関係，位置，状態）」と「類」および「本質」によって分類が行われる。

オブジェクト指向は哲学的で理解が難しいとされているが、その本質はオブジェクトが実体であり、第一実体がインスタンス、第二実体がクラスというアリストテレス的な分類学に端を発していることに起因しているのではないだろうか。これらの抽象的な概念を、如何に理解しやすく応用の利く方法論に変換するかがオブジェクト指向を普及させる手立てのひとつであろうし、当委員会の役割でもあると認識する。

（注1）『Object Design: Roles, Responsibilities, and Collaborations』（Rebecca Wirfs-Brock／Alan McKean 著、Addison-Wesley 刊）。

（注2）『Object Design: Roles, Responsibilities, and Collaborations』（Rebecca Wirfs-Brock／Alan McKean 著、Addison-Wesley 刊）。

2.4.2 CRC カード法 (Class Responsibility Collaboration) について

2.4.2.1 背景にある考え方

クライアント／サーバモデルでは、

(1)クライアントは、サーバへ、サービスの提供を要求する、

(2)サーバは、要求に応じてサービスのまとまり(Sets)を提供する、

(3)クライアントとサーバとの間には、サービスの提供を取り決めた契約(contract)がある

と考える。ここに言うサービス提供契約とは、要求(クライアント側の責務)に応じて提供されるサービス(サーバ側の責務)を取り決めたものである。

クライアント／サーバモデルは、IT活用システムばかりでなく、ビジネスを行う人間の活動(ビジネス・システム)をモデル化する場合に汎用的に活用できる。ビジネス活動に焦点をあわせてクライアント／サーバモデルを見てみると、クライアントからの要求に従ってサーバの責務が駆動される活動というメカニズムが浮かび上がってくる(活動分析[1])。このダイナミックな活動に焦点を合わせれば、クライアント／サーバモデルによるモデル化は、責務駆動型(Responsibility Driven Approach)[2]モデル化と捉えることができる。

オブジェクト指向分析／設計のモデル化の方法論としては、責務駆動型モデル化とデータ駆動型モデル化とがある[3]。世の中では、データ駆動型モデル化のほうが、普及している。しかし、人間が主体で行なわれる経営やビジネスの活動を素直にモデル化する場合には、より能動的な活動に着目して分析を行い、システムを構築しようとする責務駆動型モデル化のほうが扱いやすいのではないかと考える。ここでは、責務駆動型モデル化の方法を紹介する。

責務駆動型の方法論に基づく、具体的なシステムの分析／構築では、オブジェクトを抽出し、

(1)各オブジェクトがどのようなふるまい(action)に責務をもっており、

(2)個々のクラスが責務を有効に果たすためにオブジェクト間でどのような情報を共有し、

(3)どのように責務の委譲(delegation)をオブジェクト間で行い、協調(collaboration)を行うかという仕組みを明らかにする。さらに、その仕組みが、システム全体の責務を効率的かつ効果的に果たすことが出来るかどうかを確認することが重要になる。責務駆動型の方法論に基づくシステムの分析／構築作業に有効なツールが、CRC カード法[4]である。CRC カード法は、エクストーリム・プログラミング(XP eXtreme Programming)で有名な Kent Beck と Ward Cunningham によって考案された。

---

[1] Kenneth S. Rubin and Adele Goldberg: Object Behavior Analysis, CACM vol.35-No. 9 (1992/9)

[2] Rebecca Wirfs－Brock and Brian Wilkerson: Object-Oriented Design: A Responsibility-Driven Approach, ACM OOPSLA '89 Proceedings (October 1-6, 1989) pp.71-75

[3] 中谷多哉子:ユースケースをマスターしよう オブジェクト指向に強くなる(青山幹彦他編著、技術評論社 2003)の第2章(pp.105-156)

[4] Kent Beck and Ward Cunningham: A Laboratory For Teaching Object-Oriented Thinking , ACM OOPSLA '89 Proceedings (October 1-6, 1989) pp.1-6

CRC カード法は、オブジェクト指向を学ばせるための教育ツールとして初めて紹介されたが、その後、David A. Taylor によって、コンバージェント・エンジニアリングの手法として紹介され有名になった[5]。

2.4.2.2 CRC カードとは

CRC カードは、小型の紙のカード(Beck と Cunningham は、4インチ×6インチの index card を提案[6])であって、クラス毎に1枚作成される(下図参照)。各カードには、下図に示される通り、左側(2/3)に、当該クラスが果たすべき責務、右側(1/3)に、当該クラス以外に、責務とその権限を委譲して、協調して責務を果たすことになる他のクラス(Collaborators 協調クラス)を記入する。

図 2.4-3 CRC カード

2.4.2.3 CRC カードの記入例

銀行預金を例にとって、CRC カードの記入例を以下に示す[7]。預金者、銀行、行員、預金口座、利子、金庫などが、クラスである。

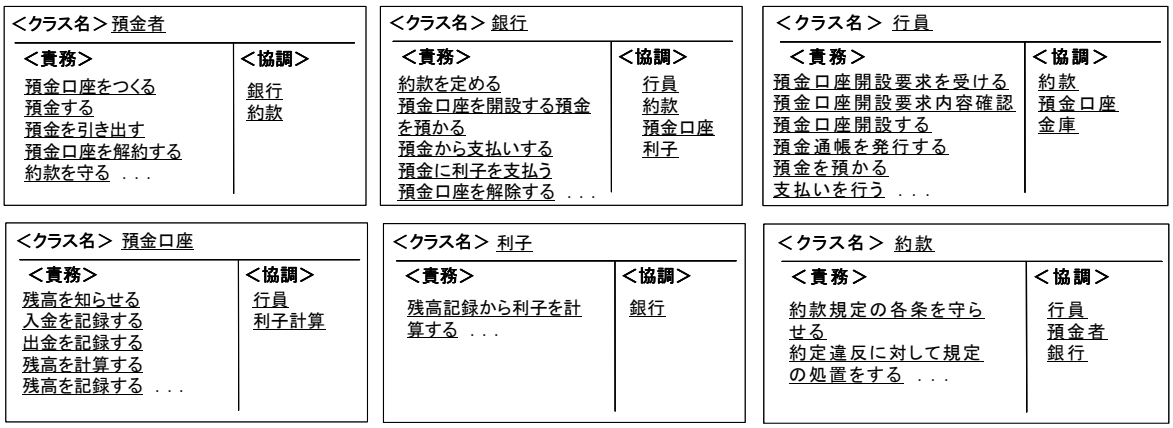


図 2.4-4 CRC カード記入例(銀行預金の例)

[5] David A. Taylor: Business Engineering with Object Technology, Wiley 1995

[6] 承前(Kent Beck and Ward Cunningham: A Laboratory For Teaching Object-Oriented Thinking , ACM OOPSLA '89 Proceedings (October 1-6, 1989) pp.1-6)

[7] 報告者は、銀行業務は全く知らないので、単なる預金者としてモデル化している。

2.4.2.4　　CRC カードの利用方法

ここではコンバージェント・エンジニアリングの手法[8]として紹介されている手法を基本として、D. Bellin と S. S. Simone の著作[9]を参考にし、報告者の知見を加え、CRC カードを用いて、オブジェクト指向によるモデル化をプロジェクト・チームで行う方法を述べる。最近、OMGが提唱しているモデル駆動開発（MDD: Model-Driven Development）への関心が高まり、モデルについて精緻な議論が展開されつつある現状を踏まえて、ここで対象とするモデルの位置づけを明確にして置くことにする。

次の図は、David S. Frankel[10] によるものであるが、比較的分かりやすいので引用する。
下記に示す CRC カードの利用法で作成するモデルは、次図の「ビジネスモデル」（自動化、IT化とは関係なくビジネスの振舞いを表すモデル）である。

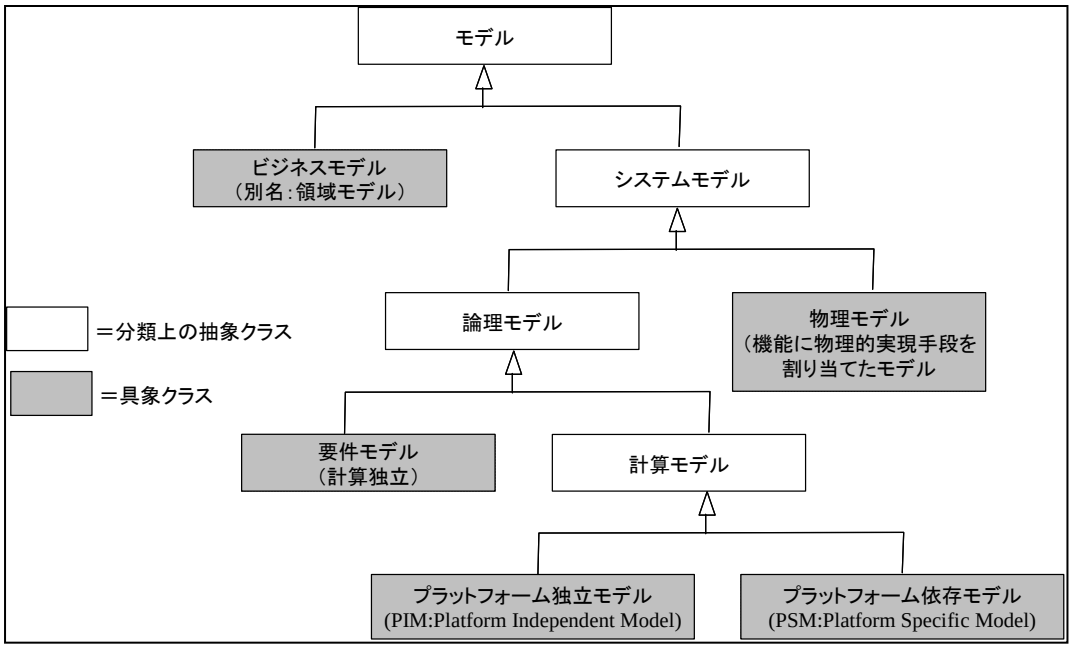


図 2.4-5　　モデルの分類（David S. Frankel による[11]）

[8] 承前（David A. Taylor: Business Engineering with Object Technology,　Wiley 1995）

[9] D. Bellin　and S. S. Simone: The CRC Card Book , Addison-Wesley Pub Co; 1st edition (June 4, 1997)

[10] David S. Frankel:　Model Driven Architecture John Wiley & Sons: (January 10, 2003)　8.1 P.193（日本語訳 エスアイビー・アクセス社(2003/11/1)版では、210 頁）

[11]同上の David S. Frankel の分類のうち、「物理モデル」の解釈は、著作者の解釈と報告者の解釈とは異なっている可能性があるので注意を要する。

2.4.2.4.1 ＣＲＣカード・セッションの開始に先立つ作業

① プロジェクト・チームの選抜

チームメンバとして、

・改革の対象とする業務の精通者、

・オブジェクト指向設計の理論と実践に精通したモデル作成者、

・ファシリテータ（チームリーダ）、

・記録係（ライブラリアン）

からなる4～10名程度のチームを結成する[12]。6名のチームであれば、業務精通者2名、モデル作成者2名、ファシリテータと記録係各1名になる。10名のチームであれば、業務精通者とモデル作成者とをそれぞれ2名増員して4名ずつとする。ファシリテータは、チームを統括する技能を有し、経営工学の知見を持ち、オブジェクト・モデリングの一通りの知識もあることが条件となる。記録係も、議論が理解でき、正しい情報を有効な形で記録できるために、オブジェクト・モデリング技法の教育を受けている必要がある。オブザーバは議論の中途半端な参加になるため出来る限り参加させないことが望ましい。

② セションチームの準備

セションを開始する前に下記の準備が行われる必要がある。

・チームメンバは、1週間のセションに専念できるように専従とする。

・作業用の個室が用意されること。

個室には、10名程度が着席できるテーブル、テーブルの周りを自由に歩きまわれるスペース、一方向はテーブルから離れてロールプレイイングができるオープンスペースがあること。

・作業用備品が整えられていること。

チームメンバ全員が記入したり、ポストイットを貼たり出来るホワイトボードやフリップチャートが1セット用意されており、十分なカラーマーカーやポストイットが用意されており、十分な白紙の CRC カードが用意されていること。

・チームによるセッション（合宿）に入る前にミーティング（1回以上）が行われること。

チームによるセッション（合宿）が行われる前に、チームメンバ全員に、業務に係わるビジョンとミッションが徹底的に伝えられ、メンバ間でプロジェクトの目標、構築すべきモデルの範囲についてある程度合意ができていることが必要である。また、対象業務の典型的な業務についてシナリオの概要（上記の銀行預金の例では、次の図にみられる

[12] 承前（David A. Taylor: Business Engineering with Object Technology, Wiley 1995）では、10名以下、
承前（D. Bellin and S. S. Simone: The CRC Card Book, Addison-Wesley 1997）では、6名以下が最適と主張している。

程度のシナリオ)、あるいは、今回のプロジェクトに対する要件定義書(要求確認書)が用意されていることが望ましい。また、現業務の問題点等も明らかにされていることが望ましい。

シナリオ1: 預金口座の開設
(a)預金口座の開設を希望する者(将来の預金者)は、約款を読み、下記の情報を申込書に記入し、初回の入金を添えて、銀行窓口の行員に提出する。
・住所(フリガナ)、郵便番号、電話番号、電子メールアドレス、氏名(フリガナ)、生年月日、性別、登録印(押印)、その他(預金種別、勤務先、家族構成、住まい状況など)
(b)銀行窓口の行員は、提示された申込書を受け取り、
・記入漏れをチェックし、
・申込書に記入された初回の入金額と入金とが一致しているかどうかチェックする。
(c)申込みに不備がなければ、
・口座番号を決定し、口座を開設し、
・入金を金庫に納め、
・通帳に、申込受理店名、電話番号、店番号、口座種別、口座番号、初回の取引記録(年月日、事由、入金額、残高)を記入して、通帳を発行する。

シナリオ2: 通常の取引
(1)預金
(a)預金者は、下記の情報を申込書に氏名と預金金額とを記入し通帳を銀行窓口の行員に提出する。
(b)銀行窓口の行員は、提出された申込書を受け取り、
・記入漏れをチェックし、
・申込書に、口座種別、店番号、口座番号、入金事由を記入する。
・申込書に記入された入金額と入金とが一致しているかどうかチェックする。
(c)申込みに不備がなければ、
・入金を金庫に納め、
・通帳に、取引記録(年月日、事由、入金額、残高)を記入して、通帳を預金者に渡す。
(2)預金引出
(a)預金者は、下記の情報を申込書に氏名と預金引出金額とを記入し、押印し、通帳を銀行窓口の行員に提出する。
(b)銀行窓口の行員は、提出された申込書を受け取り、
・記入漏れをチェックし、
・申込書に、口座種別、店番号、口座番号、入金事由を記入する。
・預金引出金額以上の預金残高があることを確認する。
(c)申込みに不備がなく、預金引出金額以上の預金残高があれば、
・預金残高から支払金額を差引き、新残高を計算し、
・通帳に、取引記録(年月日、事由、支払金額、新残高)を記入して、
・金庫より支払金額に相当するお金を取り出し、
・通帳と支払金を預金者に渡す。

シナリオ3: 預金口座の解約
(a)預金口座の解約を希望する者は、申込書に氏名を記入の上、登録印を押印し、預金通帳と共に、銀行窓口の行員に提示する。
(b)銀行窓口の行員は、提示された申込書と預金通帳とを受け取り、預金口座情報に照らして、登録印の印形を、照合チェックを行う
(c)銀行窓口の行員は、申込みに不備が無ければ、預金残高を確認し、利息を計算し、手数料控除を行い、払戻し金額を計算し、清算書を作成する。払戻し金額額に相当する金種を揃える。
(d)銀行窓口の行員は、預金口座を削除し、預金通帳を無効とする。
(e)銀行窓口の行員は、解約申込者に、清算書、払戻金、無効処理を施した預金通帳を渡す。

図 2.4-6 シナリオの記入例(銀行預金の例)

2.4.2.4.2 ＣＲＣカード・セッションの開始

セションは、1週間（5日間連続）程度の合宿形態が望ましく、標準的スケジュールは、次の表に示す通りである。

CRC カード・セッションの標準スケジュール

（David A. Taylor: Business Engineering with Object Technology　Chap.5 より翻訳）

曜日	AM	PM
月	オブジェクトを探す	モデルをスケッチする
火	責務を割り当てる	関係を定義する
水	トーク・スルー	モデルを拡張する
木	ウォーク・スルー	モデルを一旦くずす（break する）
金	ラン・スルー	モデルを最適化する

図 2.4-7

セションは、次の手順で進められる。

①　目的の再確認

セションの冒頭で、既に合意が得られている、業務改革のビジョン、ミッション、プロジェクトの目標、構築すべきモデルの範囲、モデル化の視点（この例では、ビジネスモデルを作成するという視点）、対象業務の典型的な業務についてシナリオ、現業務の問題点等を再度確認する。

ホワイトボードやフリップチャートの最上位部に、セションの目的を書く。

②　クラスの候補のリストアップ

対象業務の業務についての典型的なシナリオの記述からオブジェクトのクラス（以下、クラスという。）の候補をリストアップする。とりあえず記述の中に出てくる名詞[13]をリストアップする。これらの名詞の中から、

・業務に欠かせない何らかの行為を行うもの(ex.　預金者、行員、銀行など)、

・業務に取って大切で、管理が必要なもの(ex. 預金口座、お金（金庫）、約款など)、

をクラスとして抽出する。

なお、対象業務の業務についての典型的なシナリオの記述にとらわれることなく、チーム全員のブレーンストーミングによるクラスの候補をリストアップに加えることが望ましい。

チーム全員のブレーンストーミングでは、ラウンドロビンテクニック（テーブルの周りに着席したチームメンバ全員に対して、一定方向に、順次クラスの候補を挙げていって貰

[13] 名詞：自立語で、活用がなく、主語となる品詞。

い、誰もクラスの候補を挙げられなくなるまで繰り返す方法）を採用することが推奨される。

David A. Taylor[14]を含めて、一般的な責務駆動型では、クラスのうち、「業務に取って大切で、管理が必要なもの」のような受動的なクラスを見落とすことが多い。しかし、「業務に欠かせない何らかの行為を行うもの」だけを取り上げると、現状の業務で業務を遂行する組織や人に囚われて、将来行為者として登場するようなものを見落とし、抜本的な組織改革の芽を摘んでしまう危険性など、単なる業務の現状分析に留まってしまう可能性があるので注意を要する。

動作を示す名詞は、動作の主体をクラスとして抽出し、動作は動詞化して動作の主体の「責務」に加える。（ex.「支払い」→動作の主体である（銀）行員をクラスとし、預金を「支払う（＝出金する）」を行員クラスの責務とする。）

クラスか、あるクラスの属性（クラスの特性を規定する名詞）かを区別する必要がある。クラスの属性は、とりあえず、CRC カードの裏面にリストアップする。

③ クラス毎 CRC カードを1枚作成し、責務を記入する。

業務に欠かせない何らかの行為を行うものからなるクラスについては、行為者の責務を CRC カードの左の責務欄に記入する。業務に取って大切で、管理が必要なものからなるクラスについては、その管理者の責務（管理というあいまいな言葉が包含している業務を具体的に詳細化した表現）を CRC カードの左の責務欄に記入する。

同じような内容の CRC カードがたくさん作られ、煩雑になるようであれば、抽象クラスを導入して、共通の振る舞いを汎化し、継承を活用するようなことも考え、記述量を減らすことも、実務面では必要になる。これには、オブジェクト指向技術に精通した技術者のサポートが必要である。汎化は、主として煩雑性や複雑性を除く「最適化」の手法である。ビジネスモデルの作成の場合では業務を的確に表現するかどうかが最大の関心事であるため「最適化」の重要度は必ずしも高くない。しかし、概要設計から基本設計を行う場合（ＭＤＤでは、「ビジネスモデル」から「要件モデル」または、「プラットフォーム独立モデル」への変換する場合）では、同じ振る舞いになる責務は出来る限り同じ表現にする[15]などと共に、汎化の洗練などの「最適化」が、主要な関心事となる。

④ 協調クラス（責務及び権限の委譲先）を記入する。

あるクラスの責務としては、主たる責務か、責務に伴う副次的な責務かを区別する。副次的な責務については、その責務を主たる責務とするクラスに、その責務及び権限を委譲する。お互いに、主たる責務を持つクラス間での協調によって、責務を果たすような

[14] 承前（David A. Taylor: Business Engineering with Object Technology, Wiley 1995）

[15] 同じ振る舞いになる責務は出来る限り同じ表現にするということは、多相性(polymorphism)を導入する準備となる。

モデル化を行う。これは、実業界で行われているアウトソーシングとかビジネスアライアンスと基本的に同じ考え方である。

次図は、行員(窓口出納係　teller)は、預金口座(の管理者)と協調して、預金口座を開設し、預金の出納(入金、支払い業務)の責務を果たすことを示している。

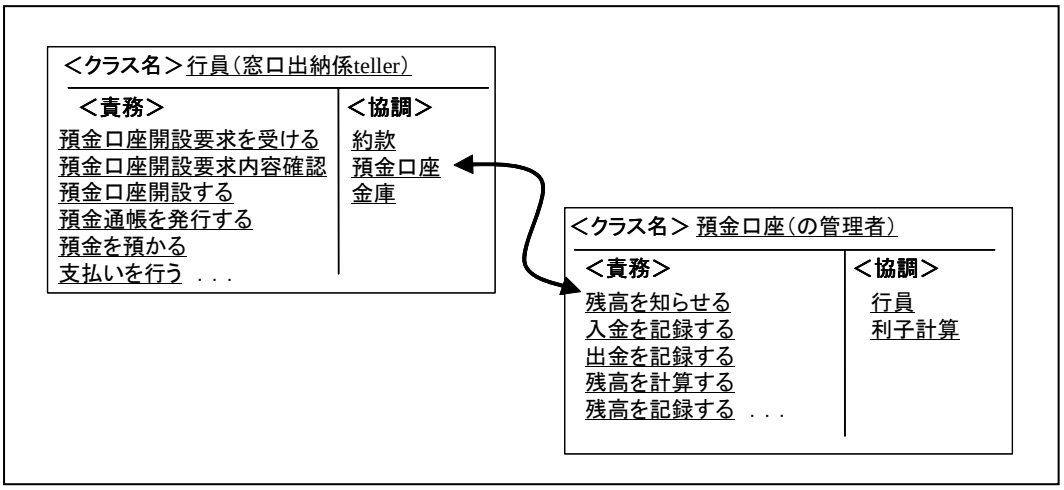


図 2.4-8　協調関係説明図

⑤　業務のシナリオを洗練する。

セションの開始にあたって、業務のシナリオを用意するし、この段階では、業務のシナリオを洗練し、下記の分類を行う。

- 基本的かつ重要な業務シナリオ(正常処理に相当する)
- 主要な例外のシナリオ(主要な例外処理に相当する)
- 稀な例外のシナリオ(稀な例外処理に相当する)

先の「図 2.4-7 シナリオの記入例(銀行預金の例)」に示したシナリオ1、シナリオ2及び、シナリオ3は、基本的かつ重要な業務シナリオに属し、残高以上の支払いのシナリオなどは主要な例外のシナリオ、盗難通帳による支払いのシナリオなどは稀な例外のシナリオに該当するのではなかろうか。

シナリオは、1つの動作を1行で書き表すようにする。最終的には、完全なシナリオの一覧が出来る必要があるが、初期の段階では、基本的かつ重要な業務シナリオ(正常処理に相当する)が出来ていれば良い。

これらのシナリオは、次に示す3種類のリハーサルの「台本」(スクリプト)になる。

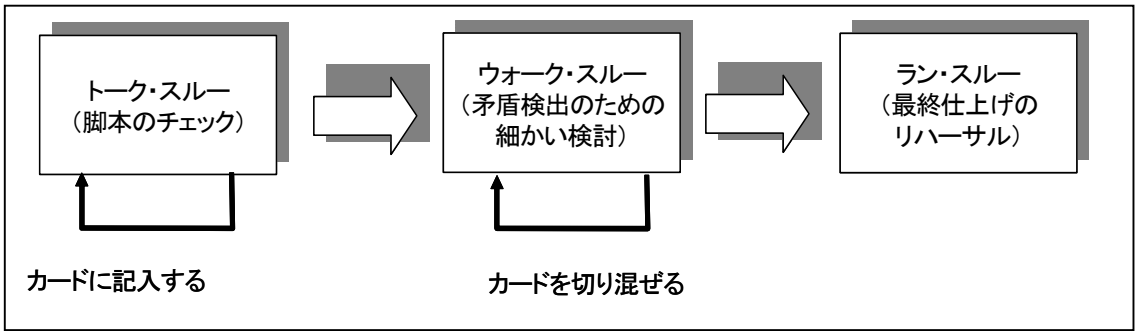


図 2.4-9　三種類のリハーサルの手順

（David A. Taylor: Business Engineering with Object Technology,　Wiley 1995　p.80
（翻訳　トッパン「コンバージェント・エンジニアリング入門」1996（絶版）　p.102 ）

⑥　チームの協議検討（トーク・スルー）によりクラスに漏れなくす

CRCカードをテーブルの上に置き、上記⑤で作成／洗練した基本的な業務シナリオに沿って、ストーリを通し（トーク・スルー）、モデルが適正な振る舞いを示すかどうか、チーム全員の目でチェックする。必要があれば、クラスの漏れや重複、責務の漏れや重複、責務の表現の修正、責務の多すぎるクラスや少なすぎるクラスを分析し、クラス編成改善などを行い、モデルの改良を行う。

次に続くウォーク・スルーが一応流れる事を確認することが重要である。

また、各 CRC カードに記載されている協調関係について、どの責務を果たす時に、どの協調先と強調するかを対応させて整理する必要がある。

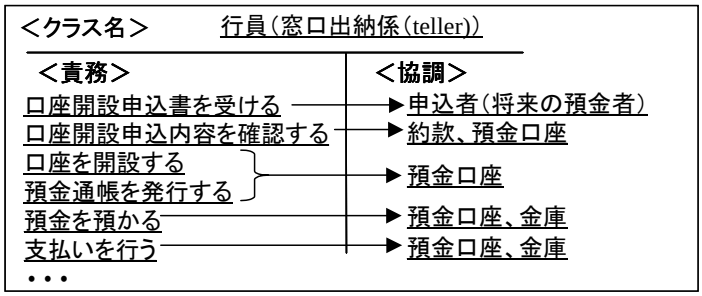


図 2.4-10　責務と協調の対応付けの例

課題の複雑さにもよるが、トーク・スルーは、2～10回程度は、繰り返す必要がある。特に協調関係については、個々のオブジェクトのクラスからどのように要求され、どのようにサービスを提供するかを明らかにする必要がある。これは、サービス・インターフェースの形式的定義の準備作業になる。

⑦　ウォーク・スルー（ロールプレイ＝最も重要なリハーサル）の実施

上記⑥のトーク・スルーが終了したら、次のステップとして、CRCカードを切り混ぜて（シャッフルして）全カードを1枚ずつファシリテータ（チームリーダ）と記録係（ライブラリアン）以外のチーム全員（「役者」）に配る。

この際出来れば、

- 協調は別のメンバが行うようにカードを配る
- 可能な限り、カードの領域に専門家が充てられるようにカードを配る

などの配慮を行う。

全カードが「役者」に配ばり終えられたら、ファシリテータ（チームリーダ）は、「コンダクタ（ディレクタ）」の役割として、シナリオを1つ選択し、シナリオの第1行目に記述されている動作を読み上げる。以下、選択したシナリオのすべての記述が尽きるまで、次の順序で、「上演」をすすめる。シナリオのすべての記述についての「上演」が完了すれば、更に、別のシナリオを選択し、シナリオが尽きるまで、「上演」を繰り返す。「上演」の経過を記録係が記録する。

シナリオの選択の順序は、

基本的かつ重要な「業務シナリオ」→「主要な例外のシナリオ」→「稀な例外のシナリオ」の順序にする。

（ⅰ）読み上げられた動作を責務とする CRC カードを持っている「役者」が、カードにある責務を読み上げる。

（ⅱ）責務を果たすために協調が必要な場合は、協調先を読み上げ、どのような協調が必要かを協調先の「役者」に言葉で伝える（記録係はどのような言葉がどの「役者」（クラス）からどの「役者」（クラス）に伝えられたかを正確に記述する）。

責務を果たすための協調が必要ない場合は、責務の完了を指示が発せられた「役者」に責務の完了を知らせる（「コンダクタ（ディレクタ）」から指示を受けた場合は、「コンダクタ（ディレクタ）」に責務の完了知らせる）。

（ⅲ）上記（ⅱ）で責務を果たすために協調が必要な場合は、指定された協調先の CRC カードを持っている「役者」が、自分のカードに読み上げられた動作が責務として記載されていることを確かめて、その責務を読み上げる。（その責務にさらに、協調関係がある場合は、（ⅱ）→（ⅲ）の順にネストする。）

（ⅳ）上記（ⅱ）で対応した「役者」の読み上げた責務に、複数の協調が必要な場合は、

協調が尽きるまで、（ⅱ）と（ⅲ）を繰り返す。

この一連の「役者」の交代の流れ（ストーリの流れ）は、記録係によって正確に記述されなければならない。また、誰が演じているか分かるように、何らかの旗とか、帽子

のような目立つ「token」を使用して、演技者にその「token」を受け渡して、ストーリの流れを目に見えるようにし、演じている「役者」のみが立って、他はイスに着席するなどの工夫や、誰がどのカードを持っているかチーム全体が常に把握できるようにするなど、演技が円滑に進行する工夫が必要である。

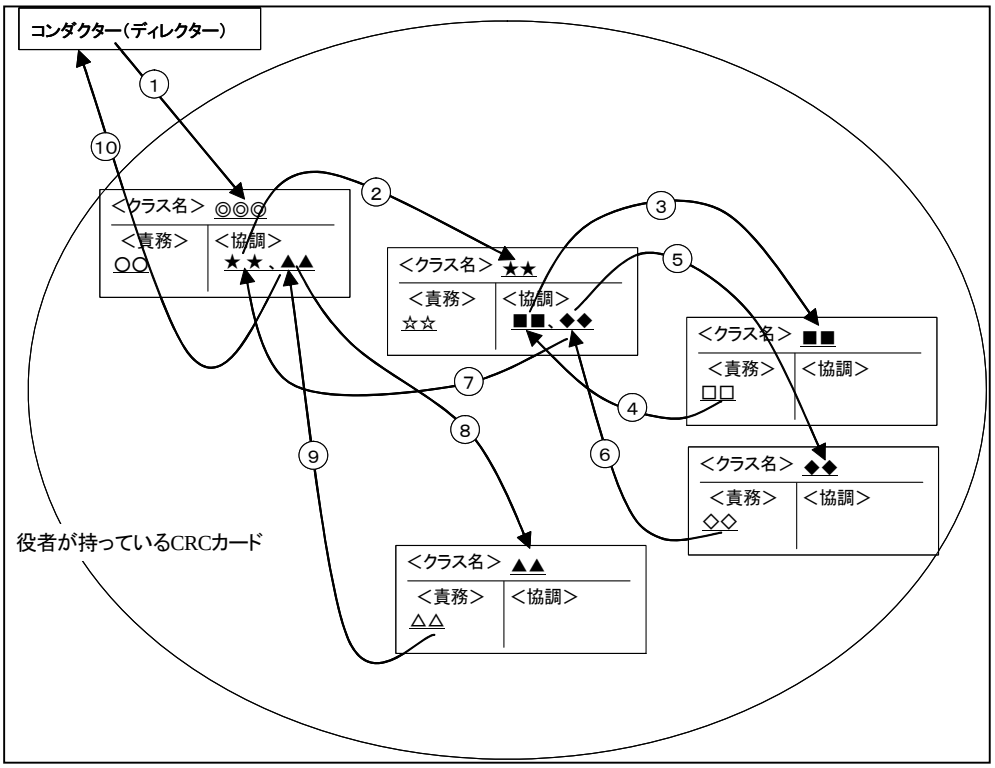


図 2.4-11 協調の関係概念図(ネスト②→③と順次処理(②、⑧)、(③、⑤))

ウォーク・スルーで、責務を果たす「役者」がいない(その責務を記入した CRC カードが無い)場合は、「コンダクタ(ディレクタ)」の判断で、

・ウォーク・スルーを中断して、CRC カードを見直すか、

・不具合の状況を正確に記録係が記録し、とりあえずその責務をスキップして、ウォーク・スルーを継続し、区切りのついたところ(1ストーリが終了した場合か、あるいは、不備が多く、円滑なウォーク・スルーの継続が難しくなった場合など)で、CRC カードを見直す。

出来るだけ曲がりなりにもスピーディーに「上演」リハーサルを続行して、ダレてしまわないように配慮することも必要である。要するに、楽しい雰囲気で、「上演」リハーサルを行うのがコツである。ウォーク・スルーは、全ストーリが、円滑に「上演」されるまで、ウォーク・スルーを継続する。

⑧　最終確認のためのロールプレイ(ラン・スルー)を行い、完成を確認する

ラン・スルーは、最終のいわゆる通し稽古であり、これにより最後の仕上げを行う。ラン・スルーでは、クラス間の通信のみが許され、「オフライン」の会話や、その場での手直しは一切認めないようにする事が重要である。どうしても、ストーリが通らなくなったら、ウォーク・スルーへ戻る事も必要である。

2.4.2.5　CRC カードの利用法についての若干のコメント

上記 2.4.2.4 では、「ビジネスモデル」(自動化、IT化とは関係なく表すモデル)への CRC カードの利用法を説明した。報告者は、「ビジネスモデル」作成と検証(三種類のリハーサルによるシステムの机上シミュレーションによる検証)に当たって CRC カードを利用するのが、最も重要であると考える。しかし、David S. Frankel モデルの分類の、物理モデル、要件モデル、プラットフォーム独立モデル、プラットフォーム依存モデルの作成と検証でも有効に活用できる。事実 2002 に調査した際、業務への IT 活用にあたり、要件モデルとプラットフォーム依存モデルの作成と検証に、2回 CRC カードを採用し効果を上げている企業が存在した。

UML のユースケース図は、モデル化の関係者に理解しやすく、有益であるという人が多いが、CRC カード法のどの時点で記述するかを明確に記述した文献を報告者は知らない。ユースケースの作成については CRC カードを作成する前に、概略のストーリを作成したあとで、ユースケース図を描いて、クラスを抽出するのも良い方法であると考える。CRC カード法と並行して、ユースケース図をブラッシュアップし、CRC カード法のラン・スルーが終わったところで、リハーサルに用いたシナリオをベースとして、ユースケース図を補完するユースケース記述[16]をできるだけ正確に記述して、ユースケースを完成させるのが推奨できるのではないかと思っている。この段階で、ユースケース記述の基本系列(成功のシナリオ)、代替系列(例外、異常時のシナリオ)、契約(事前条件、成功時の事後条件、代替系列の事後条件)などの記述が同じになるクラスが多くなる場合には、汎化(generalization)と継承を行い、スーパクラスに共通記述をまとめて記述量を減らすなどの、冗長性の削除などの工夫が重要になる。またモデルの実現手段を考える上(David S. Frankel のモデルでは、物理モデルの作成)では、コンポーネント(component)[17]の記述が必要になる。世の中の動向が、既存コンポーネントの活用(ソフトウェアの再利用)指向が強まっているので、物理モデル作成の重要性が増大しているが、責務駆動型モデル化には、汎化や、集約／コンポジションは、本質的ではないので、この報告では記述を省略した。CRC カードとしても、Kent Beck and Ward Cunningham のオリジナルのものや、そ

---

[16] ユースケース毎に、ユースケース名、スコープ(位置付け、目的、ゴールなど)、トリガー、アクター、基本系列(成功のシナリオ)、代替系列(例外、異常時のシナリオ)、契約(事前条件、成功時の事後条件、代替系列の事後条件)などを記述したテキスト形態の文書。

[17] コンポーネント(component) : 実装をパッケージにまとめ、一連のインターフェースに適合し、それを実現するもの UML リファレンスマニュアル ピアソン・エデュケーション ISBN コード:4-89471-267-9

の直系の D. Bellin and S. S. Simone のもの、さらには、S.W. Ambler[18]のものには、汎化の記述欄がないが、David A. Taylor が発展させたコンバージェント・エンジニアリングの CRC カードや、N. M. Wilkinson[19]のものには、次頁の2つの図に示される用に、汎化(スーパクラス)の記述欄がある。なお、David A. Taylor の CRC カードのコンポーネントは、上記に述べた実装パッケージをさすものではなく、サブクラス(特化(specialization)の結果)のリストである。

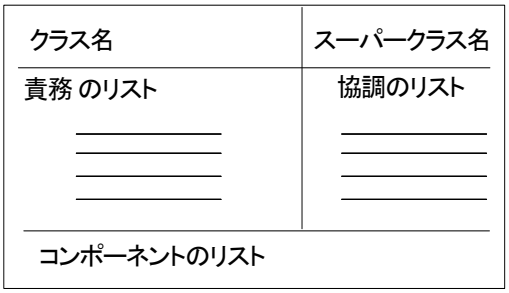


図 2.4-12 David A. Taylor のコンバージェント・エンジニアリングで用いられる CRC カード

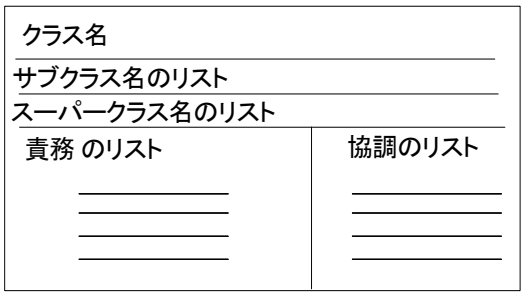


図 2.4-13N. M. Wilkinson が推奨する CRC カード

2.4.2.6 業務システム構築における CRC カード活用の効果

本報告では、ビジネスモデル作成への CRC カード活用を重点に述べてきたが、IT 活用システム構築への展開という局面で、多大な効果を発揮する事が期待できる。その理由は、次の2点にある。

(1)業務情報システム構築にあたり業務精通者と IT 技術者との間で緊密なコンセンサスが得られる。

(2)机上ではあるがロールプレイイングのようなシミュレーションによって動的に検証された業務モデルを持つことができる。

最近、要求仕様化の重要性が再認識されているが、上記(1)は業務精通者と IT 技術者との間誤解やコミュニケーション・ギャップを除く根本的な唯一と言って良い解決方法である。

[18] http://www.ambysoft.com/crcModeling.PDF

[19] N. M. Wilkinson: Using CRC Cards Cambridge Univ. Press 1995 ISBN 0-13-374679-8

従来、次図のように、コンセプトのような情報が極めて乏しい開発の初期段階で、最も重要なテストの計画（受け入れテスト計画）が立てられ、開発の最後で最も重要なテスト（受け入れテスト）が行われる状況を改善されないまま開発が行なわれるのが一般的であり、開発効率、仕様変化対応、情報システムの信頼性等の問題を生じてきた。

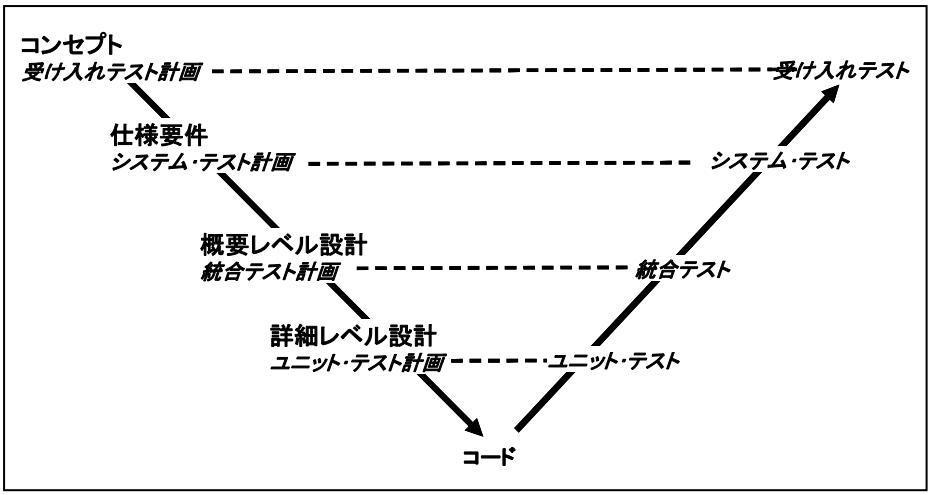


図 2.4-14 開発の最後で最も重要なテストが行われる状況

これに対し CRC カード法を採用すれば、ビジネスモデル作成し、即時にそのモデルを机上ではあるがロールプレイイングのようなシミュレーションによって動的に検証することは、開発初期で、受け入れテスト計画を立て、その実行（受け入れテスト）が開発の最後になるのとでは、質的に、比較にならない効果がある。さらに、今後、期待が大きいモデル駆動開発（MDD：Model-Driven Development）と連携できれば、次図に示す通り、ソフトウェア開発の諸問題の抜本的解決への道が開けるかもしれない。

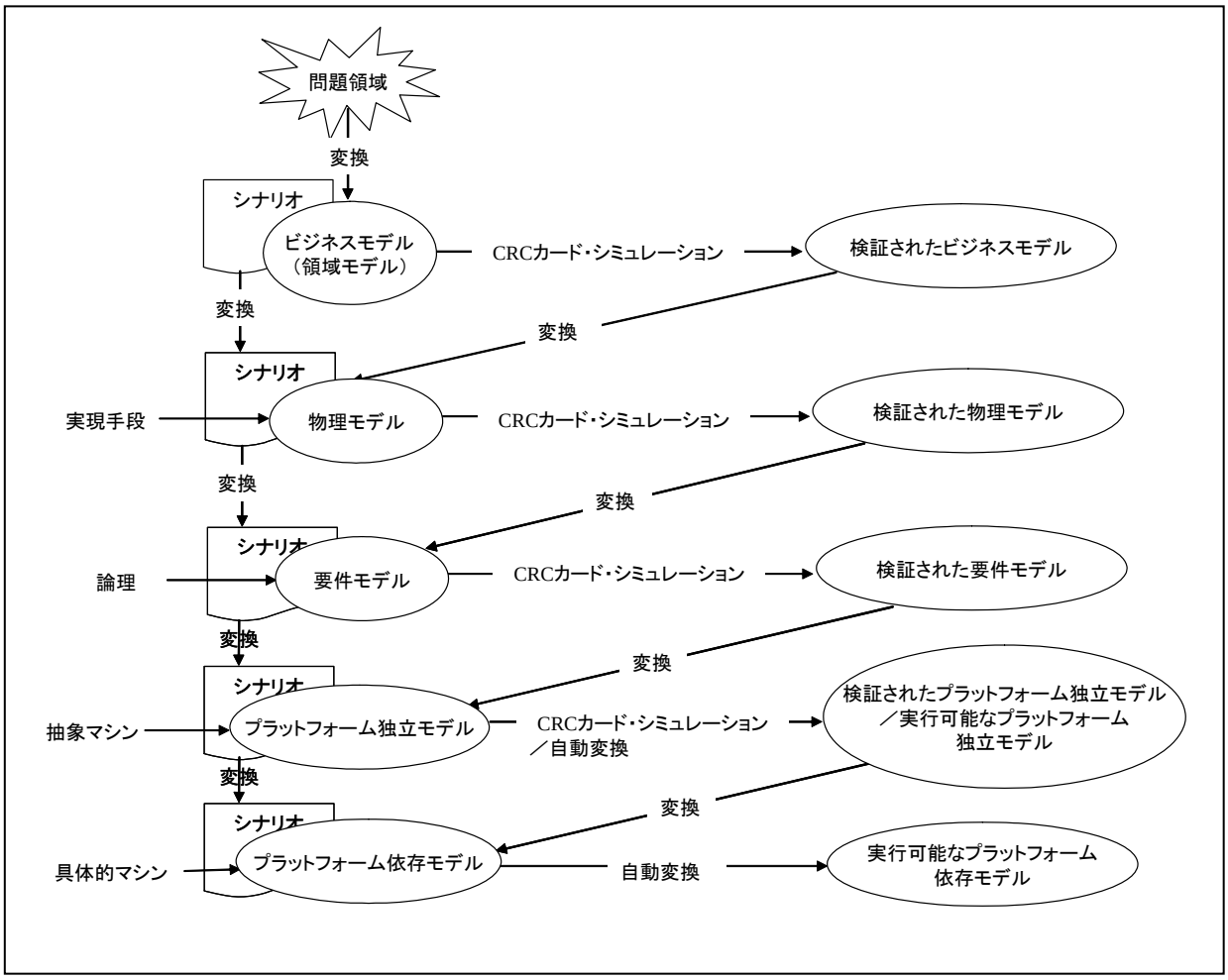


図 2.4-15　CRC カード法とモデル駆動開発（MDD：Model-Driven Development）との連携への期待1

ビジネスモデル、物理モデル、要件モデル、プラットフォーム独立モデル、プラットフォーム依存モデルへと順次、シナリオとモデルが変換される。変換ルールがモデル駆動開発で整備される変換ルール記述法によって機械的に記述できれば、自動変換が可能になる。自動変換が行なわれなくとも、CRC カード法を用いることにより、検証されたモデルが出来ることは、情報システムの信頼性を高めることになる。さらに、次図のように逆変化により、実行可能なモデルの検証を組織的に行えるようになるかもしれない。

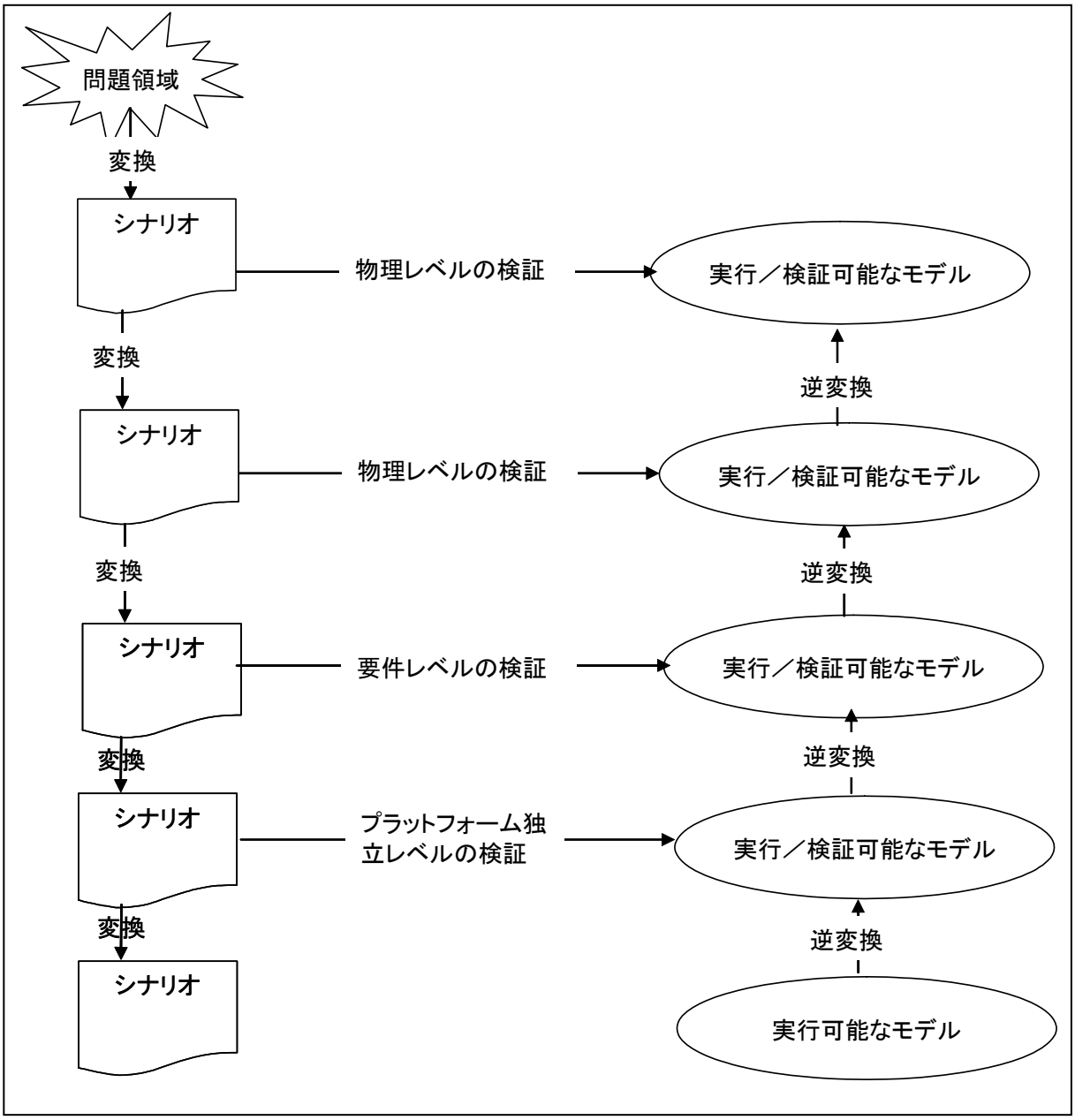


図 2.4-16CRC カード法とモデル駆動開発（MDD： Model-Driven Development）との連携への期待2

2.4.2.7　　BO 研究会における議論

2.4.2.7.1 クラス抽出について（本文 2.4.2.4.2 ⑦参照）

(a)クラスの候補について

BO 研究会では、当初から、責務駆動型モデル化の視点を重視してきた。当然のことながら、BO 研究会メンバ全員、CRCカード法の意義を認めた。CRCカード法の有効性を検証するために、JUASの教育ビジネスを題材に取上げて、よってオブジェクトのクラスを抽出することを試みた。教育ビジネスでは、「講師」と「教材（テーマ、カリキュラムを含む）」などが価値の源泉である。その際、「講師」のようになんらかの責務をもつ主体性のある動作主をオブジェクトのクラスとしてとり上げることは、直ちに BO 研究会の全員の合意を得た。しかし、データ駆動型モデル化では、当然取上げられると思われる「教材」というような動作主が明確ではないものをオブジェクトのクラスに取上げることには異論があった。検討の結果、メンバの一人から、「管理者」という動作主が管理する対象というように考えればどうかという提案があった。議論の結果、本文4（2）（g）に記述した「業務に取って大切で、管理が必要なもの」をオブジェクトのクラスに取上げることで、BO 研究会で合意が得られた。

表面的な動作主のみをオブジェクトのクラスとする場合、本来、代替可能な実装手段のあるオブジェクトのクラスに対して、実装手段（実行者／実行マシン）を固定化し、本来、より良い実装するが考えられる場合に、選択の幅を狭めてしまう危険性があると考える。責務駆動型モデル化の盲点ではないかと考える。BO 研究会行われた議論は、責務駆動型モデル化におけるオブジェクトのクラス抽出において、貴重な知見をえたのではないかと思う。

その後、報告者は、「責務」という言葉が少し限定的であるように感じている、むしろ「役割」という言葉のほうか良いのではないかと感じている。「役割」ととらえれば、上記の教育ビジネスを題材の場合でも、「教材」が教育ビジネスにおける「役割」を考えることが、「責務」という言葉を使用する場合に比べ、自然に「教材」をオブジェクトのクラス候補として採用ができるのではないかと考える。「役割駆動型モデル化」という用語の提案を検討中である。

(b)クラスの抽出について

BO 研究会では、当初から、半ば機械的にオブジェクトのクラス抽出を行う方法を追求してきた。システム化要件を記述した文章（日本語などの自然言語による記述）からのオブジェクトのクラスの候補を機械的に抽出することを試みた。「名詞」を抽出し、オブジェクトのクラスになりうる条件などを検討した。ただし、同様の適用分野（ドメイン）での、オブジェクトのクラスの抽出例が、リファレンス・モデルになっている場合以外、誰が行っても、同じようなオブジェクトのクラスが抽出できるのは難しいという結論に傾きつつある。

2.4.2.7.2 ＣＲＣカード法の大規模システムへの適用可能性

CRC カードの研修会で、CRC カード法は大規模なシステムには活用できないのではないかという質問を受けた。4～10名程度のチームで、1週間（実質5日間）でモデル化を行える範囲は限られている。CRC カード法が提唱している、規模（セッションのメンバ数）や期間を広げるのは、実施上難しいように思われる。最近注目を集めている Agile 情報システム開発方法論[20]の中で、CRC カード法が数多く取り上げられていることを考え合わせても、大規模のモデル化を大規模のまま行う扱う方法ではないと考えて良いように思う。

ただし、大規模なシステムの場合も、テーマを絞り、段階的にモデル化を行うことが、良い結果を生むことはできると考える。例えば、全社規模の経営改革のような場合、経営改革の方向性が出た段階で、その方向性を推進するための経営の「仕組み」作りのためのモデル化を、CRC カード法が提唱している規模と期間で行い、さらに、個々の経営課題について、CRC カード法を活用して、モデル化を行うような推進も有功ではないかと考える。

また、最近、EA(Enterprise Architecture)[21]のような組織全体の包括的なアーキテクチャモデルを階層的に作成し、個々の課題を全体の Architecture に位置付けて、調和を図りながらモデル化を行う方向性が出てきた。いずれにしても、大規模、複雑なシステムを、大規模、複雑なまま扱う方法は、改善される方向にあり、中小規模のモデル化、開発方法論は、将来を見通した方法論と言えるのではないかと考える。

2.4.2.7.3 抽象化について

BO 研究会で、CRC カード法に使用するシナリオが膨大なものになるのではないか、との懸念が聞かれた。例えば、大企業では膨大な業務マニュアルがあり、行政では業務を規定した規則は何万にもなることを考えても、1週間の作業でこなすことなど不可能ではないかという議論があった。この指摘を受けて、IT の技術者は、プログラムの開発量を減らそうとして抽象化という技術を活用するが、業務においては、抽象化ということが一般にあまり行なわれていない事に気が付いた。似たようなマニュアルや、規則集があまりにも多く存在することに気が付いた。業務の抽象モデル化を行うことによって、作成しなければならないマニュアルや、規則集を大幅に減らす事ができるのではないかと考える。CRC カード法でロールプレイを行う際に、似たようなシナリオを繰り返し、行うのは耐え難いであろう。David A. Taylor や、N. M. Wilkinson の CRC カードで汎化（スーパクラス、特化（サブクラス、David A. Taylor では、コンポーネント）が加えられているのは、このような事情からであると考えられる。ただしこれらは、責務駆動型モデル化ということとは別な事を行うので、CRC カード法として

---

[20] (1)http://agilemanifesto.org/, (2)http://c2.com/cgi/wiki?AgileProcesses,
(3) http://www.martinfowler.com/articles/newMethodology.html, (4)http://www.agile-modeling.com/などから情報が得られる。

[21] http://www.gao.gov/special.pubs/eaguide.pdf、
http://www.feapmo.gov/resources/E-Gov_Guidance_Final_Draft_v2.0.pdf
日本政府の取り組みは、http://www.meti.go.jp/kohosys/press/0004840/

手順化するのは難しくなる。本報告では、この部分は省略した。

2.5　ワイン配送センター演習

2.5.1 目的

演習課題のUMLによるモデリング作業を通じて、本委員会の目的である『分析者の経験、考え方に依存しないモデル作成手順』(工学的モデリング)の検討、実証を試みる。

2.5.2 演習課題文書

【演習原文】

ワイン配送センター問題

[1]このワイン配送センターは、ワイナリーからワインを調達し、適切な在庫を保持し注文に応じて小売店はワインを配送することを業務としている。

[2]原則的には、小売店からの注文は翌日配送される。また、適切な在庫水準を保つために、毎夕、ワイナリーへ必要な発注を行う。なお、請求／支払処理、売掛金／買掛金管理等の業務は、対象外とする。

[3]より具体的な業務記述は以下の通り。

[4]当センターは、午前9時から午後5時まで、電話にて小売店からの注文を受け付ける。

[5]小売店からの注文は、図1に示される注文受付書にて受け付ける。

[6]各商品の未引当在庫量が常に把握されていて、小売店からの注文を受け付ける際、各注文商品に対して、十分な未引当在庫量があるかどうかがチェックされる。

[7]十分な未引当在庫量がある場合は、在庫引当が行われ、当該注文商品は、「引当済」台帳に登録される。(「引当済」台帳に登録されたものは翌日配送される。)

[8]十分な未引当在庫量がない場合は小売店がバックオーダーを希望する場合のみ、注文商品は、「入荷待ち」台帳に登録されバックオーダーとなる。(「入荷待ち」台帳に登録されたものは、入荷の翌日に配送される。)

[9]毎日午後5時になると、ワイナリーへの発注処理、翌日の小売店への配送のため配送票作成処理が行われる。

[10]ワイナリーへの発注処理は、未引当在庫量が最低在庫レベルを下回っている商品に対し、未引当在庫量が最大在庫レベルになる数量が発注される。

[11]ワイナリーへの発注書は、図3の通りである。

[12]最低在庫量、最高在庫量は、各商品別に設定されている。

[13]配送票作成処理は、「引当済」台帳に登録された注文商品が集計され、配送数量、配送先を考慮して、各配送車両への割当が行われ、配送車両別、配送先(小売店)別に配送票が作成される。

[14]配送票は図2の通り。

[15]翌朝、各配送車両は、配送票に基づいて倉庫より商品を出庫し、配送する。

［16］各配送票には、配送が完了すると「配送完了」のチェックが付けられる。何らか理由で配送が出来なかった場合は、「再配送用」のチェックがつけられる。

［17］「配送完了」となった注文商品は、「引当済」台帳より、当日午後5時までに削除される。

［18］「再配送要」となった注文商品は、一旦倉庫に戻され、そのまま「引当済」台帳にのこり、再度翌日の配送票に含まれることになる。

［19］ワイナリーからの商品の納入は、午前10時から午後4時までの間に随時行われる。

［20］商品が納入されると、「入荷待ち」台帳に登録された注文商品に対して、ファースト・イン・ファースト・アウトで引き当てられ、この引当ても考慮した上で、当該商品の未引当在庫量が更新される。

［21］「入荷待ち」台帳に登録されていた注文商品は、商品の納入により引き当てられると「引当済」台帳へ登録変更がなされる。

【演習伝票】

注文受付書

注文日付
受注番号
顧客番号
顧客名
電話番号
配送先住所

商品番号	商品名	数量

図１　小売店からの注文受付書

配送票

配送車両番号
配送日付
受注番号
配送結果：　完了・再配送要
顧客名
電話番号
配送先住所

商品番号	商品名	数量

図２　配送票

発注書

発注日付
発注番号
ワイナリー
電話番号

商品番号	商品名	数量

図３　ワイナリーへの発注書

図 2.5-1　伝票

2.5.3 クラス図作成作業

2.5.3.1　　作業内容

・業務の内容を明確にするために、前掲の文章からUMLをつかって概念モデル、業務シナリオを作成し、業務システムの対象領域、記述の曖昧な部分の確認を行う。
・業務シナリオから名詞に着目しクラスを抽出する、
・クラス間の関係を明らかにする。
・結果として下記UMLを作成する。
　①ユースケース図　　図 2.5-2
　②ユースケース記述　 表 2.5-1,2
　③シーケンス図　　図 2.5-9
　④クラス図　　図 2.5-5
・作業過程の理解を得るため下記のワークシートを添付する。
　⑤クラスの抽出作業表　 表 2.5-3
　⑥提起された疑問点と見解　表 2.5-5

2.5.3.2　　概念モデル、シナリオの作成

実際の作業ではユーザの要件定義をもとに、現状の概念モデルを確認するためユースケース図を作成する工程に相当する。ユーザと会話しながら作成できるのが理想だが、実際にはあらゆる手段を用いて要件を聞き出し、ユースケースに整理して確認するという作業になるのではないだろうか。

2.5.3.2.1 原始シナリオの整理

今回は演習の原文をヒアリング録として作業を進める。
原文を整理していくと、実業務ではユーザに確認する作業が発生するようないくつか曖昧な点が浮かび上がる。ここでは、それぞれ次の様に解釈する。

①　数が不明
　倉庫　→　1つとする。
　ワイナリー　→　複数。
　　注）単数、複数の曖昧さは日本語特有の問題で常に意識しておく。

②　曖昧な記述
　・「商品」とはワインを指すのかそれ以外の品物もあるのか　→　ワインのみ。
　・注文の受付は一カ所か（電話は一本？）→　一カ所と考え受付係を想定する。
　・配送票の作成、ワイナリーへの発注は誰が行うか　→　倉庫係を想定する。
　・配送先は小売店の所在と同じか　→　同じとする。
　・未引当在庫量はいつ更新するのか
　　→　小売店からの注文により引当てされる毎、納入（ワイナリーからの）、入荷待ち

の引当時にタイムリーに更新されている。

・配送、配送完了、再送要等更新は誰が行うのか → **配送係**を想定する。

・複数商品の注文があった場合に部分出荷するか → する。

・同一注文書の商品を数量分割して出荷するか → しない。

・ワイナリーへの発注数量を正確に定義する。

→未振当在庫量＋発注済未納入数量―入荷待数量が最低在庫レベル以下の商品の発注を行う。発注数は最大在庫レベル―(未振当在庫量＋発注済未納入数量)＋入荷待数量

2.5.3.2.2 用語の整理統一

クラスの切り出し作業を行うには用語は事前整理しておく必要がある。演習でも、注文、発注、納品、配送、他、販売、仕入れどちらにも使われる用語が存在する。

本演習の用語は次のように定義する。

注文:小売店から当センターへの注文(販売)

発注:当センターからワイナリーへの発注(仕入)

納品:当センターへのワイナリーからの納入(仕入)

配送:当センターから小売店への配送(販売)

2.5.3.2.3 原始シナリオの修正

以上の整理作業から演習の原文を訂正する。訂正は概略記述の[1]、[2]は下記。以降はユースケース記述を参照のこと。

原始シナリオの修正(概略記述)　変更部分

[1]この配送センターは、複数のワイナリーからワインを調達し、倉庫(一ヵ所)に適切な在庫を保持し小売店からの注文に応じてワインを配送することを業務としている。

[2]原則的には、小売店からの注文は翌日配送される。また、適切な在庫水準を保つために、毎夕、ワイナリーへ必要な発注を行う。なお、当センターの商品はワインのみとし、請求／支払処理、売掛金／買掛金管理等の業務は、対象外とする。

2.5.3.2.4 階層化 (粒度の検討)

ユースケース図はユーザが直感的に業務要件を確認できるための記述である。従って解りやすいことに主眼をおくべきである。

そのため、下記を念頭にユースケース図を作成する。

・粒度を揃え、必要に応じて階層化を図る。

・例外事項は図から外しユースケース記述に記載する。

・実装を意識しない。

・解り安さに配慮し、機能の並びをある程度意識する。

演習では全体の業務フローを述べている[1]、[2]をユースケース図で表し、[3]以下の詳細説明をユースケース記述とする方針で作業する。

作成されたユースケー図を図 2.5-2、ユースケース記述表 2.5-1,2 を表に示す。

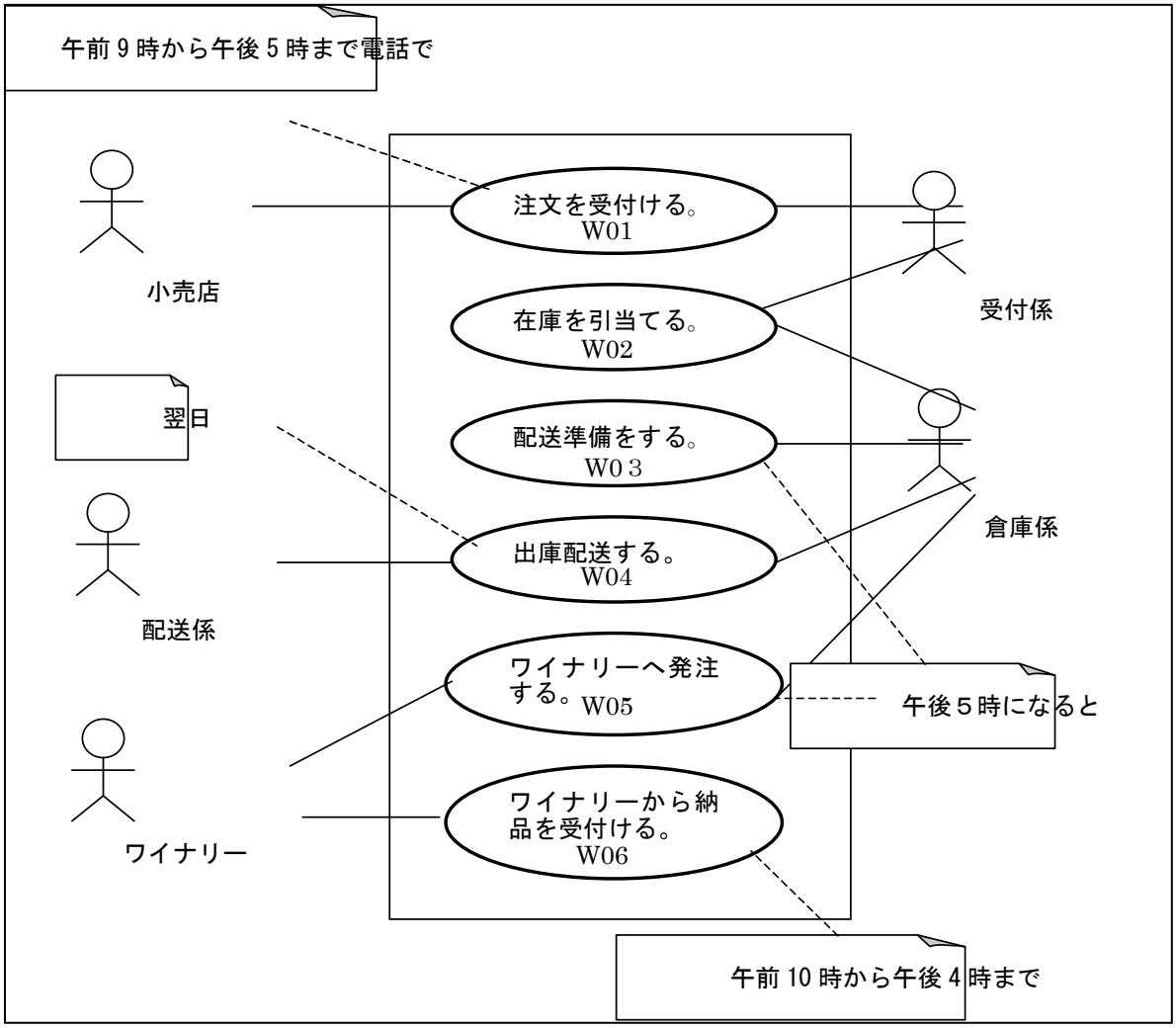


図 2.5-2　ワイナリー配送センターユースケース図

表 2.5-1 ユースケース記述（1／2） 下線部分は最初に出現した名詞、名詞句。 は原始シナリオの修正部分。

ユースケース名	W01	W02	W03
概要	注文を受付ける	在庫を引き当てる	配送準備をする
アクター	小売店、受付係	受付係	倉庫係
事前条件		・未引当在庫量を把握できる。 ・注文受付書が起票されている。	・引当済台帳の本日分登録が完了している。
業務要件	[4]当センターは、午前9時から午後5時まで、受付係（一ヵ所）が電話にて小売店からの注文を受け付ける。 [5]小売店からの注文は、図1に示される注文受付書にて受け付ける。 [6]各商品の未引当在庫量が~~常に~~在庫引当、商品納入があるたびタイムリーに更新され把握されていて、小売店からの注文を受け付ける際、各注文商品に対して、十分な未引当在庫量があうかどうかがチェックされる	[7]十分な未引当在庫量がある場合は、在庫引当てが行われ、当該注文商品は、「引当済」台帳に登録される。（「引当済」台帳に登録されたものは翌日配送される。） [8]十分な未引当在庫量がない場合は小売店がバックオーダーを希望する場合のみ、注文商品は、「入荷待ち」台帳に登録されバックオーダーとなる。（「入荷待ち」台帳に登録されたものは、入荷の翌日に配送される。 [8－1]注文受付書の商品は、引当済台帳又は入荷待ち台帳に登録され、以降の作業は商品単位で行われる。 [8－2]商品の注文数量を分割して小売店に配送することはない。	[9]毎日午後5時になると、倉庫係が（ワイナリーへの発注処理、）翌日の小売店への配送のため配送票作成処理が行われる。 [13]配送票作成処理は、「引当済」台帳に登録された注文商品が集計され、配送数量、配送先を考慮して、各配送車両への割当が行われ、配送車両別、配送先（小売店）別に配送票が作成される。 [14]配送票は図2の通り。
事後条件	・注文品の注文受付書が起票される。	・商品が引当済台帳に登録される。	・配送票が作成される。

		・入荷待ち台帳に登録される。 ・未引当在庫量が更新される。	

表 2.5-2 ユースケース記述(2／2) 下線部分は最初に出現した名詞、名詞句。 　 は原始シナリオの修正部分。

ユースケース名	W04	W05	W06
概要	出庫配送する。	ワイナリーへ発注する	ワイナリーから納品を受ける
アクター	倉庫係、配送係	倉庫係、ワイナリー	倉庫係、ワイナリー
事前条件	・配送票が作成されている。	・未引当在庫量が確定している。	・ワイナリーから納品がある。 ・入荷待ち台帳が更新されている。
業務要件	[15]翌朝、配送係は配送車両毎の配送票に基づいて倉庫より商品を出庫し、配送する。 [16]各配送票には、配送が完了すると「配送完了」のチェックが付けられる。何らかの理由で配送ができなかった場合は、「再配送要」のチェックが付けられる。 [17]「配送完了」となった注文商品は、「引当済」台帳より、当日午後5時までに削除される。 [18]「再配送要」となった注文商品は、一旦倉庫に戻され、そのまま「引当済」台帳にのこり、再度翌日の配送票に含まれ	[9]毎日午後5時になると、ワイナリーへの発注処理(、翌日の小売店への配送のため配送票作成処理)が行われる。 [10] ワイナリーへの発注処理は、入荷待ちを解消して、最大在庫レベルになるように行われる。即ち、 未引当在庫量＋発注済未納入数量 ―入荷待ち数量 が最低在庫レベルを下回っている商品の発注を行い、発注数は 最大在庫レベル―(未引当在庫量＋発注済未納入数量)＋入荷待ち数量	[19]ワイナリーからの商品の納入は、午前10時から午後4時までの間に随時行われる。 [20]商品が納入されると、「入荷待ち」台帳に登録された注文商品に対して、ファースト・イン・ファースト・アウトで引き当てられ、この引当も考慮した上で、当該商品の未引当在庫量が更新される。 [21]「入荷待ち」台帳に登録されていた注文商品は、商品の納入により引き当てられると「引当済」台帳へ登録変更がなされる。

	ることになる。	とする。 [11]ワイナリーへの発注書は、図3の通りである。 [12]最低在庫量、最高在庫量は、各商品別に設定されている。	
事後条件	・配送完了がチェックされる。 ・再配送品が引当済台帳に登録される。	・ワイナリーへの発注が行われる。	・入荷待ちで引当可能な商品が引当済台帳に登録される ・未引当在庫量が更新される。

2.5.3.3　　クラスの切り出しからシーケンス図、クラス図の作成

2.5.3.3.1 クラスの抽出

ユースケース記述からクラス候補として名詞、名詞句（以下名詞と記す）を抽出し（ユースケース記述の下線部分）、次の手順でクラス候補を絞り込んでいく。

名詞の絞り込み過程を表 2.5-3 クラス抽出作業表に示す。

① 抽出条件による絞り込み

抽出条件

- 状態を表すものを除く

  例　配送完了、再配送要
- 動作の状態を再定義したものは除く

  例えば　配送票作成処理は[13]で『配送票作成処理は・・・』とその内容が説明されておりこの名詞はクラス対象から除外する。
- 数値項目、時間を表すものは除く

  例　日時、数量
- 固有名詞はクラスのインスタンスである可能性があるので残す。

② 同義語を統一する。

例　ワインと商品　配送先と小売店

③ 管理対象にならないものを除外する。

例えば、この配送センターでは配送車両は車両番号を取り配送のルートなどを考慮して管理されておりクラス候補といえる。車両が1台しかない又は運送会社に委託しているようなケースで、特に管理する必要がない場合はクラス候補にはならない。

④ 汎化の考え方によるクラスの整理

固有名詞で抽出されたクラスを一般名詞におきかえる。さらに、①～③で抽出されたクラス候補を汎化することでスーパクラスを設ける。または、属性調整により1クラスとして統合するなどクラスの整理を行う。

例えば、小売店とワイナリーを取引先というスーパクラスとサブクラスにする。

あるいは、属性として取引先分類（小売店orワイナリー）を設けることで取引先クラスとして統合することを検討する。

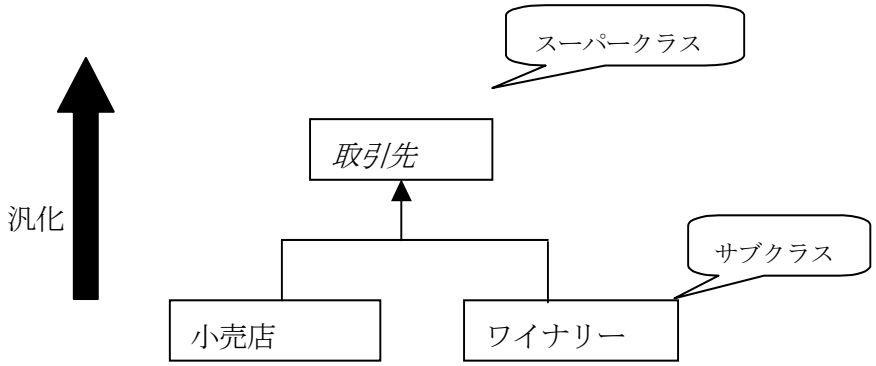


図 2.5-3

⑤　集約関係に着目することによるクラスの分解

注文受付書、配送票、発注書はそれぞれ商品の明細をもっている。このような集約関係に着目し、別クラスに分解する。

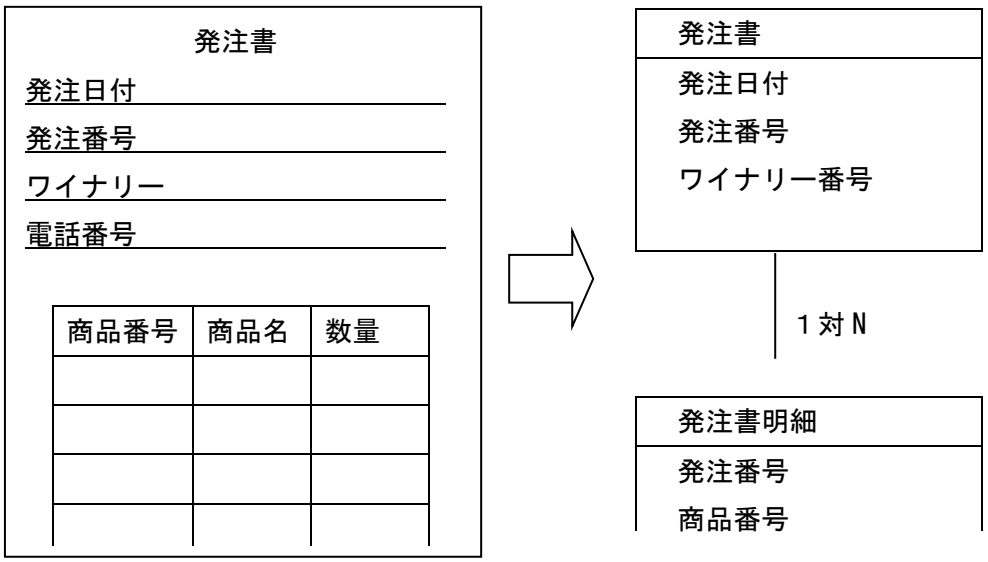


図 2.5-4

注文受付書と配送票クラスにも明細があるが、それぞれ下記の理由でクラスを設けない。

①　注文受付書

・注文商品は引当済台帳または入荷待ち台帳に注文受付時に記入されると解釈する。

・注文受付書の商品明細クラスは引当済台帳と入荷待台帳の商品から構成される。

②　配送表

・配送対象の商品明細は引当済台帳の商品明細から抽出される。

・配送票は抽出された商品明細を配送単位にくくりなおしたもと考える。

表 2.5-3 クラスの抽出作業表

ユースケース	名詞、名詞句	クラス	判定内容	備考
W01	当センター		管理対象外	
	午前9時		数値項目、時間	
	午後5時		数値項目、時間	
	受付係		管理対象外	
	一ヵ所		数値項目、時間	
	電話		管理対象外	手段
	小売店	○		
	注文		動作の再定義	[5]注文受付書にて受付
	注文受付書	○		管理者:受付係
	在庫引当		動作の再定義	[7]引当済台帳に登録
	商品納入		動作の再定義	[20]商品が納入・・・
	各商品		同義語	商品
	未引当在庫量		状態	管理者:倉庫係、受付係
	各注文商品		同義語	商品
W02	当該注文商品		同義語	商品
	引当済台帳	○		管理者:受付係
	翌日		数値項目、時間	
	バックオーダー		動作の再定義	入荷待ちの意味
	注文商品		同義語	商品
	入荷待ち台帳	○		管理者:受付係、倉庫係
	入荷		動作の再定義	[20]商品が納入・・・
	商品	○		管理者:倉庫係
	商品単位		同義語	商品
	注文数量		数値項目、時間	
W03	倉庫係		管理対象外	
	配送票作成処理		動作の再定義	[12]配送票作成処理は・・・
	配送数量		数値項目、時間	
	配送先		同義語	小売店
	各配送車両	○	(クラス名は配送車両とする)	管理者:配送係

	配送票	○		管理者:倉庫係、配送係
W04	翌朝		数値項目、時間	
	配送係		管理対象外	
	倉庫		管理対象外	
	配送完了		状態	
	再配送要		状態	
W05	毎日		数値項目、時間	毎週、毎月等繰り返しの時間間隔
	ワイナリー	○		
	発注処理		動作の再定義	[10]ワイナリーへの発注処理は・・・
	最大在庫レベル		状態	
	発注済未納入数量		数値項目、時間	
	入荷待数量		数値項目、時間	
	最低在庫レベル		状態	
	発注		動作再定義	[10]ワイナリーへの発注処理は・・
	発注数		数値項目、時間	
	発注書	○		管理者:倉庫係
W06	納入		動作の再定義	[20]商品が納入されると・・・
	午前10時		数値項目、時間	
	午後4時		数値項目、時間	
	ファースト・イン・ファーストアウト		状態	注文の早いものから振当てがおこなわれるという要件の説明に置換え可
	引当		動作の再定義	[21]引当済台帳への登録変更
	登録変更		動作の再定義	引当済台帳への登録のことを言っている

2.5.4 クラス図の作成　（図 2.5-5）

抽出されたクラス間の関連を定義する。

実際のモデリングではクラスの切り出しからクラス図まで手戻りを繰り返しながらの作業になる。なお、関連の多重度の記述は下表による。

記号(例)	多重度の意味
1	厳密に1
*	複数
0..*	0以上
0..1	0or1
1..*	1以

2. . 5	2～5
1,3,5	1or3or5

表 2.5-4

2.5.5 シーケンス図の作成　（図 2.5-16）最終頁別紙

抽出されたクラス間の相互作用をユースケース記述に従って時系列で記述し、シーケンス図を作成する。この作業を通じて各クラスの持つべき機能（責務）が明らかになる。

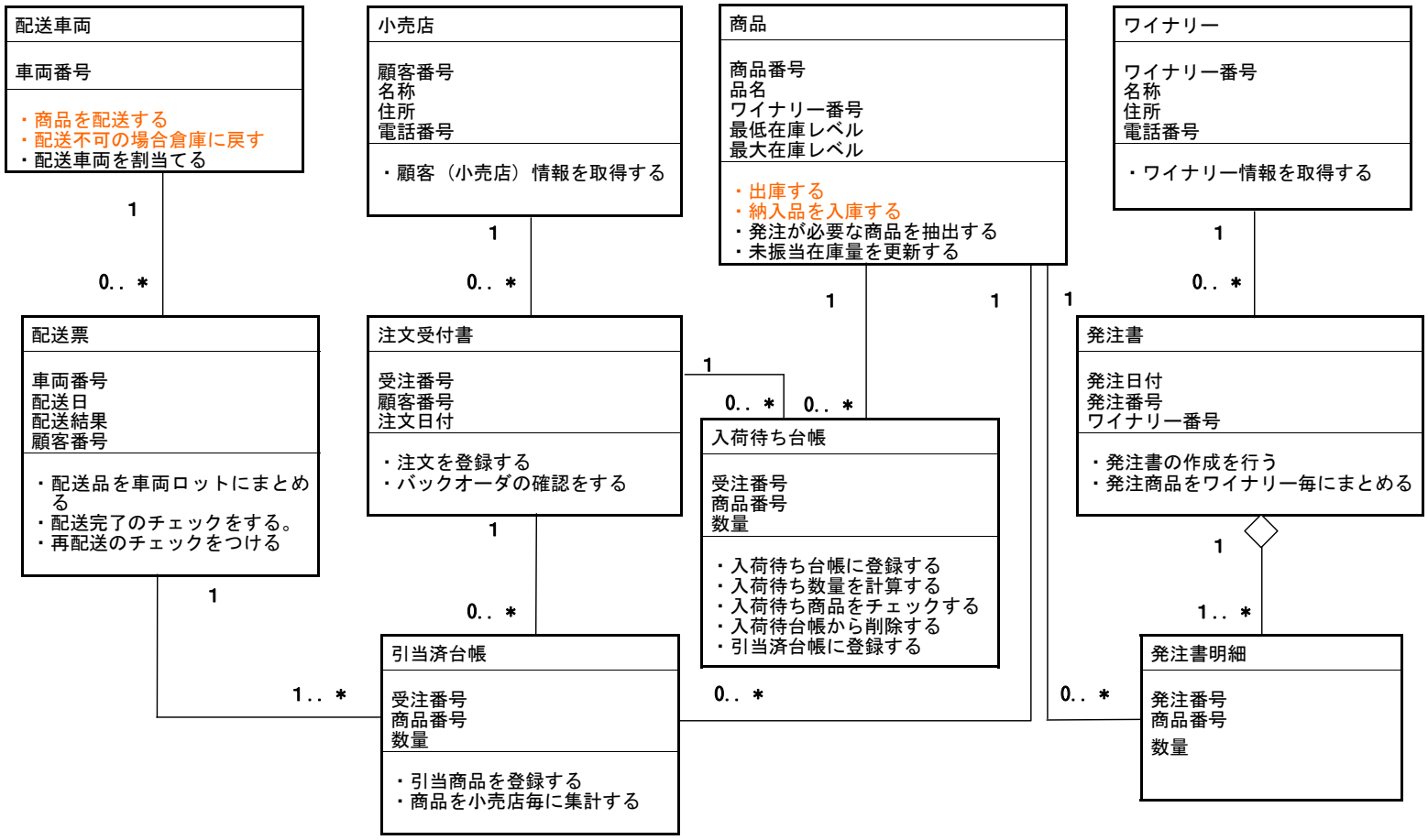

配送車両
車両番号
・商品を配送する
・配送不可の場合倉庫に戻す
・配送車両を割当てる
小売店
顧客番号
名称
住所
電話番号
・顧客（小売店）情報を取得する
商品
商品番号
品名
ワイナリー番号
最低在庫レベル
最大在庫レベル
・出庫する
・納入品を入庫する
・発注が必要な商品を抽出する
・未振当在庫量を更新する
ワイナリー
ワイナリー番号
名称
住所
電話番号
・ワイナリー情報を取得する
配送票
車両番号
配送日
配送結果
顧客番号
・配送品を車両ロットにまとめる
・配送完了のチェックをする。
・再配送のチェックをつける
注文受付書
受注番号
顧客番号
注文日付
・注文を登録する
・バックオーダの確認をする
入荷待ち台帳
受注番号
商品番号
数量
・入荷待ち台帳に登録する
・入荷待ち数量を計算する
・入荷待ち商品をチェックする
・入荷待台帳から削除する
・引当済台帳に登録する
発注書
発注日付
発注番号
ワイナリー番号
・発注書の作成を行う
・発注商品をワイナリー毎にまとめる
引当済台帳
受注番号
商品番号
数量
・引当商品を登録する
・商品を小売店毎に集計する
発注書明細
発注番号
商品番号
数量
1
0.. *
1..*

図 2.5-5　クラス図

2.5.6 演習の作業過程で提起された疑問点と見解

表 2.5-5

分類	疑問点・検討事項	当委員会の考えかた
クラスの抽出	・本作業で言う名詞、名詞句とは何か。	・固有名詞、抽象名詞が対象 ・『独立語で活用がない』、『助詞がつく』という特徴がある。
	・名詞を抽出することに意味があるのか、管理対象を抽出する必要があるのではないか。	・シナリオから機械的作業でクラス候補を抽出する為の最初の作業と捉える。 ・抽出されたクラス候補からさらにクラスに絞りこむために抽出条件を整理する。(3．3．1項参照)
	・管理対象とはなにか。	・対象となる業務に重要な概念 ・状態があるもの ・「手段」は対象外　例　電話 例えば配送車両は車両番号、積載量等管理を行っていれば管理対象になり、専門の業者に委託し、業務外で管理される場合は対象外となる。
	・数値、時間を表すものはなぜクラス候補とならないか。	・数値、時間は振る舞いを持たずクラスにはなり得ない。(数値項目はクラスの属性候補) ・オブジェクトを共通の特性で括ったものがクラスで、数値項目、時間は何らかのクラスが実体化した状態を表しており、その数値項目を属性とするクラスが存在する。(但し、そのクラスが管理対象であるかは別の検討が必要) 例)　東京　北の風　5m／s、 千葉　北東の風　2m／s はオブジェクトで共通特性として風力というクラスと地域、風向、風速という属性が浮かびあがる。
	・ワイナリー、倉庫は単数か複数かをどう解釈するか。	・日本語特有の問題である。 ・常に単数、複数を意識して読む(記述する)ことが必要。 ・現在、将来の業務イメージを推定するには業務知識をもった人が行う必要がある。

	・誰がやっても同じクラスを抽出できるか。	・業務シナリオからクラスを抽出する作業は今回の作業で標準化が出てきた。 ・対象業務を正確に記述したシナリオを書くことが「誰がやっても・・・」に通ずる。
作業方法	・新規業務でフロー、伝票がない段階で概念モデルを作成する場合はどういう作業になるか。	・最終的には業務シナリオが作成できればよい。 ・業務フローは要件定義の中心となるが、使用する帳票などのイメージを作成することは、業務の詳細を確定する意味でも重要な作業であり、モデル作成の作業の一部となる。
	・業務フロー（図）は必要か、UMLのどの図に相当するか。	・UMLの9種類の図には従来型の業務フローは出てこないが、ユーザとの業務シナリオの確認には有効で、実務的には必要と思われる。 （アクティビティ図が近い）
	・クラスをまとめる作業（汎化）は何時やるか。	・抽出したクラスが機能すればよいのでこの段階ではやる必要がない。
	・例外処理（追加、修正、削除）の確認はどのタイミングでやって、どこまで記述するか。	・例外処理には「業務・作業」的な面と「システムアーキテクチャ」的な面の2種類ある。 ・業務・作業面では ・ユースケースの段階で機能毎にチェックする。 ・ユースケース図に書くかどうかは粒度、ユーザ要求によるが、基本的にはユースケース図はメインの機能のみとし、ユースケース記述にするとわかりやすい。 ・システムアーキテクチャ面では イベントシーケンス図から記述する。
ER図との比較	・クラスとエンティティの違いがよくわからない。 ・エンティティとクラスは同じか	・エンティティには振る舞いが記述されない。 ・クラスはDBにマッピングされるとは限らない。 従ってエンティティにならないクラスがあり エンティティはクラスの一部である。 ・オブジェクト指向とは 「人間が物を認識する方法」で、それは次の3つの視点によるといわれている。 ①機能（どんな働きがあるのか） ②分類（何に似ているか）

		③構造（何によって構成されているか） 従来型の方法論である構造化設計では機能の詳細を表現し、ERDでは（データ）構造を表現しているが、いずれも上記の一面しかとらえていない。

2.5.7 ワイン配送センター演習を DOA で考える

オブジェクト指向とDOAのモデリング手法の比較については多くの専門家によって様々な研究がおこなわれている。ここではクラスの切り出しからクラス図の作成の作業を行うに当って一般のモデラーがER図をどのような位置付けで捉えるべきかを考察してみる。

ワイン配送センターの問題から作成されたER図を図 2.5-6 に示す。この演習ではクラスとエンティティは全く同じになった。ER図では実装のデータベースを意識するためアイデンティファイアとして引当 NO、配送 NO 等を設けたがそれ以外では差はない。表記上はクラス図が ER を参考にして考案されたものであり細かい記法の違いを除けばかなり類似している。

大きな違いはクラス図にはデータとともに振る舞い（メッソッド）が記述されていることである。つまりER図はモデルのデータの関係を明らかにしようとしているのに対し、クラス図はモデル全体を表現しようとしており、注目している範囲に違いがある。

従ってER図として実体が表現されない（データベースにマッピングされない）オブジェクトがクラス図では表現される可能性がある。

例えば、演習のワイン配送センターに、配送車両のルートを印刷する機能があったとする。クラス図には配送先の住所、道路情報からルートを計算する**配送ルート図作成**クラスが登場するが、ER図ではこの計算部分が何らかの形でデータ表現されない限り出現しない。

モデリングに当って、モデルのデータ構造を明らかにする視点でER図を用いるという姿勢でよいのではないだろうか。但し、エンティティはクラスになる可能性が高いがそれが全てではないのでクラスの抽出には十分ではないことに留意が必要である。

参考）オブジェクト指向技術が、これまでのソフトウェア工学で構築された様々な方法論を活用しながら、新しいソフトウェア構築の方法論として登場したのは、ネットワークやデータベースなどが分散処理環境の充実を促し、従来型の方法論では、これらの環境に十分適用できなくなった背景がある。

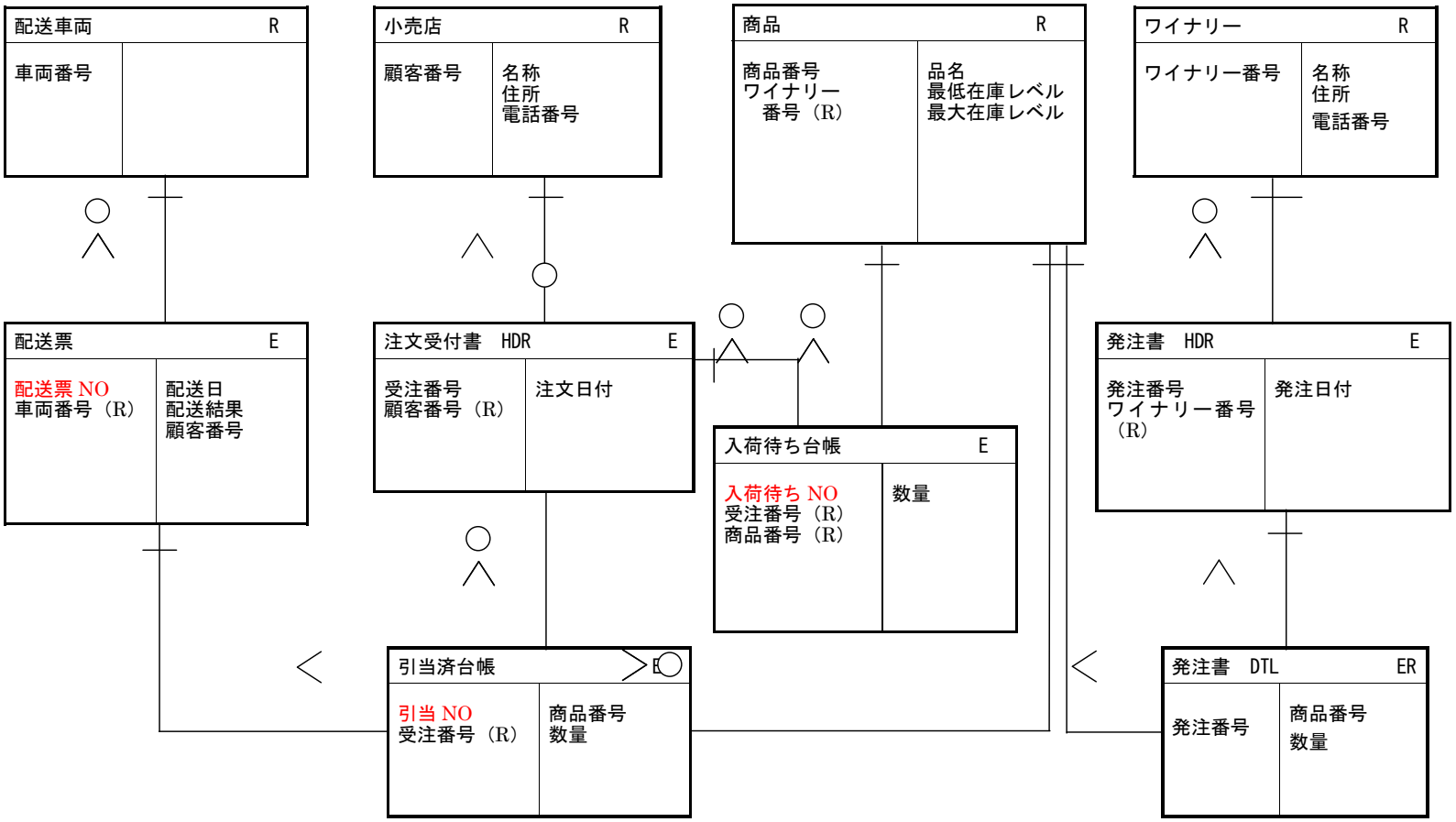


図 2.5-6　ワイン配送センターＥＲ図

2.5.8 モデル作成作業の実際

ＵＭＬは概念モデルから実装まで全工程をカバーすることができるといわれる。本稿では概念モデルの構築について演習を通じて手順を検討した。

既存システムを何らかの目的で再構築、改訂する場合、実務上で発生するモデルの作成作業は下図のようになる。

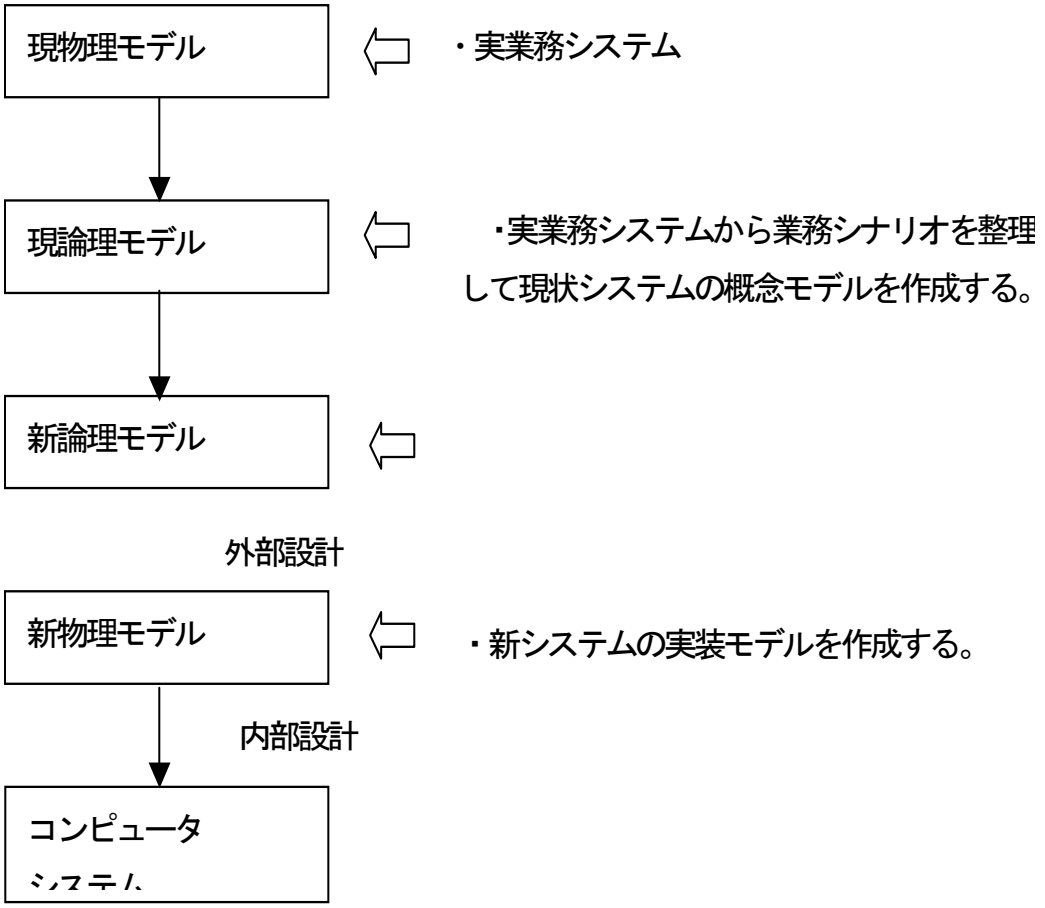


図 2.5-7

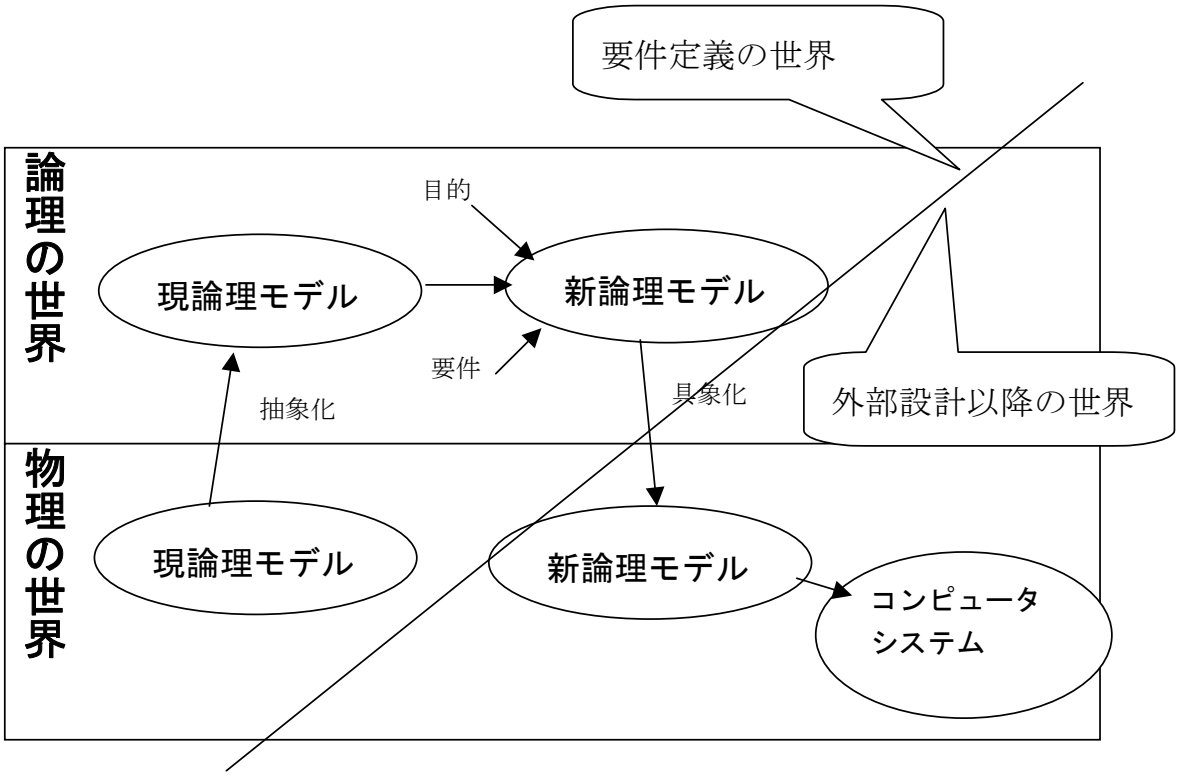

要件定義の世界
論理の世界
目的
現論理モデル
新論理モデル
要件
抽象化
具象化
外部設計以降の世界
物理の世界
現論理モデル
新論理モデル
コンピュータ
システム

図 2.5-8

2.5.9 ビジネス・プロセス・モデルとしてのオブジェクト指向モデル

UMLに代表されるように、オブジェクト指向モデルは、情報システムの仕様を表すモデルとしては、非常に成功を収めている。一方で、オブジェクト指向は、本来は、「実世界をありのままにモデル化する」手法であるとも言われ、その意味では、情報システムの仕様に留まらず、業務そのものの仕様を表すのに適したモデルでもあるはずだ。しかし現実には、UMLによるビジネス・プロセス・モデルや、エンタープライズ・モデル等が提唱されているにもかかわらず、成功を収めているとは言い難い。そこで、UMLに囚われることなく、最適の業務設計のためのビジネス・プロセス・モデルとしてのオブジェクト指向モデルについて考察してみる。

このような、位置づけのものとして、最もポピュラーなものは、おそらくIDEF0であろう。IDEF0は図 2.5-9 に示されるように、業務機能をアクティビティとして切り出し、それらを、入力、出力、制約、機構で関連づけたものである。IDEF0においても、他の多くのモデルと同様に、何を、アクティビティ、入力、出力、制約、及び機構として捉えるかは、必ずしも明確ではないように思われるが、各アクティビティが、制約に従い、機構を利用して、ある入力に対して、ある出力を生成しているという意味では、入力イベントに対して出力イベントを生成するオブジェクト指向におけるオブジェクトと相通じるものがある。そこで、非常に簡略化したワイン配送センター問題を例題に、制約及び機構を省略したIDEF0に近い表記を出発点として考えることにする。

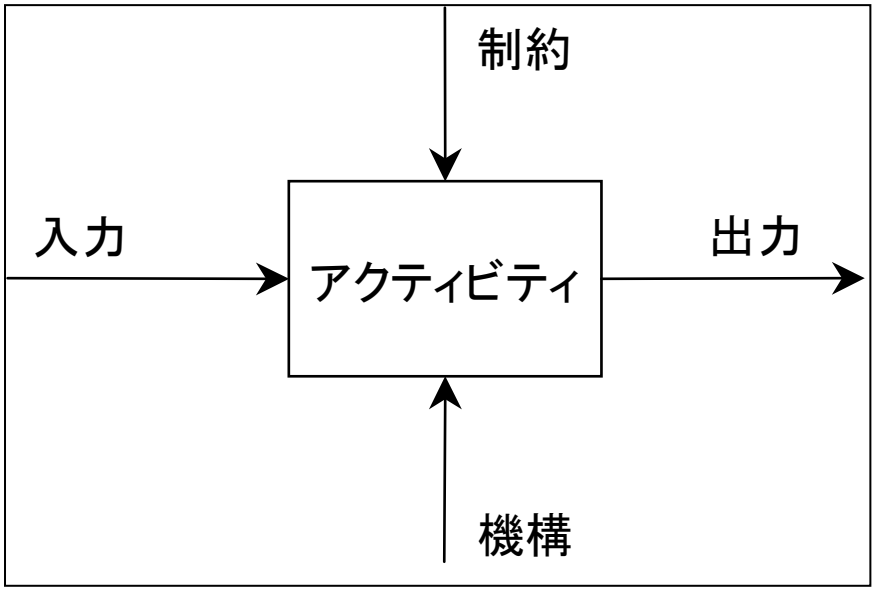


図 2.5-9　IDEF0 の概要

図 2.5-10 は、ワイン配送センターの機能から、「ワイン配送在庫管理機能」と「小売店への配送トラック(の機能)」を2つのアクティビティとして切り出し、情報の流れと物の流れをアクティビティに対する入力・出力として表したものである。なお、金の流れは、簡単化のために省略した。

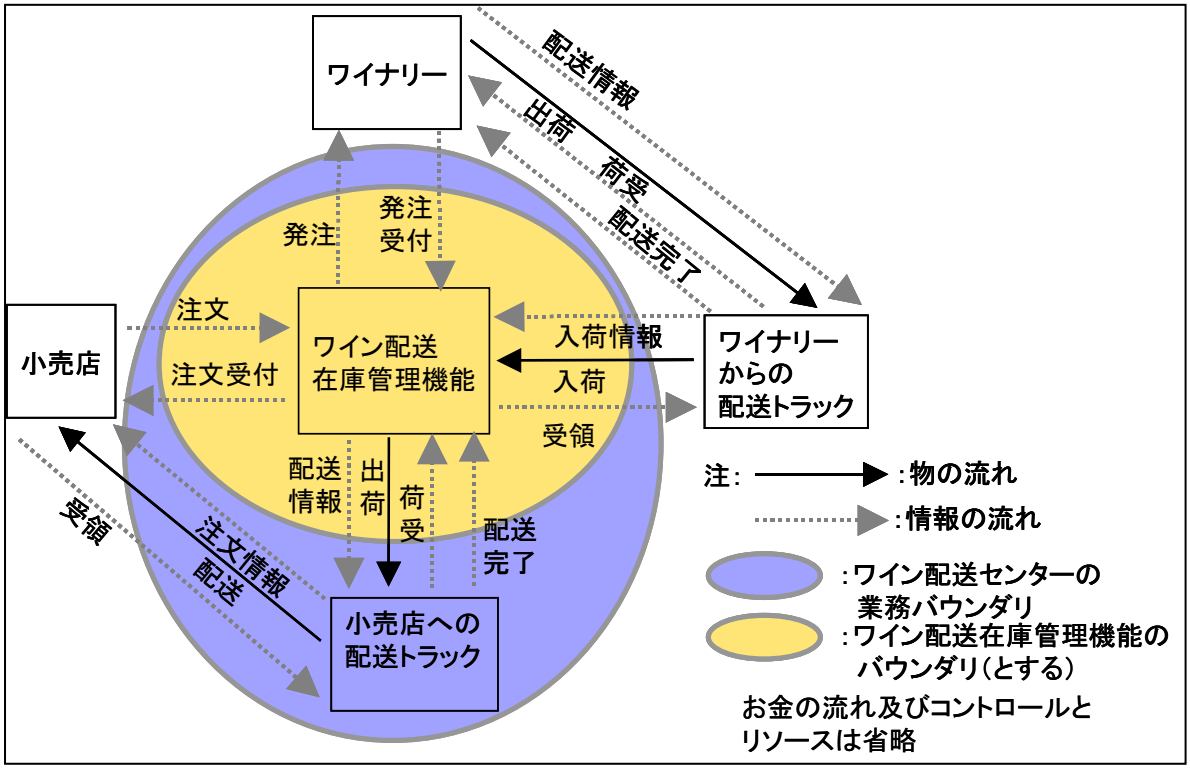


図 2.5-10　検討素材として簡略化したモデル

この段階において検討すべきことは、アクティビティの切り出しが、経営環境・目標に照らして、業務機能の分割として妥当かどうかという点である。例えば、「ワイン配送在庫管理機能」は、小売店からの注文に対する配送という重要な責務を配送トラックに委譲している。これは、お客さまに対する最も基本的な責務を自己完結的に全うしていないという意味で、一般的には、あまり良いアクティビティの切り出しとは言えないかもしれない。ただし、経営環境・目標が求める第一のものがコストダウンである場合には、そのために、実際のお客さまへの配送を外部の配送会社にアウトソースすることも考えられ、その意味では、このようなアクティビティの切り出しもあり得る。

ここでは、検討の結果、図 2.5-2 のような、アクティビティの切り出しになったという前提のもとで、「ワイン配送管理機能」のあるべき姿を導き出すためのビジネス・プロセス・モデルについて考えることにする。即ち、小売店からの注文に応じて小売店への配送トラックに出荷し、そのために必要に応じてワイナリーに発注し、ワイナリーからの配送トラックから入荷を受けることが「ワイン配送在庫管理機能」責務であることを前提に、その責務を果たすのに最適な業務設計を行うことを考える。その場合、アクティビティ「ワイン配送在庫管理機能」の入力と出力として

は、何を考え、何を検討するべきであろうか。

図 2.5-11 は、単純に図 2.5-10 の内、関連したすべての物および情報の流れをアクティビティ「ワイン配送在庫管理機能」の入力・出力として記述したものである。これは、いわゆる as-is 的なものとしての検討をする上での基礎、ないし、いわゆる to-be 的なものとしての検討した結果としては、情報量としては一番豊富である。ただし、現実の問題においては複雑になり、as-is から如何にして to-be を導くか、to-be をどう活用するか、いま一つはっきりしない。

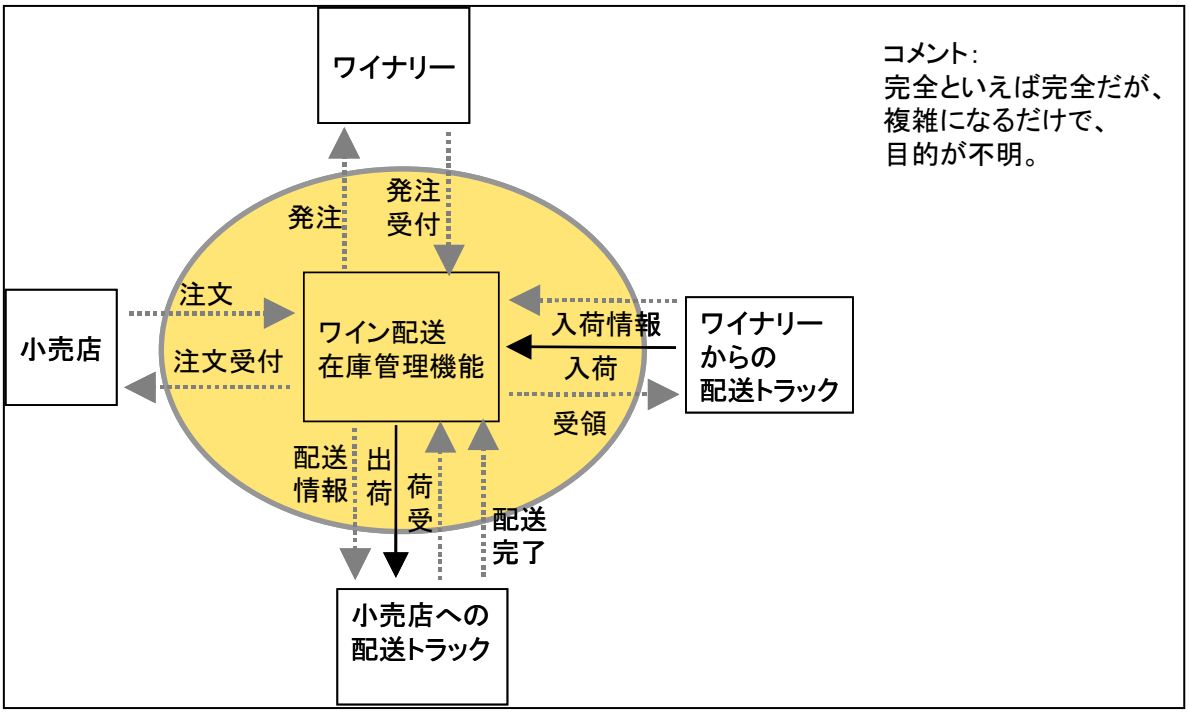


図 2.5-11　物の流れ・情報の流れをともに入力・出力とした場合

そこで、物の流れは捨象し、情報の流れだけに着目してみたのが、図 2.5-12 である。図 2.5-12 のようにアクティビティに対する入力と出力として情報に関するもののみを扱うのは、実際にIDEF0を実務に適用したモデルにおいても、よく見かける例である。また、情報のみを扱っているという意味で、情報システムに対するUMLのモデルや、あるいはデータフロー・ダイアグラムとの整合性も比較的高く、情報システムの設計のための一過程としては扱いやすい。

ただし、最適な業務の設計という観点から見ると、「ワイン配送在庫管理機能」の基本的責務の中心となる部分（小売店への配送トラックへの出荷、ワイナリーからの配送トラックからの入荷）が捨象されていて、基本的責務との関係で、この情報の流れが最適化どうかの検討ができない。図 2.5-12 での検討から得られるのは、基本的責務との関係のない範囲での単なる事務手続きの合理化だけである。これは、実際の情報システム設計の際にも、陥りがちな罠である。

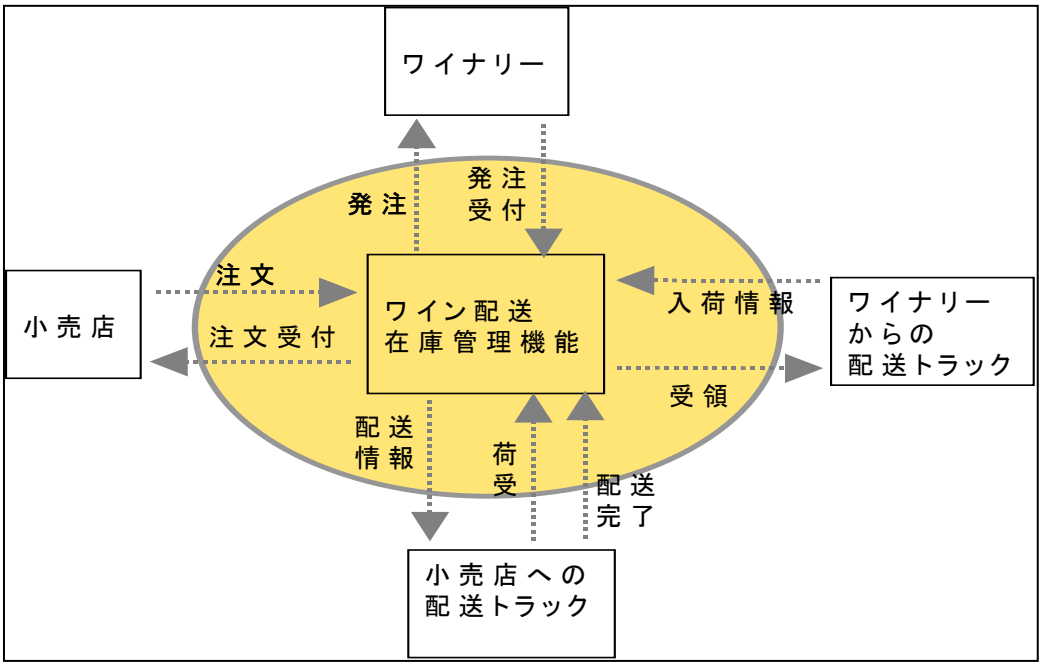


図 2.5-12　情報の流れのみを入力・出力とした場合

一方、アクティビティに対する入力・出力に関して、責務に直接関係する部分のみを抜き出すと、図 2.5-13 のようになる。図 2.5-13 は、確かに、最適化したい基本的責務を表していることは事実である。

ただし、これでは、その最適化のために何が必要なのか検討できない。最適な業務実現のために必要なのは、図 2.5-13 における基本的責務を是とした場合（もし、これを是すべきでないということが明らかになった場合は、図 2.5-9 にもどって、責務の分割・切り出しをやり直すことになる）に、その実現のために必要な状況把握（入力）、必要な指示（出力）を明らかにすることであり、それがまさに業務の設計である。

その検討結果の最も安直な一例が例えば図 2.5-10 になるのである。あるいは、小売店への実際の配送の責務は、「小売店への配送トラック」に委譲したにせよ、小売店への不測の配送遅延を招かぬようその状況を十分に把握することが必要であるとすれば、「小売店への配送トラック」から定期的に配送状況の報告を受けるといった図 2.5-14 のようなモデルも考えられるだろう。

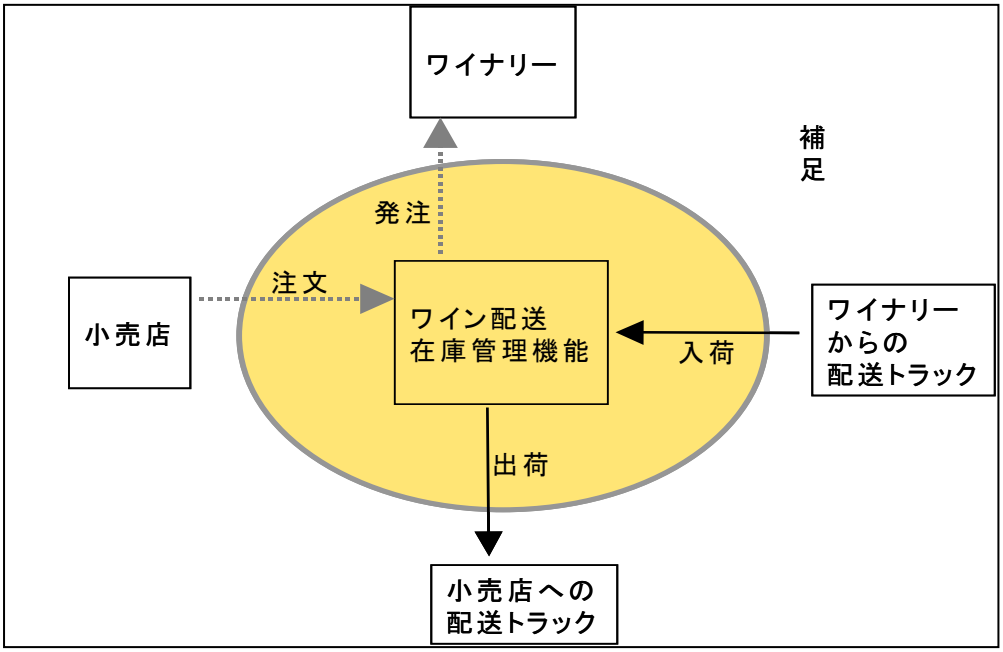


図 2.5-13　基本的責務に関するもののみ入力・出力とした場合

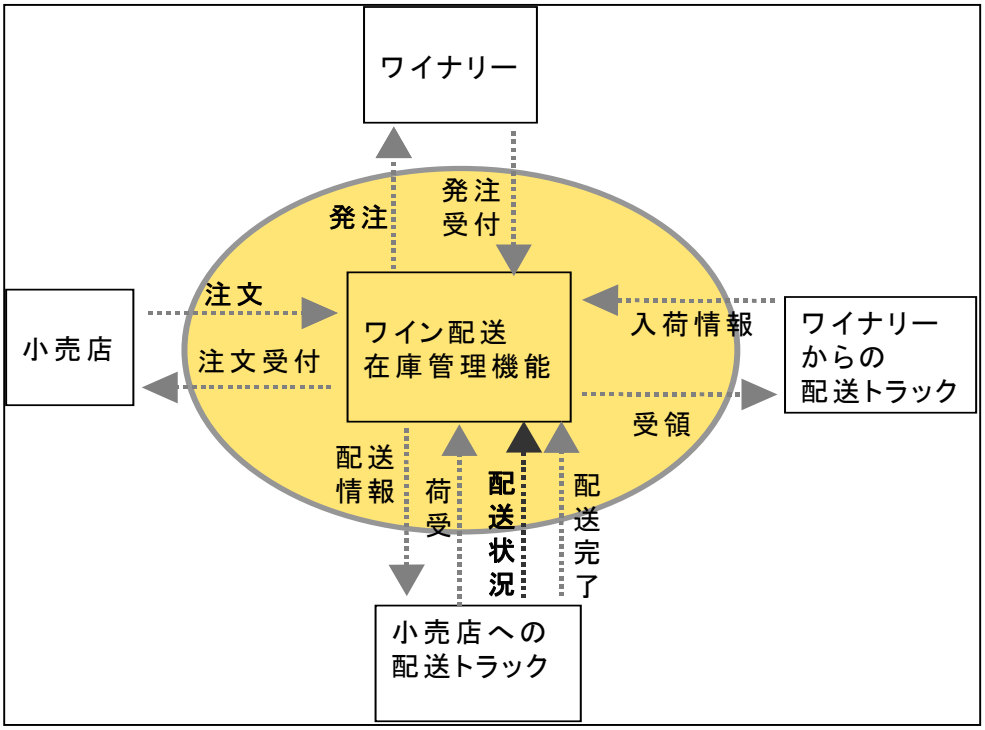


図 2.5-14　別のモデル例

この検討結果に基づき、入力・出力に対する「ワイン配送管理機能」の反応を示すと、図 2.5-15 のようになる。

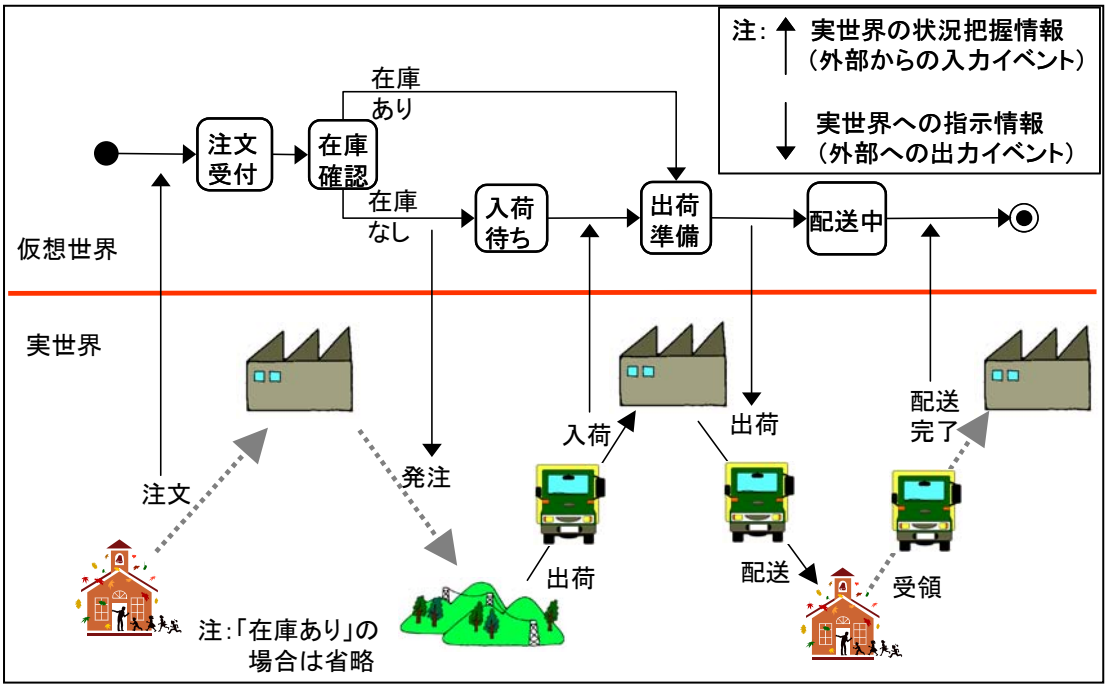


図 2.5-15　実世界のシミュレーションとしてのモデル

即ち、最適な業務設計としてのモデルは、実際の出荷・入荷といった実世界の責務の実現に対するいわば仮想世界でのシミュレーションになっている。また、模式的に書いてはあるが、これはまさしく、「ワイン配送管理機能」をオブジェクトと見た場合の入出力イベントに対するオブジェクトの状態遷移を表している。即ち、責務の実現を最適化するための制御（状況把握と指示）としての業務設計が、オブジェクト・モデルになる。この考え方は、責務駆動型のCRCカード法とも、相通じるものがあるのではないかと思われる。

ただし、これまでの考察からも明らかなように、オブジェクトの切り出しや粒度に関しては、責務を中心に考えるというだけであって、置かれ経営環境・目標にも依存し、明確なものはない。また、このモデルにおけるオブジェクトは、J2EE等RDBを前提にした分散オブジェクト環境への実装を意識したUMLモデルでのオブジェクトとは、粒度はもとより、切り出しの考え方においても一致するかどうかは定かでない。むしろ、そもそも目的の異なる両モデルにおいては、オブジェクトの切り出しも異なるのは当然と考えるべきであり、両者の安易な整合化を求めるのは、両モデルのそもそもの目的の没却につながる危険性があるように思われる。

3 ユーザ取り組みの実態

3.1 オブジェクト指向技術先進国　韓国

3.1.1 訪問の目的

韓国の経済成長は、1992年の通貨危機を乗り越え、目覚しい発展を遂げている。経済成長と同様に、勧告のIT産業も目覚しい。また、韓国は日本にも勝る高学歴社会であり、知的人材が豊富な国である。特にソフトウェア産業においては、この知的人材の量と質が、その成長に大きな影響を与える。人的資源に加えて、ITに関する国家戦略と企業戦略、さらに大学と研究機関の活動も成長の重要な鍵を握っている。

当研究委員会では、ビジネスオブジェクトに関する研究を通じて、CBOP(ビジネスオブジェクト推進協議会)との交流を持っている。CBOPでは、アジア圏でのコンポーネント流通の取り組みを行っており、韓国のビジネスオブジェクトとコンポーネントに対する取り組みが、わが国を上回る速さで進んでいるとの認識をもっている。この知見を得て、当委員会では、以下の目的を前提に、今回の韓国訪問を実施した。

(1)韓国におけるオブジェクト指向技術、及びビジネスオブジェクトに関する取組状況の調査。
(2)教育機関、民間企業におけるオブジェクト指向に対する考え方と対応の調査。
(3)ビジネスオブジェクトに関するコンポーネントや部品の国際協業の実現可能性について調査。
(4)可能で有れば、韓国政府のソフトウェア産業育成施策とオブジェクト指向技術への取組施策の調査

3.1.2 世界そしてアジアのＩＴ動向

3.1.2.1　　世界経済の動向

まず、韓国に焦点を当てる前に、世界経済全体の動向を鳥瞰することで大きな流れをつかみ、その後アジア経済全体を見た上で、韓国のアジアにおける位置付けを見ることにする。
21世紀前半に位置する現在、これまでに至る1997年代を振り返ってみる。アジア諸国、すなわち、NIES(newly industrializing economies:新興工業経済群)、ASEAN(Association of Southeast Asian Nations:東南アジア諸国連合)、中国、インド、ベトナムの11カ国は、実質経済成長率が91年に5．8%を記録し、92年が7．6%、93年には7．9%に達し、94年度には8．3%、95年度には7．9%を記録するに至っている。

アジアの力強い成長の背景には、欧米の緩やかな経済回復と円高があると言われており、全体的に輸出の増加が経済成長を牽引している。NIESはアジア経済の中でも、特にその傾向が強い。NIESでは経済成長に伴って、国民の生活水準が向上しており、内需拡大が輸出

と並んで経済成長を支えるもう一本の柱となっている。

一方、ASEAN経済は、インドネシアやフィリッピンにおける外資規制大幅緩和政策により外資導入が増加し、輸出が好調に転じると共に、民間消費もそれに引きずられる格好で伸びた。

1990年代における、この目覚ましいアジアの成長は、今後も堅実に続くというのが専門家に一致した意見となっている。(NJA…None Japan Asia)

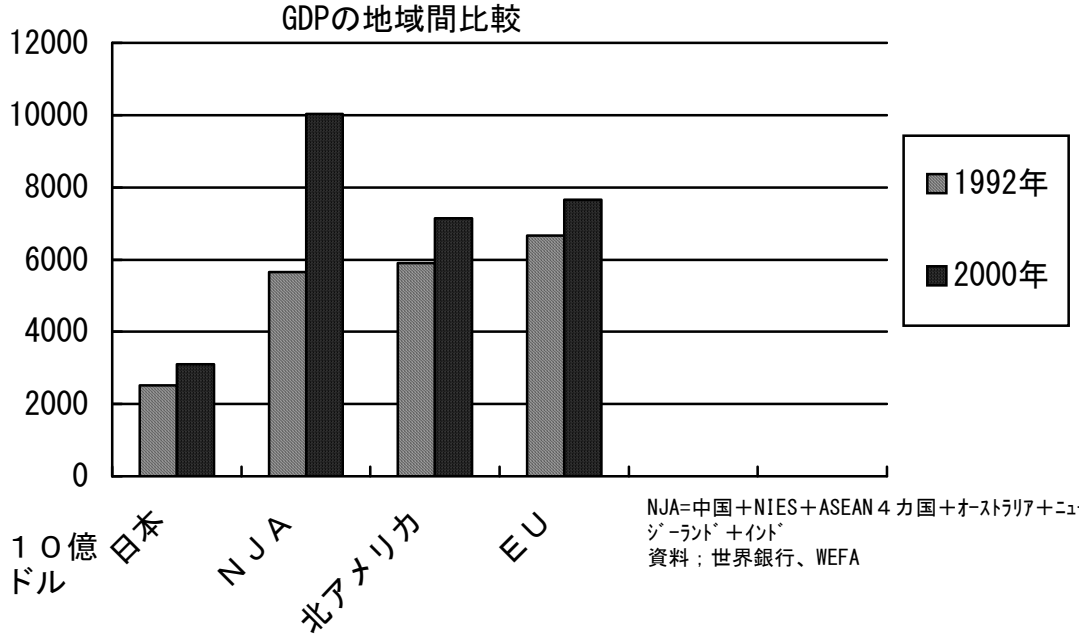


図 3.1-1　ＧＤＰの地域間比較

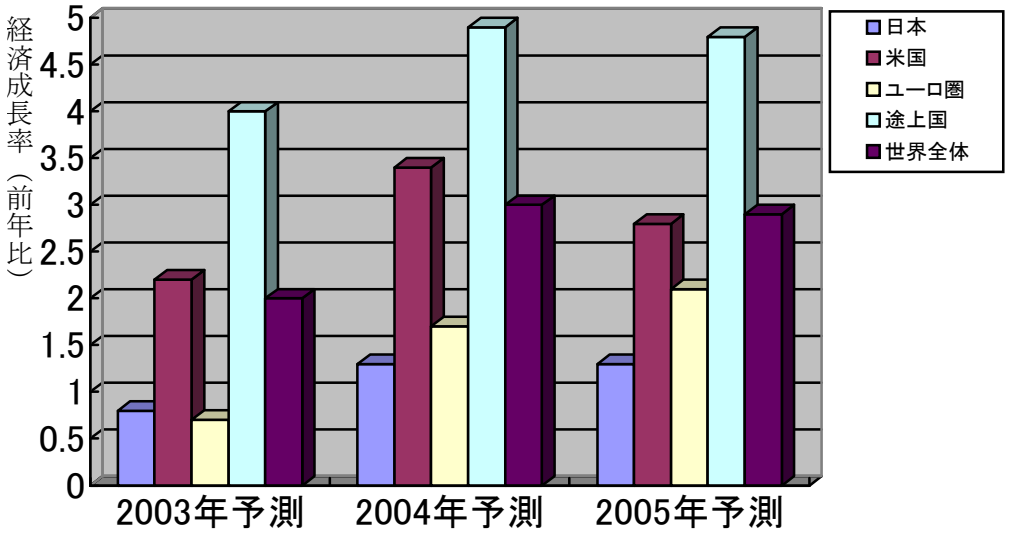
図 3.1-1 における格地域間のGDP成長率は、日本が2．7%、NJAが7．4%、北アメリカが2．4%、EUが1．8%となっており、NJAの92年度経済力は日本よりやや大きく、欧米よりもやや小さい規模に育ってきており、既に世界の四大経済圏に成長してきたと言うことができる。

さらに驚くことに、経済成長率は7．4%と世界で最も高く、2000年のGDPは世界で最も高い地域となっている。

既に世界経済は従来の日米欧といった三極モデルでは考えられないようになってきている。従来の三極モデルにNJAを加えた四極モデルがこれからの基本となるだろう。

このようなアジアの成長に日本がどのように参画してゆくかと言った問題は、国家間の取り組みだけにとどまらず、民間企業にとっても重要な経営課題の一つとなっている。

次に世界経済の 21 世紀前半に関する成長予測を見てみる。



【出典:世界銀行(平成 15 年 9 月 3 日発表)】

図 3.1-2　２１世紀前半に関する成長予測

世界全体では、ここ 3 年間で約3%程度の成長をする見込みである。日本は、1．3%ほどのなだらかな成長を行い、米国が2．2～2．8%の成長、ユーロ圏が0．7%～2．1%の成長見込みであるのに対して、途上国では4～4．8%の高い成長率を示している。同じグラフを経済圏毎に並び替えてみたのが次の図である。

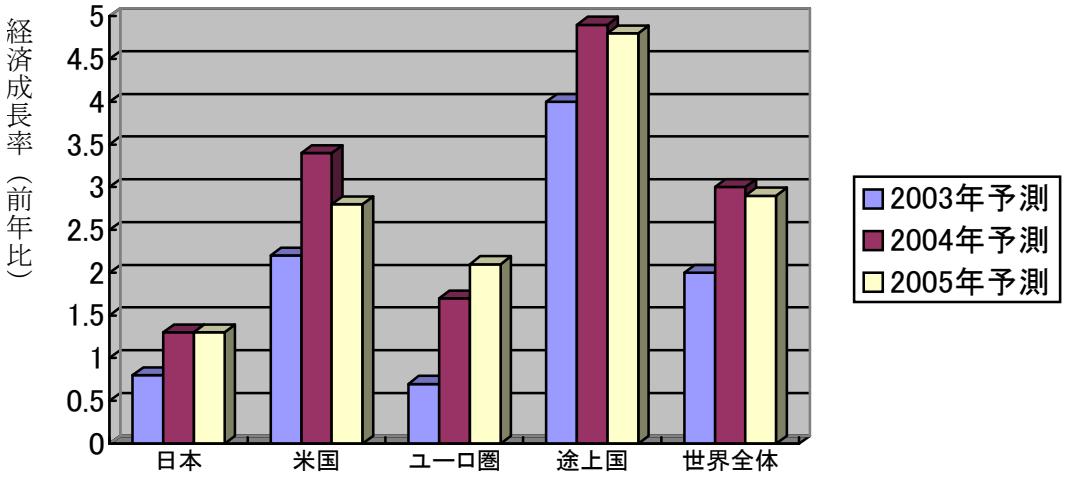


図 3.1-3

この予測で見る限り。世界経済の牽引役は、もはや米国や日本、EUなどではなく、開発途

上国であることがはっきりしている。

	2003 年予測	2004 年予測	2005 年予測
日本	0.8	1.3	1.3
米国	2.2	3.4	2.8
ユーロ圏	0.7	1.7	2.1
途上国	4.0	4.9	4.8
世界全体	2.0	3.0	2.9

注）単位：%、実質GDP成長率の前年比

表 3.1-1

3.1.2.2 アジア経済の現状と見通し

アジア主要17カ国・地域の実質国内総生産（GDP）は、1990年代後半よりそれぞれ平均7．1%、7．3%と世界的に見て突出した成長が続いた。

95年度のASEAN加盟国・地域のGDP伸び率は中国、香港、シンガポールの景気減速が影響して7．9%と94年度に比較して0．4ポイント低下したが、それでも世界全体の伸び率の3倍近い数字となっている。

中国は、引き締め政策の結果、96年まで成長の鈍化が続いた代わりに、消費者物価指数は95年の14．8%から10．0%に落ちついている。

NIES（新興工業経済群）の成長率は6%台へと緩やかに低下したが、南アジアでは成長が加速した。東南アジア7カ国の95年度以降成長率は7～8%台を記録し域内貿易を中心に今後2年間は輸出入とも20%近く伸び続け、7%代後半の成長率を維持している。

特に、ベトナムやフィリピンなどは、日本やアジアNIES、ASEAN（東南アジア諸国連合）からの投資拡大が寄与しており、域内での投資・貿易の自由化と経済関係の強化が後押しする形で、今後とも成長が加速する見込みである。

アジア開発銀行の分析では、アジア経済地域に於いては、労働力不足の資本輸出地域と資本不足の後発地域の間に強い補完関係があると見ている。

双方の資源を再配分することによって、たとえ技術革新がなくとも生産性は上がり得るとしている。

この分析結果は、ソフトウェア開発に関しても「双方の資源の再配分」と言う観点から同様のことが言えるであろう。

アジア経済過去5年のGDP成長率

アジア主要国・地域	1998	1999	2000	2001	2002
中国	7.8	7.1	8.0	7.3	8.0
インド	6.0	7.1	4.0	5.5	4.4
ベトナム	5.8	4.8	6.8	6.8	7.1
バングラディシュ	5.2	4.9	5.9	5.3	4.4
スリランカ	4.7	4.3	6.0	−1.5	3.0
ラオス	4.0	7.3	5.8	5.7	5.0
ネパール	2.9	4.5	6.1	4.8	−0.6
パキスタン	2.6	3.7	4.2	2.7	4.4
カンボジア	1.8	5.0	7.7	6.3	4.5
フィリピン	−0.6	3.4	4.4	3.2	4.6
シンガポール	−0.9	6.4	9.4	−2.4	2.3
香港	−5.0	3.4	10.2	0.5	2.3
韓国	−6.7	10.9	9.3	3.1	6.4
マレーシア	−7.4	6.1	8.3	0.5	4.2
タイ	−10.5	4.5	4.7	1.9	5.2
インドネシア	−13.1	0.8	4.9	3.4	3.7

出典：世界銀行調査

表 3.1-2　アジア経済過去５年のＧＤＰ成長率

3.1.2.3　　アジアにおけるＩＴ産業の動向

アジアにおける高度情報化社会を支えるインフラの整備は、1980年代以降急激に進行している。世界的にもアジア地域でのインフラ整備は盛んであり、これを可能にしたのが規制緩和である。アジア各国の基本インフラ整備の状況は、表3で見ることができる。

2000 年時点で、シンガポールや韓国ではそれぞれ48．2%、44．10%と先進国レベルに達している。一方、中国やタイ、フィリッピンなどはまだ10%台に達しておらず低い数字にとどまっている。普及率の低い国では、都市と地方の格差が大きく社会的な問題となっている。たとえば、フィリッピンの場合、マニラでは6．7%、地方は0．5%（93年）といった状況である。

このようなインフラが未整備の途上国では、未整備であるがゆえに光ファイバーやデジタル交換機、移動体通信、衛星通信などの最新技術が導入しやすいといった、「後発のメリット」もある。

アジア各国の基本インフラ整備状況

	固定電話普及率	移動電話普及率	インターネット普及率	インターネット利用者数
	台/100人	台/100人	台/100人	千人
日本	55.75	44.94	30.40	38,639
シンガポール	48.20	41.88	29.90	5,000
台湾	56.72	77.12	28.84	4,600
韓国	44.10	50.44	40.30	19,040
香港	57.60	54.90	48.69	2,360
ブルネイ	24.60	15.60	7.80	
マレーシア	20.30	13.70	15.00	3,500
中国	8.58	3.42	1.80	22,500
タイ	8.57	3.84	2.00	1,200
フィリッピン	3.95	3.66	1.30	1,000
スリランカ	3.64	1.22	0.30	65
インドネシア	2.91	1.06	0.70	1,450
ベトナム	2.68	0.42	0.10	100
パキスタン	2.22	0.21	0.10	80
ブータン	1.80	0.17	0.10	
ネパール	1.11		0.20	35
ラオス	0.65	0.17		2
ミャンマー	0.55	0.03		
バングラディ	0.34			
カンボジア	0.25	0.81	0.04	4
インド			0.45	5,000

出典:ITU資料(2000/10)及びNUAより当委員会作成

表 3.1-3　アジア各国の基本インフラ整備状況

また、アジアにおける携帯電話の普及率では、台湾が最も高く、次いで韓国と香港が日本やシンガポールを追い抜いてそれぞれ50．4%、54．9%と高い普及率を示している。

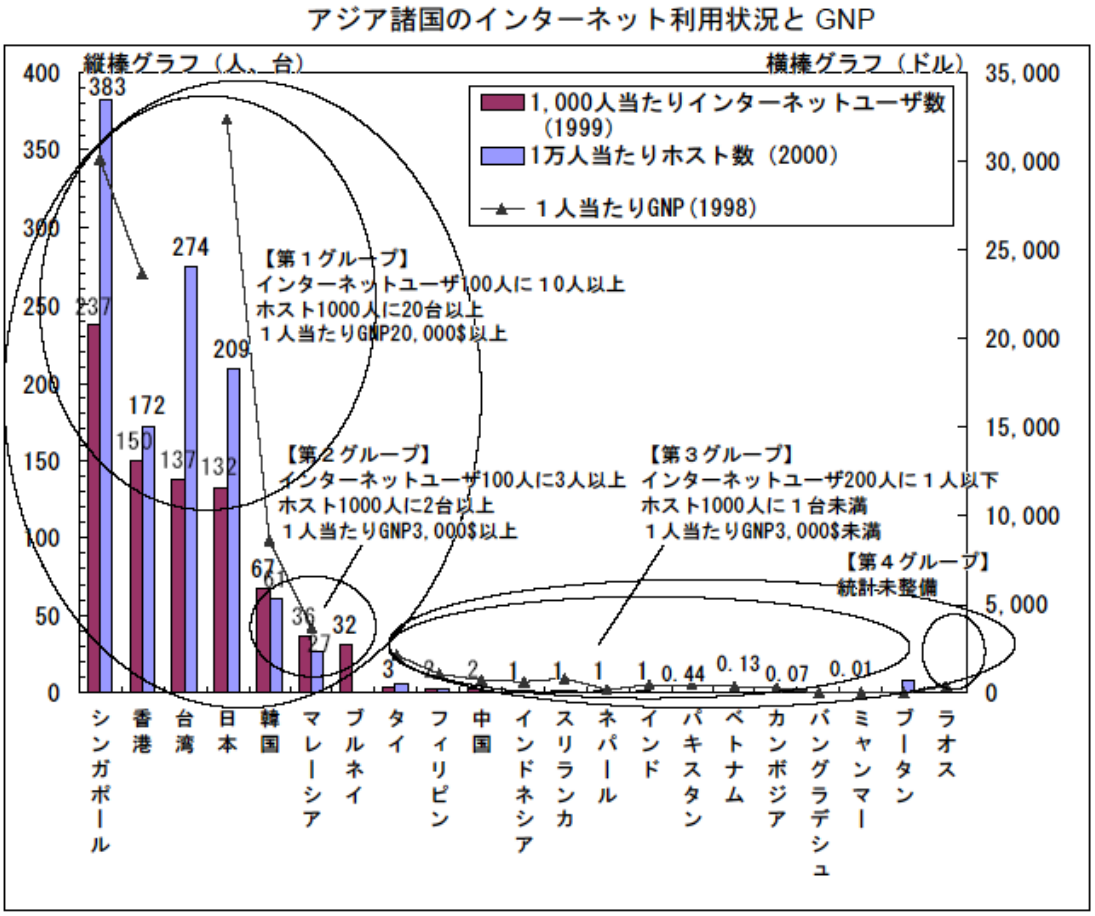
一方、固定電話の普及率が低い国々では、移動電話が一般用として利用されるケースが多い。経済発展によって、通信需要は急激に増加している。この増加する通信需要に膨大な資金と時間をかけなければならない固定電話の整備よりは、安価で短期間の通信インフラ整備ができる移動体通信が普及するのも当然といえる。また、広大な国土を持つ中国やインドでも、無線による通信がコストをかけない通信形態として発展している。

さらにインターネットの普及率をみると、香港や韓国では日本を上回っている。IT産業の高成長が期待されているインドや中国では、インターネットの普及数が増加しても、10 億以上の人口を抱えていることから、普及率の伸びは低くとどまらざるを得ない。

インターネットの利用者はアジアでも急激な増加を示している。アジア諸国のインターネット利用状況をみると、シンガポール、香港、台湾、日本など 1,000 人あたりユーザ数が 100 人を超えるグループ、韓国、マレーシア、ブルネイなど 1,000 人あたりユーザ数が 30-70 のグループ、および 1,000 人あたりユーザ数が3人以下のグループに、大きく二極化している。

JIPNIC(日本ネットワークインフォメーションセンター)の分析によると、インターネット利用状況は、1人あたりGNP とほぼ比例関係にあり、第1グループは1人あたりGNP が2万ドル以上、

第2グループは3,000 ドル以上の国々である。また、インターネット利用は国家経済規模と密接な関係があり、かつアジア諸国の利用状況は大きく二極化しているとしている。



出典：JIPNIC:2000/9

図 3.1-4　アジア諸国のインターネット利用状況とＧＮＰ

現時点でインターネットの普及率が高くないアジア諸国では、インフラとしてのバックボーンが整備されていないこと、PCの普及率が低いこと、ISPが少ないことなどが上げられる。しかしこれらの国におけるホスト数の伸びは著しく、95～97年の2年間でタイを除くASEANで11～54倍の伸びを見せている。

これらアジアの国々が、インターネットの整備を加速度的に行うことによって、アジア経済圏における国の壁、言語の壁、通貨の壁、距離と空間の壁が取り払われてゆくことになる。国際的な水平分業の時代が訪れようとしている。

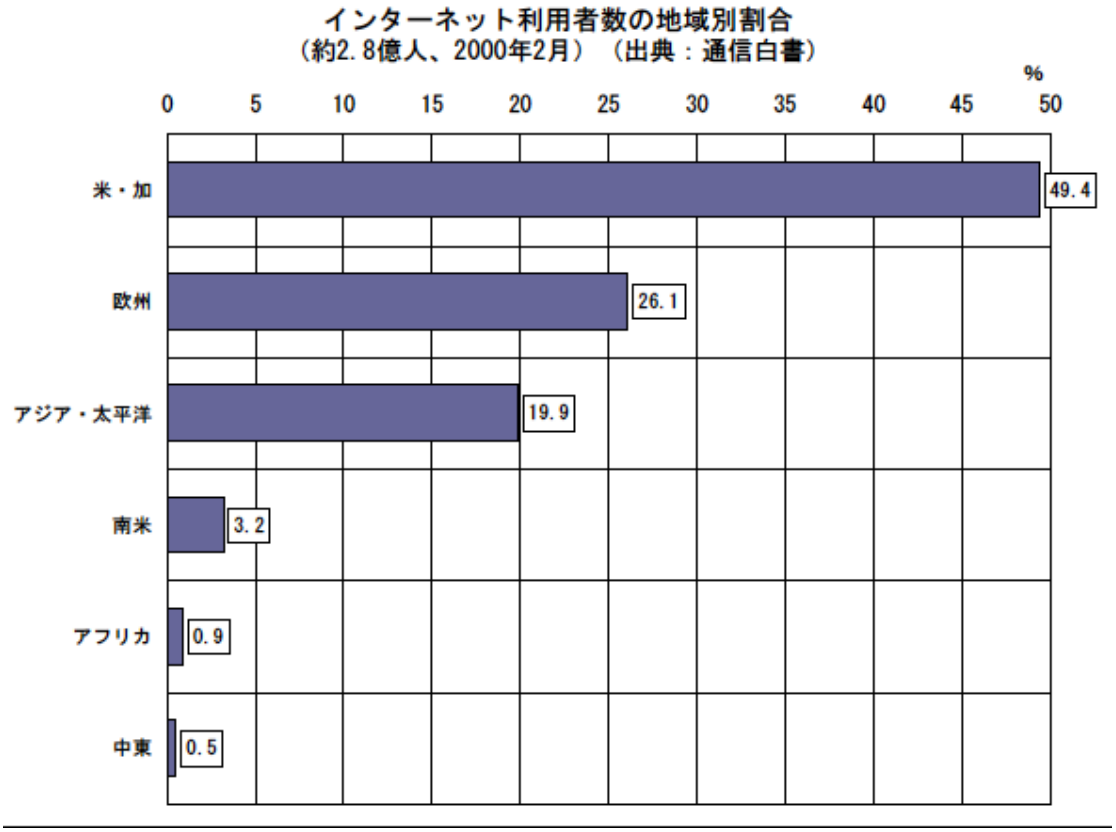


図 3.1-5　インターネット利用者数の地域別割合

エストラダ比元大統領は「欧州を見習い東アジア共通通貨の有効性を真剣に検討すべき時期に入った」と主張している。また、シンガポールのリー・クアンユー上級相は「米国は情報技術（IT）をいち早く導入しグローバル化を主導した」と述べ、日本を含むアジアの情報産業への出遅れに警鐘を鳴らしている。

アジア経済は1997年の通貨危機に端を発し停滞していたが、99年にはようやく回復の兆しが見えはじめた。とりわけ、経済回復を牽引するとして情報化推進のための投資はアジア各国で優先的に取り組まれている。

この取り組みは、国によって差はあるものの，データでわかるように情報インフラ整備計画への取り組みは非常に活発であり，パソコン普及やインターネットユーザ数も着実に増えつつある。さらに電子商取引（EC）は経済再生の起爆剤と認識されており、法的整備も積極的に取り組まれている。一方，情報産業分野においても，日本政府の提唱する e-Japan 政府や構想に代表される国家レベルおよび民間レベルの情報化投資を追い風に堅調な成長を遂げている。これらアジアの情報産業市場が持つ潜在成長力の高さは、世界の情報産業界によって注目されている。

さて、日本とアジアの関係を考えた場合、現在世界を席巻しているインターネットのネットワ

ーク技術について考慮を必要とする。 かつて日本が高度経済成長を遂げる事のできた大きな要因は、鉄鋼や自動車などの大規模基幹産業に集中投資を行い、必要に応じて規制による保護主義的な国内産業の育成を図るといった50年代から60年代の施策のモデルは、NJAに対しては通用しないからである。

当時と異なり、現在の世界経済は東西冷戦の終焉とともに、きわめて自由主義貿易の色を濃くしている。資本はボーダレスで世界中から調達し利用する事が出来る。技術についても同様に最高の技術をどこからでも入手できる。先進国は自国の高度な技術を持て余し気味であり、発展途上国に積極的にアプローチしている。

インターネットの技術はこの傾向にますます拍車をかけるであろう。開かれた市場としてのNJAは、自国の経済発展の為に、世界の先進国企業が持っている先進技術の導入に対して市場を解放するようになる。

IT産業は、最も先進的な技術を保有しているものの一つである。NJAでは先進企業のITを今後積極的に求めるようになるであろう。

アジアのIT産業に目を付け、今の内から手を打って置くことは、日本企業にとって戦略上も先行メリットを取る上で、重要な選択であると考える。

3.1.2.4　　アジアのＩＴ産業とＰＣマーケット

IT産業が興隆するためには、コンピュータ・リテラシーの向上が不可欠である。
コンピュータ・リテラシーが向上するためには、PCの普及が必要である。アジア各国におけるPCの需要を見てみる。(図 3.1-6)

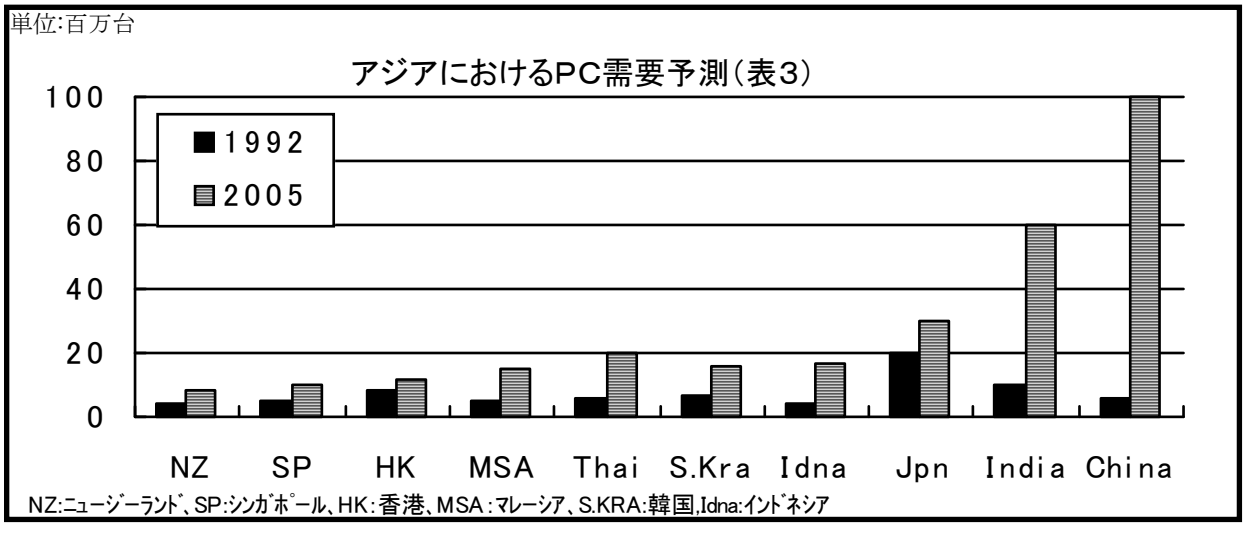


図 3.1-6 アジアにおけるＰＣ需要予測

やはり、中国とインドのPC需要がずば抜けているが、15万円のPCを購入するのに年収が350万円必要であると言われており、中国やインドでは生活費が年収に占める割合が30%に

及び、かつ年収が平均的に日本の20%といわれていることから、中国とインドの経済成長に伴い、ここ数年急激に需要が伸びることが予想される。

さらに、マレーシアやタイ、インドネシアなども経済成長率に伴って、購買層も増える確率が高くなる。PC市場としては注目してよいだろう。また、PCの普及に伴って、IT産業の成長も加速してくるものと思われる。

3.1.2.5 アジアにおける日本の役割

グラフを見やすくする為に、日本、韓国、台湾、及びインドの4カ国について、各々国民一人当たりのGNPの伸び率を過去の実績と今後5年間の予測をもとに、点描してみた。(図3.1-7)NIESについては、韓国、台湾を参考に考え、開発途上国は、インドを基準に考えれば十分である。

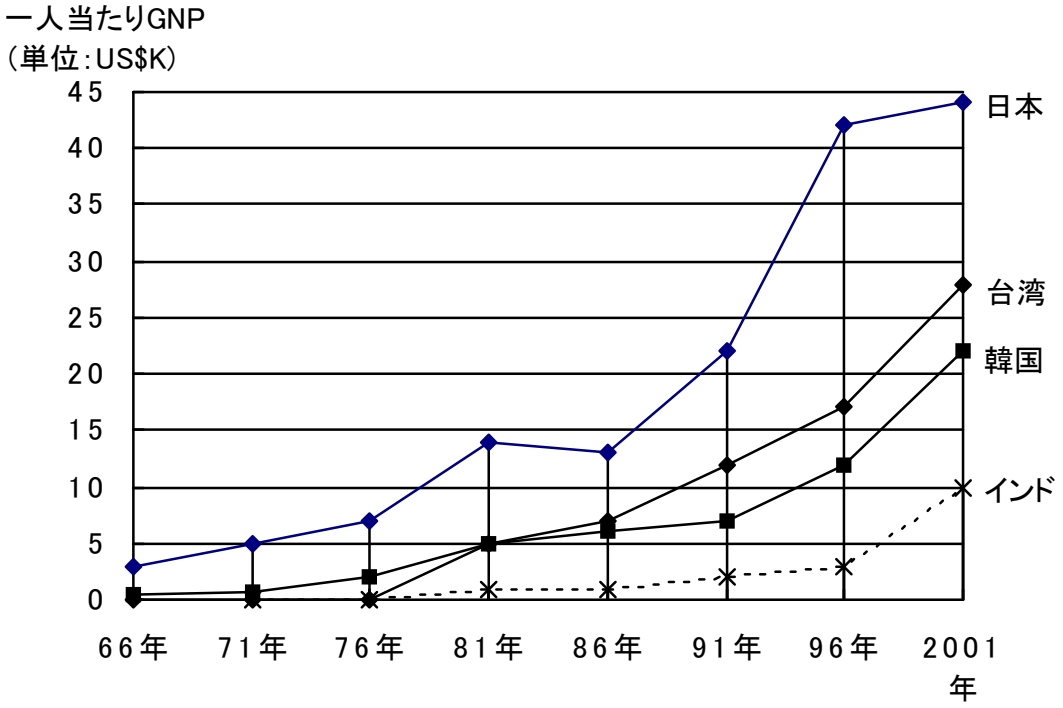


図 3.1-7

このグラフから考えられることは、台湾や韓国のようにアジアの中でも経済的に成長している国が日本よりGNPにして10年遅れていること、インドについては20年ほど遅れている事である。しかし90年以降の各国の成長率は急激な立ち上がりを見せており、状況の変化は加速度的となろう。

先に、現在の経済指数が日本の何年昔と同じだとは言えないと書いた。しかし、インフラの

整備など基幹工業についてはある程度の類推がきく。

いま、これらの国で何が起きているかは、日本の10年前、20年前を考えればおおよその想像ができる。85年時代の日本、75年時代の日本に何が起きて、何を必要としていたかがアジア・ビジネスを考える上でヒントになるだろう。例えば、75年時代には製鉄業界での大規模なコンピュータ化が行なわれていたし、85年時代には金融機関の第3次オンラインが進行していた。

同様な動きが、インドや韓国で今起きている。このような考えに対して、インドでも宇宙衛星を打ち上げ、水爆を持ち、韓国でもハイテク産業が興隆しているではないかと反論もあろうが、このグラフはあくまでも国民一人当たりのGNPを指し示しており、国家としての頂点に位置づけられた技術力を示したものではない。このことは、1940年代の日本において、農業中心の国家でありながら、戦艦大和や武蔵などの当時のハイテクが貧困と共にあったことを考えれば、納得してもらえるだろう。

経済と言う言葉が「経世済民」から来ているように、豊かさとは国民一人一人が実感できるものであるはずである。その為の経済であり社会であり、政府である。

今、日本がアジアの為に何ができるのか、について考えるとき、豊かさの再配分がその基幹にあると考える。戦後50年間に綿々と我々の先輩が築いてきた工業や経済の基盤と、その上に成り立っている現在の我が国の豊かさを、これらの国々の人々と共有する事が、今後日本がアジアとともに生きて行くことの出来る道であると考える。アジアの人々が必要としているものが日本にあるのなら、それを提供することで、我々の役割の幾つかは果たせるはずである。

日本企業がアジアのビジネスを考える場合にも、その行動原理となる思想と方向性が必要である。

経済的には豊かになった我が国も、心の豊かさにおいては、他アジアの人々の方が勝っているのでないかと感じることがある。経済的に日本より10年遅れていることが、国家として、あるいは民族として劣っている事を示してはいない。

アジアと取り組むときに、日本人が陥りやすい陥穽がここにある。ビジネスとしては対等、かつ公平な立場に立って、相互の利益を求め合う姿勢が、最初になければならないはずである。

3.1.2.6　　世界のＧＮＰ割合の変化

95年度の調査による世界のGNP成長率は世界平均で2%程度と言われている。先ほど見てきたように、日本を除くアジアの成長率は7．5%とされているから、これらの成長率を前提として2005年のGNP比率を計算すると次のようになる。

	1992 年 GNP	%	2005 年 GNP	%
EU	68,385	28.8%	81,102	26.4%
米国	60,385	25.5%	71,615	23.4%

日本	36,689	15.5%	43,512	14.2%
ロシア,アフリカ, ラテンアメリカ	45,406	19.2%	53,850	17.6%
アジア	18,565	7.8%	47,534	15.5%
カナダ	5,638	2.4%	6,686	2.2%

表 3.1-4

2005年のアジアの世界におけるGNP比率は15．5%となり、92年に比較して約2．6倍の成長率となる。これに日本のGNPを加えると、両者で29．7%を占める事になり、この数字はアメリカやEUよりも大きな市場に成長することを示している。

3.1.2.7　　アジアを狙った世界の動き

米国やEU各国が、アジアのビジネスに躍起になっているのもこのような予測があっての事である。

多くの多国籍企業は、香港、シンガポール、東京、ソウルに拠点を持って積極的な営業を展開している。1980年代に東京に集中していた多国籍企業は東京の地価高騰に嫌気をさしてシンガポールへ移転している。

しかしアジアを狙った輸出の事を考えた場合、地理的な問題からすれば、米国やEUよりも日本の方がはるかに有利である。例えば、コンテナーの海上輸送では、米国やEUは30日以上かかるが、日本からは3日以内で輸送できる。航空でも同様な優位性がある。

同様な事は、韓国も考えており昨年の神戸地震以来、釜山港の貨物取引高は日本の主要港を凌ぐまでに成長している。

シンガポールは、国家としての大方針を3つに決め、政府と経済界と国民が一つになって動いている。その方針とは、シンガポールは国家規模のファイナンスで利益を得ようとしていること、観光のメッカになろうとしていること、そして東南アジアのハブとして位置づけられようとしていることである。

いずれにしても、アジアの今後の成長を見越した各国のビジネス活動は熾烈を極めている。

3.1.2.8　　アジアにおける産業の成長度合い

アジア各国における産業の成長度合いと、今後必要とされる技術について次のような分布図を作成した。この図は、各国のGDPをもとに現在投資が行われている主要な産業についてプロットしたものである。

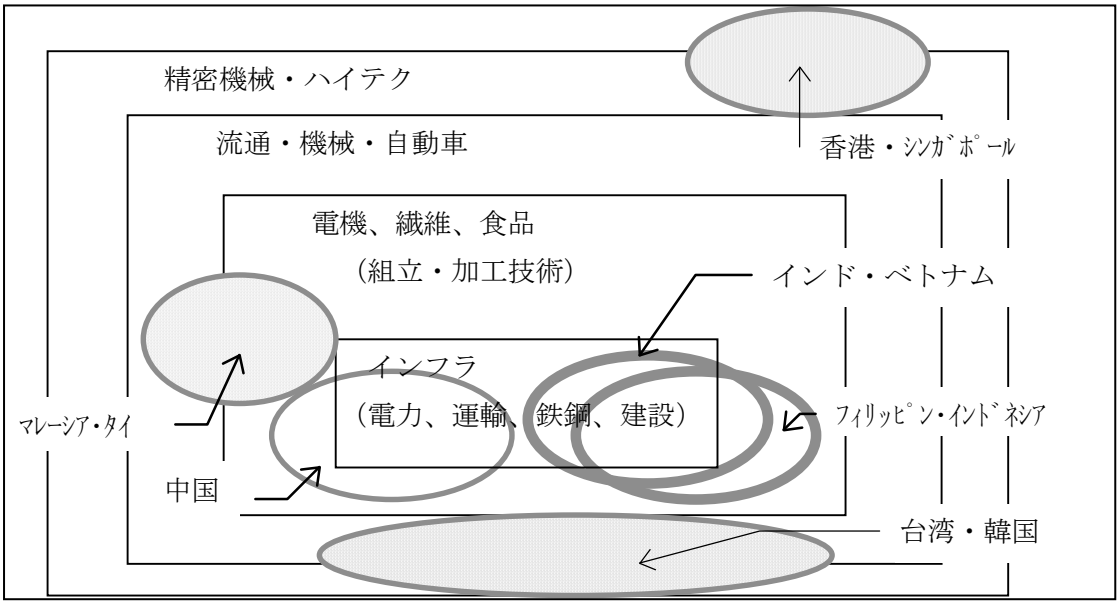


図 3.1-8

インフラ整備が中心となっている国々においては、プラントの開発や輸出がわが国にとってビジネスの機会となる。特に、インド、ベトナム、中国、フィリピン、インドネシアなどの国々がこれにあたる。

また、マレーシアやタイなどは、一応のインフラを基本に、今後は自動車や機械、流通などの分野が成長するであろう。さらに、台湾や韓国は、主要先進国からのアウトソーシングによって、精密機械などのハイテク産業が伸びる。

香港、シンガポールは場合によっては日本を越えている産業分野がある。例えば通信などの分野では、衛星通信や光ファイバーによる高度な通信網が敷設されているし、金融機関の国際システムも日本以上に近代化されている。

3.1.2.9　　アジアにおける日本のビジネス

日本は既に成熟した経済社会を迎えており、その成長率も今後はGNPで1％以下と予測されている。一方、アジア全体では7．5％の成長が今後も見込まれており、先進国にとっては、大きな市場と考えられている。

日本が今後アジアに対してビジネスを考えてゆく場合、以下に示すようなビジネスの可能性

がある。

(1)落差性のビジネス

高い技術を持った国から、低い技術の国へ落ちてゆく落差を利用したビジネスである。例えば、PCのOSについては、米国が高い技術力で他国を圧倒している。日本はこの分野では米国よりも低い技術力であるため、一方的に米国からの技術、すなわちOSのソフトを輸入している。

同様に、アジアが必要としている高度な技術が日本にあれば、それはアジアで売れるということである。日本の中古の圧延プラントが台湾に売れたという話などはこれにあたる。

例えば、高度なアプリケーション・ソフトなどは米国よりも日本の方が優秀であると世界銀行は調査報告書のなかで述べている。これなどは、日本がアジアへ輸出できるもののひとつであろう。

(2)密度差のビジネス

ある商品の投入数が少なくて、その商品に人気がある場合には、需要が供給を凌ぐようになる。たとえ技術的に高度でなくとも、品数が少なければ売れる。

このようなビジネスがアジアには存在している。例えば、日本で廃棄処分になるような自転車が、リフォームする事もなくアジアで引っ張りダコである。

ソフトウェアの世界でも、中古のPCなどは市場性があるはずである。

(3)揮発性の商品ビジネス

ソフトウェアにも揮発性のものと不揮発性のものがある。ソフトウェアにおける揮発性とは、娯楽性のあるソフトがその代表格であろう。例えば、映画のビデオやファミコン・ゲーム、パソコンゲームなどは、一度遊んだら、消費者にとっては興味の無いものと化して、さらに新しいゲームなり内容を持ったソフトが欲しくなる。

このような意味から、揮発性のビジネスとは、ライフサイクルが極めて短く習慣性・耽溺性を生じやすい商品を対象としたビジネスを指している。

ゲーム・ソフトやゲーム機、カラオケのソフトやカラオケ機器などがこれに類似した商品にあたる。

日本のコミックやアニメ、映画などがアジア諸国で人気を博しているのも、アジア経済が豊かになるにつれ、アミューズメント指向が出てきていることのあかしであろう。

3.1.3 韓国のＩＴ動向

韓国の経済は、1960 年代半ば以後経済発展の軌道に乗り、「漢江の奇跡」と呼ばれる程の高度経済成長を続けてきた。1962 年から 1999 年までの間、韓国の GNP は 23 億ドルから 4,021 億ドルに、国民一人当たりの GNP は 87 ドルから 8,581 ドルに増大している。

そして、現在に至っては「IT版の漢江の奇跡」とも言えるブロードバンドでの最先進国へと変貌を遂げている。21世紀における重要な社会インフラとしてのブロードバンドは、韓国にとっての国家戦略でもある。

韓国おけるIT情報技術革命は、インターネットを国民の半数に提供し、700万世帯を超える家庭をブロードバンド社会に参加させている。この目を見張る急成長には、韓国政府の周到な国家戦略があった。

【韓国政府の情報通信戦略年譜】

2001年6月	韓国情報通信部がOECD情報通信委員会で韓国の情報化について発表。
2000年3月	3・1節記念辞で「私に任期中に全国民がコンピュータを持ち、インターネットを使う。韓国は知識10大強国になる」と発言。
1999年6月	韓国通信（KT）がADSLサービス
1999年4月	ハロナ通信がADSLサービス
1999年3月	金大中大統領が「サイバーコリア21」を発表
1999年3月	韓国政府が国民情報化教育総合計画を推進
1998年2月	金大中大統領が就任
1998年1月	金大中氏が大統領就任直前に「知識基盤国家の建設」を提唱
1997年9月	ハロナ通信が設立される。
1997年 12 月	IMFが韓国支援決定。
1997年 11 月	韓国で通貨危機。
1996年6月	情報化推進基本法を施行。
1995年	KII（計画を策定韓国の超高速情報通信網構想計画. Korean Information infrastructure）計画を策定。

表 3.1-5

さらに、この国家戦略に基づき、いくつかの特筆すべき動きが韓国の成功を支えている。

- インターネット教育が小中高校でも積極的に取り組まれた。

- 時間当たりで料金を払い、インターネットを含め、パソコンを使うことができる「PCバン」がブームとなり、若者のインターネット利用が加速した。
- ADSL（非対称デジタル加入者線）、光ファイバーが敷設されたマンションによるブロードバンドの家庭への普及が進んだ。韓国はマンション文化である。
- デジタルデバイド（情報格差）対策として主婦や高齢者などに対するIT研修が行われた。

韓国政府の情報化戦略の矢継ぎ早な実施と、実現のための施策をみると、「迅速」であり「具体的」であることが分かる。さらに、次のような施策をも打ち出している。

- 外国人IT技術者ゴールドカード

  ゴールドカード（出入国特恵付与）とは、先端技術人力に対する拡大施行である。韓国産業資源部が BT（バイオテクノロジ）、NT（ナノテクノロジ）、新素材など先端産業分野の発展のために法務部と協議して現代の IT、e-ビジネス分野の海外人材に対する出入国特恵措置を与えるというもの。『ゴールドカード』適用は BT、NT、デジタル家電、新素材、環境エネルギー、輸送機械などの 6 ヶ先端技術分野に追加拡大されている。

  ゴールドカード適用分野拡大と共に産業技術財団内に『国際産業技術協力センター』を設立してゴールドカード発給審査、海外技術人力 DB 構築など、海外技術人材関連綜合支援サービスを提供している。
- ビジネスモデル特許に対する優先審査制度

  「ビジネスモデル特許」とは、模倣が容易なインターネットビジネスを保護する手段として効果を発揮する。韓国政府は 2000 年 7 月に「電子商取引関連特許出願優先審査」を設置し、インターネットに不可欠な技術の保護を推進している。

  対象となる発明は「ビジネス方法関連発明、電子マネー、電子決済、セキュリティー技術」などであり、これらは他の出願に優先して特許審査を受けることができる。この制度により、審査官は特許付与可能かどうかを二か月以内に出願人に通知しなければならない。これに対して、日本の「早期審査請求制度」は、出願と同じに審査を行うが、審査期間はおおむね 12 ヶ月程度となっており、韓国に比べて尾政治にも早いとは言えない。
- 情報化教育制度

  金大中政権が経済復興と併せて「情報教育に重点を置く」と公約した。前政権では、教育現場への直接的な投資をおこなったが，新政権は社会的間接資本に重点を置くことによる情報化教育の充実政策を行った。2000 年に入り，教育部は情報教育の新たな政策「初・中等情報通信技術（ICT）教育必修化計画」を発表。その内容は，2001 年度から小学校全学年を対象に週に1時間のパソコン教育の義務化を行うというものである。教育部は，初等教育では「パソコンを用いての意思疎

通能力」，中等教育では，「情報収集，分析能力及び学習活用能力，情報教育資料総合管理及び体系化」を習得する目標を掲げている。更に，政策が大学等の高等教育機関に反映させるために，既に高等学校課程で施行されている「情報素養認証制度」が中学校までに拡大され，今後，高校・大学入試に反映する計画もある。

- サイバー教育

  韓国の情報通信部（日本の郵政省に相当）が 1997 年 11 月に提案したもの。数学、情報科学や理科などの理系分野に限り、飛び抜けた才能を示す中学生と高校生を韓国高等科学技術研究所(KAIST/Korea Advanced Institute of Science and Technology)が担当して開発するインターネットを利用した「サイバースクール」で教育し、韓国から「天才」を生み出そうという提案。韓国は日本より激しい学歴社会が浸透していることから、初年度は数学、情報科学、理科の 3 つの分野で、中学生と高校生を 100 人ずつ選び、600 人のサイバー教育が実施され、今後、選抜する生徒を増やして 1999 年には 900 人、最終的には 2000 人程を選抜される。サイバースクールは一般にも公開されるが、選抜されなかった ID 番号がない生徒は教授陣のフィードバックが得られない。選抜された生徒は、実際の「学歴」でも推薦で高いランクの大学・高校に入学できる強力な学歴社会のバックアップをする制度になる。

- 高齢者情報化教育

  情報通信部と韓国情報文化振興院が、「一緒に共有するデジタル平等社会」を実現するため、情報アクセスに困難を感じている障害者、高齢者に実生活に必要な多様な情報を開発、普及し、関連ウエッブサイトを目録化し、ディレクトリサービスを提供し、情報格差解消のための法令、施策、研究報告書、海外動向などの情報サービスを提供する統合サイトとして「援助の国」がある。

3.1.3.1 韓国のＩＴ教育

韓国のIT（情報技術）革命に関する大きな特徴は、IT教育を全国民に対して実施するとしている点である。韓国政府は1999年3月、「国民情報化教育総合計画」を決定し、韓国国民の全員がインターネットを使うことができるようにするという目標を掲げた。

1999年	全国の郵便局506カ所に情報教育センターを設置。2000年末までに1000カ所
1999年10月	郵便局の「国民コンピュータ貯蓄預金」を2回以上貯金した人にパソコンを優先設置。3カ月で100万人。5カ月で24万6000台
1999年3月	国民情報化教育総合計画スタート

表 3.1-6 韓国のデジタルデバイド対策

この目標のもと、学校教育現場でのIT教育に関して、まず2001年4月までに小中高校全校を対象として100万台のパソコンの配備が行われた。もちろん、学校同士は高速インターネットで結ばれている。教室へのインターネット接続も100％完了した。これに対して日本では、全国小中高校をインターネットに接続するのが2003年度までとしているから。韓国よりも 2 年遅いことになる。

主婦のIT教育に関しては、2001年1月に韓国女性部を設立、「女性情報化基本計画」を策定している。これは、主婦を対象に全国800カ所の情報処理学校でインターネット講座が開設され、主婦のIT教育を実施するもので、大規模な女性のIT教育として、世界に類をみない試みとなっている。一ヶ月 10000 ウォンの月謝がかかるが、この施策のおかげで、韓国のITリテラシーに関する男女の格差は急速に縮小している。日本は、主婦のインターネットへの関心は高まっているが、韓国のような大規模な取り組みにはなっていない。

義務徴兵制をとっている韓国では、18歳以上の男子は2年2カ月の兵役義務がある。小中高校でITに習熟していない男性も、軍隊でIT教育を行うことにより、ITリテラシーの向上が行われている。

韓国では、少年院改革を 1999 年 9 月から開始し、英語教育と情報化教育に力を入れている。具体的には、ネイティブスピーカによる英語教育や専門家によるコンピュータ教育を行うものである。

特に情報化教育に関しては、全国の少年院で一人一台のコンピュータとその 24 時間使用可能を実現している。また、社会奉仕活動の一環として、地域住民や障碍者に対して、簡単なコンピュータ・リテラシー教育を少年院の生徒が行ない、中古PCを少年院で生徒が修理し、障碍者に寄付する活動も実施している。

さらに。一般の学校の資格として、少年院での修学を認め、前の学校の卒業証書をもらえる措置も行われている。このような取り組みの結果、多くの生徒がベンチャー企業に就職でき、再犯率も大幅に引き下がっている。

このような徹底したIT教育を支える格好で、政府、大企業、地方自治体などの公共施設でインターネットの無料使用ができる計画や、2005 年までに外国人向けも含めた、スラム地域への無料利用施設を作る計画もある。

この他、郵便局では国民PCとして日本円で 5 万円ぐらいの安いインターネット用ネットワークパソコンが売られている。さらには、PC が持てない家庭には政府から永久貸し出しを行っている。また、大手通信会社ではADSLに加入すればPCを無料で貸与するサービスを行っている。

3.1.3.1.1 住宅とＩＴ

韓国では、1997年に「住宅用建築物」にインターネット対応の認証を与える制度を発足させている。さらに翌年1998年3月には「業務用建築物」に「認証」を与える制度を発足させて

いる。この認証の仕組みは、通信速度が100Mbps の回線を敷設した建築物には「1級」を、10Mbps から100Mbps の間は「2級」を、10Mbps が「3級」として区分されている。これらのインテリジェントな建造物には、内部に通信管理室を設けることが義務付けられている。必要がある。特に、人口が集中しているソウルは高層マンションが多い。これらのマンション群がADSL（非対称デジタル加入者線）か、光ファイバーケーブル、ケーブルテレビ・インターネットかはともかく、ブロードバンドでつながれると「インターネットマンション」になる。

注目点は、韓国政府が高速通信網を整備したマンション、ビルに「認証」を与える制度を導入し、効果を上げていることだ。これがマンション社会の韓国でブロードバンドの普及、IT革命を加速する1つの起爆剤になっている。

3.1.3.1.2 韓国に関するデータ

1. 面積 9 万 9,274km^2
2. 人口 4,778.6 万人（2002 年 12 月現在）
3. 首都 ソウル
4. 人種 韓民族
5. 言語 韓国語
6. 宗教 仏教徒 27%、キリスト教 24%、その他儒教徒、天道教
7. 略史 3 世紀終わりに氏族国家が成立。三国時代（A.D.4 世紀～668 年）、新羅（668～935）、高麗（918～1392）、朝鮮（1392～1910） を経て、日本による統治（1910～1945）を経験。第 2 次大戦後、北緯 38 度以南は米軍の軍政下に置かれることになる。1948 年大韓民国成立。同時に北半分では北朝鮮（朝鮮民主主義人民共和国）が成立。
8. 経済
   （1） 韓国の GDP 成長率は、2002 年に 6.3%を記録したが、2003 年は、北朝鮮の核兵器開発問題、国内信用不良者の増加、ストの多発等のリスク 要因が指摘されており、3%台にとどまる見通し（第 2 四半期の成長率は 1.9%）。貿易収支は、2002 年 104 億ドルの黒字を記録。2003 年は 9 月までの累計で 83 億ドルの黒字。

   （2）盧武鉉政権は、構造改革の継続を表明しており、主な経済政策として、北東アジア経済中心国家の建設、国家の均衡的な発展、参加福祉の実現と所得分配の改善等を掲げている。

   （3）景気が後退しつつある中で、雇用の安定等を求める労働運動が頻発している。これに対し、盧武鉉政権は、不法な争議であっても労働者側の主張に耳を 傾けるとの姿勢を示したが、争議の頻発と財界等からの批判により、不法行為に対しては、厳格に法と原則を適用する姿勢を改めて表明している。

   （2） 主要産業 電子、自動車、機械、造船、鉄鋼、石油化学

(3)　名目 GDP 5,963 千億ウォン(2002 年)(韓国銀行速報ベース)

(4)　一人当たり GNI 10,013ドル(2002 年)(韓国銀行速報ベース)

(5)　経済成長率 6.3%(2002 年)(韓国銀行速報ベース)

(6)　物価上昇率 3.4%(2003 年 11 月)(統計庁、対前年同月比)

(7)　失業率 3.3%(2003 年 10 月)(統計庁、季節調整前)

(8)　総貿易額

1)輸出 1,625 億ドル(2002 年)(産業資源部)

2)輸入 1,521 億ドル(2002 年)(〃)

(9)　主要貿易品目

1)輸出 電気電子製品、自動車、機械類、鉄鋼

2)輸入 電気電子製品、原油、機械類、化学製品

(10)　主要貿易相手国

1)輸出 米国、中国、日本、香港、台湾

2)輸入 日本、米国、中国、サウディ、豪州

(11)　通貨 ウォン

(12)　為替レート 100 ウォン＝約 11 円(平成 15 年 10 月末:韓国銀行)

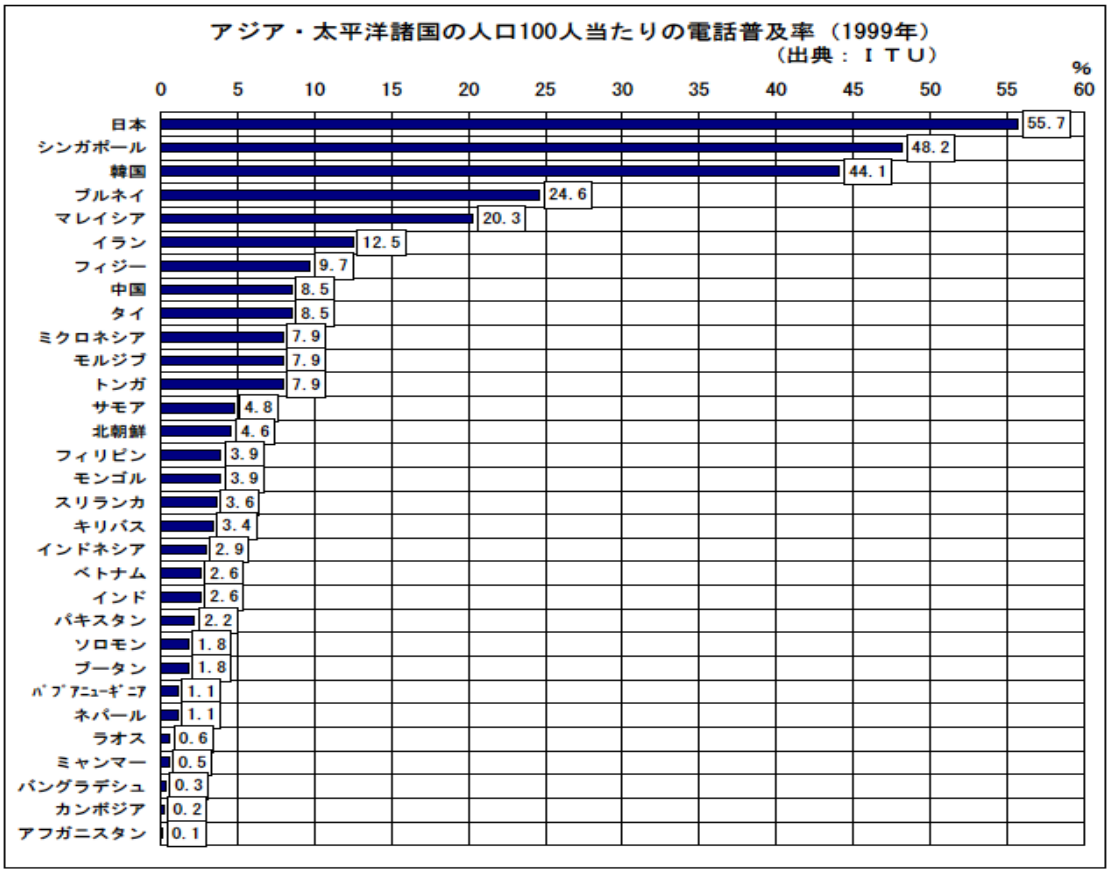


図 3.1-9　ビジネスはアジアに在り ‘95 ダイヤモンド社

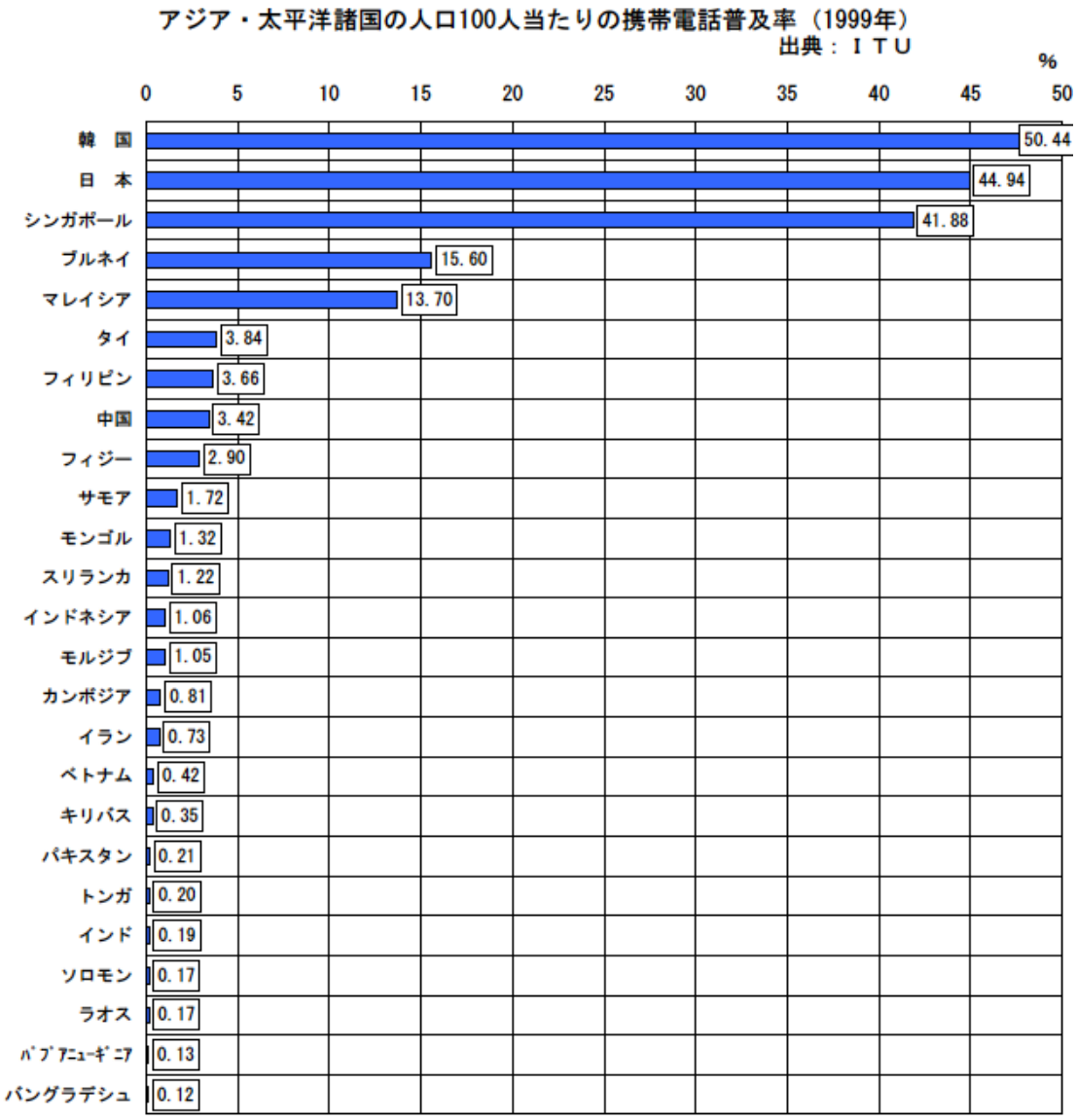


図 3.1-10　アジア・太平洋諸国の人口 100 人当たりの携帯電話普及率（1999 年）

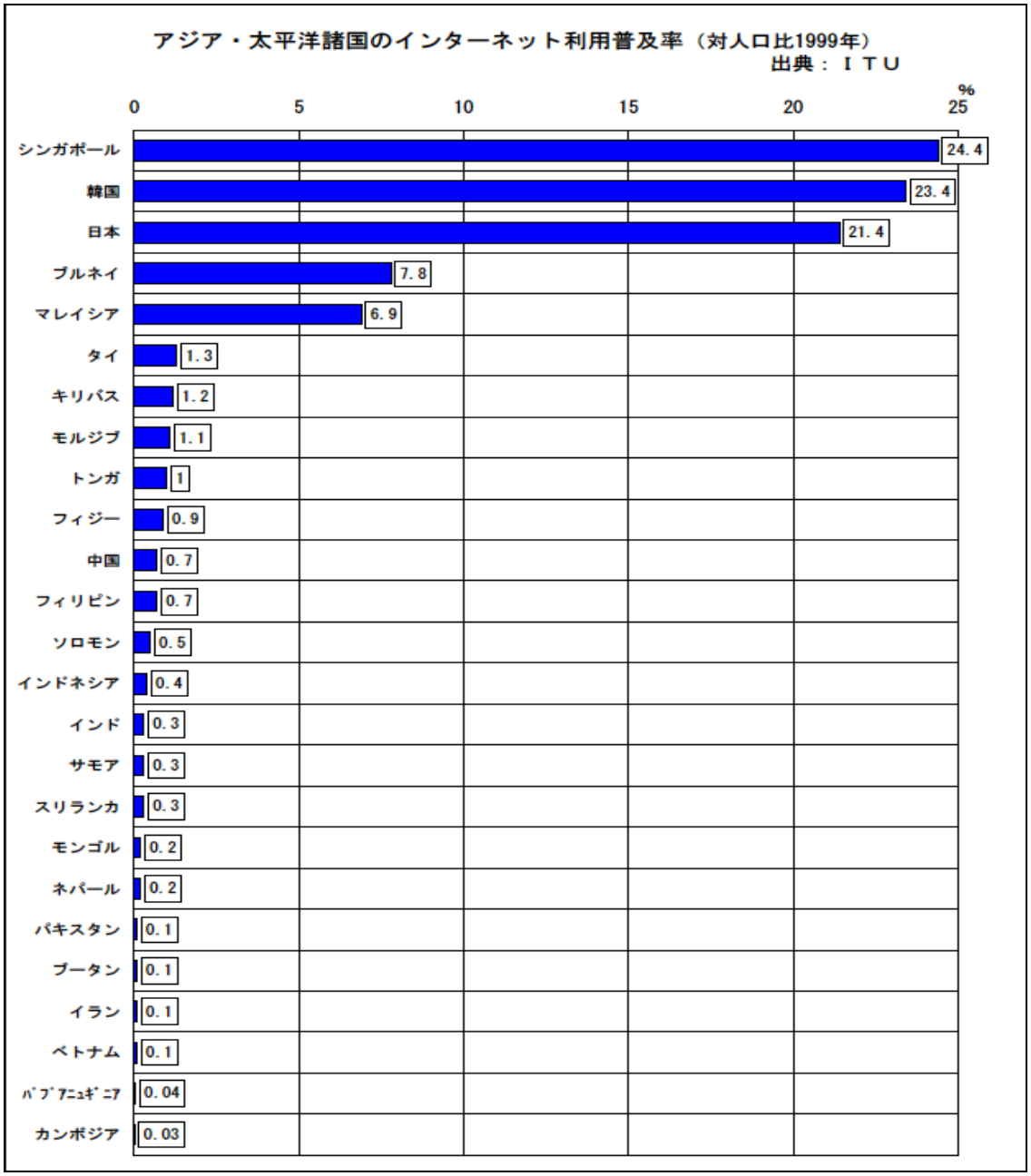


図 3.1-11

【参考文献】

資料1　アジア開発銀行発行

資料2　ATIP TECHNOLOGY SEMINAR

Prof. Kenneth L. Kraemaer

Director, Center for Research in Information Technology & Organizations

Univ. California

ISS(社会情報システム)レポート第 13 号より （2001 年 12 月発行）

3.1.4 韓国のオブジェクト指向事情

3.1.4.1 概要

韓国においては、ここ1～2年で、ビジネス・システム分野においても、オブジェクト指向開発が急速に進展しつつある。

韓国でのオブジェクト指向の特色としては、

- CBD（コンポーネントベース開発）を非常に重視していること
- UML、MDA（モデル駆動アーキテクチュア）等、OMG規格を積極的にとりいれていること

が挙げられる。

従って、主眼は、最終的な実装まで含めたシステム開発・保守の生産性・品質向上にあり、システム開発とは独立にBPR的観点から業務を見直し・再定義するという観点でのビジネスオブジェクトへの取り組みはあまり感じられなかった。

CBDを適用したシステム開発事例としては、IBM韓国やCBDの先進的ベンチャー企業であるNexgen社等により、サムソン生命を手始めに、韓国輸出入銀行、国民銀行（開発中）等、9金融機関での適用されているとのことであった。ただし、稼働しているのは韓国輸出入銀行等の単機能銀行の比較的小規模のシステムであり、国民銀行のシステムに関しては、3年をかけて開発中とのことであった。また、CBDでの開発といっても、各プロジェクト内で、初めてコンポーネントの切り出しを行っているようで、コンポーネントの再利用も企業内にとどまっていた。そのため、コンポーネントの流通・再利用による開発・保守の生産性・品質の向上というオブジェクト指向開発の本来の目的がどこまで達成されているかについては、今後の評価を待つ必要があるという印象を受けた。

また、OMG規格の積極的採用という意味では、Nexgen社は、MDAをベースにしたコンポーネントのフレームワークやツールをすでに提供していて、上記の9金融機関での事例のいくつかにおいても適用されているとのことであった。

Nexgen社では、さらに進めて、コンポーネントベースに業務別パッケージソフトウェアの発売をはじめていた。ただし、このパッケージにおいても、導入にあたっては、現実には、30％程度のカスタマイズが発生しているとのことで、コンポーネントベースにより、カスタマイズの容易性・保守性への将来的な悪影響の回避がどの程度実現されているかは、やはり、今後の評価を待つ必要があると感じられた。

3.1.4.2 進展の背景

このように、韓国でオブジェクト指向開発が急速に進みつつある主な要因と考えられる3つの背景について、以下に説明する。

3.1.4.2.1 経済危機

1997年の経済危機以降、金融機関を初めてとして多くの大企業が淘汰・合併に追い込まれた。この淘汰・合併を気に、新たに再生した企業において、システム費用の削減の要請が強まった。韓国においては、従前から、ホスト系のシステム資産・ノウハウの蓄積があまり進んでいなかったこともあり、このシステム費用削減の要請は、ダウンサイジングによるシステム再構築の流れを生み出した。また、ちょうどこのタイミングが Java の普及期にもあたり、いくつかの金融機関において、Java ベースでの開発となり、その中で、オブジェクト指向開発が取り入れられるようになっていった。

3.1.4.2.2 国策およびＫＣＳＣの存在

経済危機以降、国内経済復興策の一つとして、情報処理産業の振興・育成が図られ、その一環として、1999年に、韓国情報通信省により、ソフトウェアコンポーネント産業育成方針が掲げられた。これに呼応する形で、同年、KCSC（韓国ソフトウェアコンポーネントコンソーシアム）が設立され、情報通信省との密接な関係を保ちつつ、韓国でのCBDの普及・促進の中心的役割を担っている。現在、政府関係システム開発の発注の80％は、CBDによる開発によることを発注仕様書上明記しているとのことである。

韓国が情報処理産業の新興・育成においてCBDに重点を置いている背景には、韓国が労働集約的な意味での価格競争力では中国・インド等に対抗できなくなりつつあり、技術的な優位性が必要とされている現実もあるように思われる。

KCSCは、サムソン情報システム、韓国IBM等の大手ソフトハウス・ベンダーおよび前述Nexgen社のような先進的ベンチャー企業や大学・研究機関等約120法人から構成される産学共同機関で、わが国のCBOP(ビジネスオブジェクト推進協議会）に相当する組織であるが、ビジネスオブジェクトというよりは、国策に沿って、CBDによる実装に明らかに力点をおいている。

KCSCによれば、韓国のソフトウェア開発従事者約13万人のうち、10％程度がCBDに携わっており、また、ホスト系、クライアント・サーバ系、Web系システムの比率は、概ね、6:3:1で、CBDはWeb系システムの開発を中心に適用されているとのことであった。

なお、KCSCの18副会長法人の一つであるNexgen社は、OMGのメンバでもあり、OMGの仕様策定にも関わっているとのことであった。

3.1.4.2.3 国民性

従来の開発手法と大きく文化の異なるオブジェクト指向開発が韓国において大きな抵抗もなく受け入れられつつあるのは、以下のような韓国の国民性によるところも大きいと思われる。

- あれこれ考えるよりもやってみて、走りながら考えようとする進取の気性。
- デファクトであれデジュールであれ国際的標準があれば、それを素直に取り入れようとする許容性。

また、これまでの韓国では、システム開発において、十分なドキュメンテーションもなく、また、開発完了後は開発要員も散ってしまい、保守が困難となることが多かったとのことで、それに対する反省が、CBDに対する熱意になって現れているようにも見受けられた。

3.1.4.3 韓国におけるビジネスオブジェクトの実際

前述のように、韓国におけるオブジェクト指向の取り組みは、システム開発とは独立な業務の見直し・再定義といった観点でのビジネスオブジェクト的視点はあまりなく、あくまで、MDA、CBDによる開発・保守の生産性・品質の向上に主眼がおかれている。

開発の上流工程での要求仕様の取りまとめにおいては、UMLが使用されていて、主に、ユースケース図、クラス図、イベント・シーケンス図が使われ、一部複雑なものに関しては、状態遷移図も用いられているとのことであった。

また、この段階においては、業務のエキスパート、Rational Rose 等のモデリング・ツールのエキスパートに加え、コンポーネント切り出しのエキスパートの3者が一体となって作業を行い、CBDに繋げるとのことであった。

ビジネスオブジェクトの明確な捉え方は定かではなかったが、MDAの中のPIM（プラットフォーム独立モデル）をUMLで記述した際のクラスを、一般的には、「ビジネス・コンポーネント」ないし「ビジネスオブジェクト」と呼び、さらに、それらを機能単位に集約したものを、「ビジネス・プロセス・コンポーネント」ないし「ビジネス・プロセス・オブジェクト」と呼んでいるようであった。ただし、この「ビジネス・プロセス・コンポーネント」のことを「ビジネス・コンポーネント」と呼ぶ場合もあるようで、用語としては、統一的に明確な使い分けが必ずしもなされていなかった。

具体例としては、「顧客」、「従業員」、「在庫」といったものが「ビジネス・コンポーネント」であり、これに対し、これらを含んだ「在庫管理機能」といったものが「ビジネス・プロセス・コンポーネント」とのことであり、「ビジネス・プロセス・コンポーネント」は、粒度的には、ユースケースに近いものと想定される。Nexgen社でのコンポーネントの構成とその名称を以下の図に示す。

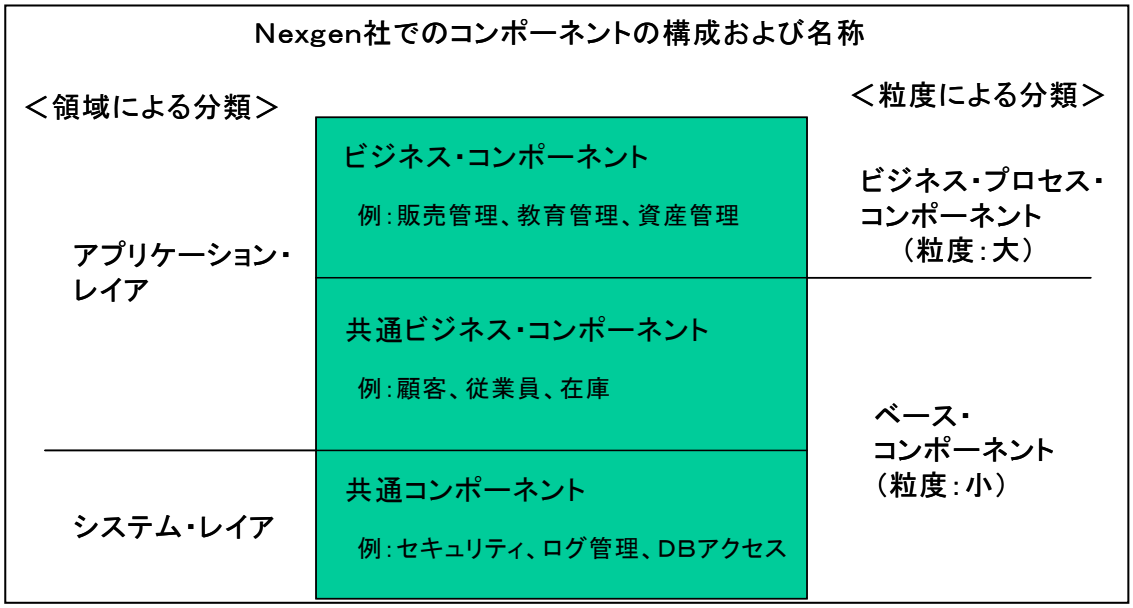


図 3.1-12

作成されたPIMモデルは、EJB での実装が中心のようであったが、例えば、パフォーマンスが重視される場合には、実装方式にはこだわらず、例えばホスト上に実装されるような場合もあるとのことであった。なお、CORBA に関しては、パフォーマンスの問題から、あまり利用されていないとのことであった。

3.1.4.4 韓国のオブジェクト指向事情

米国は、自国の技術をアジアにおいて展開する前に、韓国においてテストマーケットを実施する。このような事情は日本にはない。この事実は、新しい技術が日本よりも前に韓国に伝わることを意味しており、ITの先進技術についても同じことが言える。Nexgen 社のイ・トクサンCEO によると、日本の経済産業省が2003年に韓国を訪問した折、コンポーネント技術に関する知識がなかったらしく、「それはどのようなものか」と質問したという。同じような質問を、韓国の役人が 3 年前にしていたので、韓国は日本より 3 年ほど進んでいるのではないかと冗談とも思えないことを述べていた。

韓国では、政府の強い指導力によって、ソフトウェア産業の発展を方向付けている。このことは、韓国の IT 産業全般についても同様である。世界一のブロードバンド普及率を実現できたのも、政府の強力な後押しがあったからだ。

このような背景があって、韓国におけるコンポーネントの標準化は政府が主導している。どのような技術の普及や標準化も、トップダウンで進めた方が、効率がよく早い。コンポーネントの標準化と普及について、韓国が政府主導で対応していることは、差し詰め慧眼であろう。

韓国は、インフラとしてのインターネットおよびブロードバンドで世界一の敷設を誇るようにな

った。この堅牢な基盤と、徹底したIT教育をもとに、KCSCを中心として、コンポーネント市場の拡大に取り組みはじめている。インターネットの次は、ソフトウェアの基盤確立に着手しようと言うわけである。米国が自由市場において自然発生的なデファクトスタンダードを形成させようとしているのに比較して、韓国の国家主導戦略は対照的である。

韓国のコンポーネントベース開発はベンチャー企業が中心となって進めている。今回訪問したネクストージェネレーション社もそのうちの一社に相当する。また、現代グループの情報システム部で 40 年間の経験を持つリー・ヨンヒ CEO が設立した e コンサルティング社も同じくベンチャー企業であるが、ここでは J2EE フレームワークに基づいたコンポーネントベース開発プロセスをコンサルティングしている。

複数の SI ベンダとeコンサルティング社は、「CBNG」(Component Business Network Group)を設立している。これは、ソフトウェア技術者と人コンポーネントとを共有するコミュニティである。CBNG は韓国政府からプロジェクトを受注し、KCSC を通じて成果物を普及させようとしている。

高度な技術力を持ったベンチャーを政府が後押しし、その成果を全体で共有する取り組みが、韓国におけるコンポーネント普及を加速させている。

一方、ベンチャータイガー社は、韓国におけるベンチャー企業のインキュベーション事業を展開するコンサルティング会社であるが、同社では技術革新型中小企業『イノビズ』(InoBiz)の育成に力を入れている。イノビズは、「イノビズ協会」による厳しい審査を通過する必要があるが、いったん審査に合格すると、政府から資金や経営コンサルティングなどの支援を受けることができる。

モデリング技術を柱とするイノビズ企業が増加の傾向にあるとのこと。ベンチャータイガーは UMTP アジア(UML モデリング推進協議会)に参画し、モデリング系のイノビズ企業を後押ししている。

日本では、上場企業が元受としてユーザ企業とSI契約を行い、その下に二次請け、三次請けと面々とつらなるゼネコン体質を持っており、ソフトウェア産業としては中小企業が大半をしめている。韓国のソフトウェア業界も日本と同様で、中小企業が大多数を占めている。

このことに気づいた韓国政府は、これらの中小ソフトウェア企業の開発生産性を上げることに取り組んでいる。

そのひとつの施策として、韓国政府はソフトウェア品質の保証機関「SQEC」(Software Quality Evaluation Center)を設置した。製品品質の認定をパッケージ中心に行おうというものだ。これにより、政府調達案件への参加も、高い技術力を持っていれば、資金面で劣っていてもできるようになっている。

また、「SQEC」は米国の品質基準であるCMMと相互認証している。これは、国内市場が小さな韓国が、海外市場を視野に入れたものと考えられる。

当然、コンポーネントを中心としたビジネスを海外へ展開する戦略であろうし、隣国である日本も韓国にとって期待できる市場として映っている。

3.1.5 企業訪問

3.1.5.1 韓国ＩＢＭ

報告者　:新日鉄ソリューションズ株式会社　鋼鉄ソリューション事業部　臼井隆宏
訪問日:2003年10月23日(木)

1. 韓国 IBM における OOT への取り組み　概要
   韓国 IBM は、従業員2400名と日本 IBM の1割ほどの従業員数であるが、IBM 共通のリファレンス等を活用した、コンサルティング業務、オブジェクト指向教育、コンポーネント技術を適用したシステム開発を積極的に推進している。今回の訪問では、韓国 IBM における、オブジェクト指向教育に関する取り組みの状況　、オブジェクト指向技術を用いた銀行システムの開発事例、韓国のある企業におけるオブジェクト指向を適用した開発へコンサルタントとして参画した事例の 3 点を紹介いただいた。

2. 韓国 IBM でのオブジェクト指向技術へのアプローチについて紹介

2.1 イントロ

- 発表者が、Korea IBM の教育機関で OOT コンサルティングをした経験から、その取り組みについて紹介している。
- 韓国IBMでは、1997年～1998年にかけて C++を利用した開発に取組んできた。この頃からインターネットの普及と共に Java 言語の利用が始まった。
- オブジェクト指向、Java の急激な広がりの背景には、韓国社会へのブロードバンド普及がある。
- 米国 IBM との交流を行うことにより、オブジェクト指向、Java、インターネットに関する情報収集に努めてきた。
- アメリカにて経験を積み韓国に帰国後、ベンチャーの CEO となった若い世代の人たちが積極的にオブジェクト指向を適用してきた。
- オブジェクト指向、Java の普及に伴いこれを教える人が必要となってきたが、ウォンキムさんという人が中心となり教育活動を推進した。

2.2Java の特徴

- Java はフロントエンド(画面表示を含むこれに関連する処理と考えられる。)向けを中心に適用され、バックエンド(バックグラウンド処理、バッチ処理のことと考えられる。)向けではない。

2.3CBD(Component Based Development)

- IBM はオブジェクト指向プロジェクトを立ち上げるときに、CBD を適用して、顧客にどのように売るか検討する。そのときには、マネージャとセールスが参画する。
- CBD は、企業全体のシステムに適用できるわけではない。現時点では、部分的な適用のみ。

2.4 オブジェクト指向プロジェクトの推進時の局面

- オブジェクト指向プロジェクトを推進する時の局面を下記の様に分類。
  ① ビジネスプランニングトラック→顧客を説得する。CBD のメリットを説明する局面。
  ② プロジェクトパフォーミングトラック→プロジェクト推進の手順を考える局面。
  ③ プロセストラック→ウォータフォールモデルや UP をどのように適用していくか検討する局面。
  ④ アーキテクチャトラック→コンピュータキテクチャを検討する局面。
  ⑤ コンポーネントマネージメントトラック→再利用可能なコンポーネント抽出適用を検討する局面。

2.5 オブジェクト指向プロジェクトの体制

- オブジェクト指向プロジェクトのチーム毎の役割分担を下記の様にしている。
  ① プロジェクトマネージャ→プロジェクトの統括
  ② アーキテクト→システムアーキテクチャ検討チーム
  ③ ビジネスアナリスト
    1). ビジネスアナリスト→顧客業務分析チーム
    2). コンポーネントアナリスト→顧客業務よりコンポーネント抽出するチーム
  ④ デベロップメント→実装チーム
  ⑤ コンポーネントアーキテクト→コンポーネント設計・製作チーム
  ⑥ コンポーネントリユーザブル→再利用コンポーネントの管理・適用推進チーム
- 上記体制は、3 イテレーション規模（1 イテレーション 6 ヶ月程度の工期を必要とする開発）の開発以上で効果を発揮する。

2.6 オブジェクト指向開発推進者への教育

- オブジェクト指向の教育者は、教育方法や教育の効果についての理解が必要。
- デザイナーは、オブジェクト指向言語のプログラミングを経験し、オブジェクト指向言語の仕組みの理解が必要。
- プログラマは、オブジェクト指向や CBD 等についての理解が必要。
- オブジェクト指向関連技術者は、一般的に非常に忙しいが、チーム開発などの新しい発想、新しい技術を伝えていく必要がある。これには、電子教材や教育カリキュラムの

短縮化が有効。

2.7 導入

- 企業の基幹業務全てにオブジェクト指向技術を適用するわけには行かない。
- 細かな業務に適用してみて顧客に評価してもらうのが良いと考える。

3. オブジェクト指向技術を適用した銀行プロジェクトの事例

3.1 内容

- オブジェクト、コンポーネントを活用した銀行システムの紹介
- 世界 TOP30の銀行
- 始めは自社内でビジネス・コンポーネントを利用した開発を進めたが数億円の損失をした。その後ビジネスパートナーに IBM を選びプロジェクトを進めた。
- Component Based Business Modeling
- 機能ではなくプロセスに焦点をあてて開発を進めている。
- 200アプリケーションをパターン技術などの適用で再構築。
- 260ユースケースを製作(Rational Rose define)
- 68ビジネス・コンポーネントを抽出。
- IBM 全体でビジネスオブジェクトに関するリファレンスを持っており、これをベースにプロジェクトを推進した。

3.2 プロジェクトの体制と成果物

- Component Analysis Team -> Business Component の抽出を行う専門のチーム。
- Business Modeling Team -> Business Model を検討するチーム。UP を適用。
- Technical Architect Team -> Computer Architecture を 検討するチーム。
- 上記3つのチームが並行してプロジェクトを推進。

成果物

- Business Component
- Business UML
- Business Object Model
- Architecture

の4つが製作される。

3.3 実装について

- アプリケーションの要件により要求が異なってくるので、パフォーマンスを要求されるような場合は、ORACLE VS DB2、UNIX VS IBM ホスト などでベンチマークテスト

を行う。(基本的には J2EE ベースであるが・・・)

4. オブジェクト指向プロジェクトへのコンサルタントとして参画した事例
   - 韓国 IBM がコンサルタントとして参画した事例紹介
   - WEB ベースのシステムで、工期は4ヶ月。
   - OOT を適用した開発を進めるために、IBM では、顧客への OOT の教育及び顧客との協業によるシステム開発を推奨・提案している。
   - 韓国 IBM 独自のパターン、フレームワークを開発時に適用することを検討し、不足している部分は開発という手法をとっている。
   - EJB、JSP を適用する場合、DAC(Data Access Control)を使用している。
   - 今回のプロジェクトでは、2イテレーションにて改善点の洗い出しを行った。
   - 韓国 IBM がパターンを提示したら、顧客が Java を用いて開発を行えるようになった。(パターンは重要)

5. 総評

   オブジェクト指向技術を適用した開発において、プロジェクトメンバの役割が細分化され役割を明確に定義化していること、それぞれの役割に対して、教育プログラムが存在していること、顧客にもオブジェクト指向教育を行い、共に開発を推進するという話が印象に残っている。IBM 全体でのリファレンスがあると考えるが、韓国 IBM においても徹底されており、IBM のスピリッツを感じた。

   また、CBD の積極的な活用も印象的である。CBD の活用は、韓国情報通信省の産業育成方針と聞いているが、日本では、このような取り組みを聞いたことがなくやはり印象的である。

3.1.5.2 NexGen 社

報告者：NTTコムウェア株式会社　システム本部SE部　松崎弘人
訪問日：2003年10月24日

1．会社概要

- 2000年3月創業の新しい会社
- 従業員は30人ほど
- 業種は、
  ・コンサルティング
  CBD(Component Based Development)の方法論
  CBD での AP 開発
  ・教育サービス
  ・ソリューション製品の開発販売
  ・SI の実施（金融業向け、通信業向け、製造業向け）

2．プレゼンター　Tok Sun Yi CEO

韓国コンピュータ協会（KCC）の会員である。
政府がどのような政策をとろうとしているかの方針を聞くことができるメンバである。
韓国で数人いる韓国最初のコンピュータプログラマの一人である。
CDC 社（Control Data Corporation：米国のメインフレーマ）が韓国に進出した1967年に、韓国人プログラマとして最初に採用したメンバである。
CDC 社に勤務し、韓国で仕事してきたが CDC 社が韓国から撤退するときに米国に渡り十数年にわたり米国で働いてきたものである。数人いる韓国人プログラマ第1号のひとりであり、韓国コンピュータ界の草分けの一人である。

3．説明内容

3．1 NexGen 社のビジョン

CBD(Component Based Development) Engineering と Architecting Consulting のリーダとなることである。技術は「オブジェクト指向」→「コンポーネント ベース 開発」→「モデル ドリブン アーキテクチャ」へと進化していったが、NexGen 社は当初からこの流れで進めてきた。

3．2　NexGen 社の CBD フレームワーク

NexGen 社の CBD は統一されたアーキテクチャのうえに、CBD 方法論とコンポーネントフレームワークが乗っている形態となっており、これに沿って、個々のコンポーネントパッケージが構成されている。

NexGen 社の CBD フレームワークの概念図を下に示す。

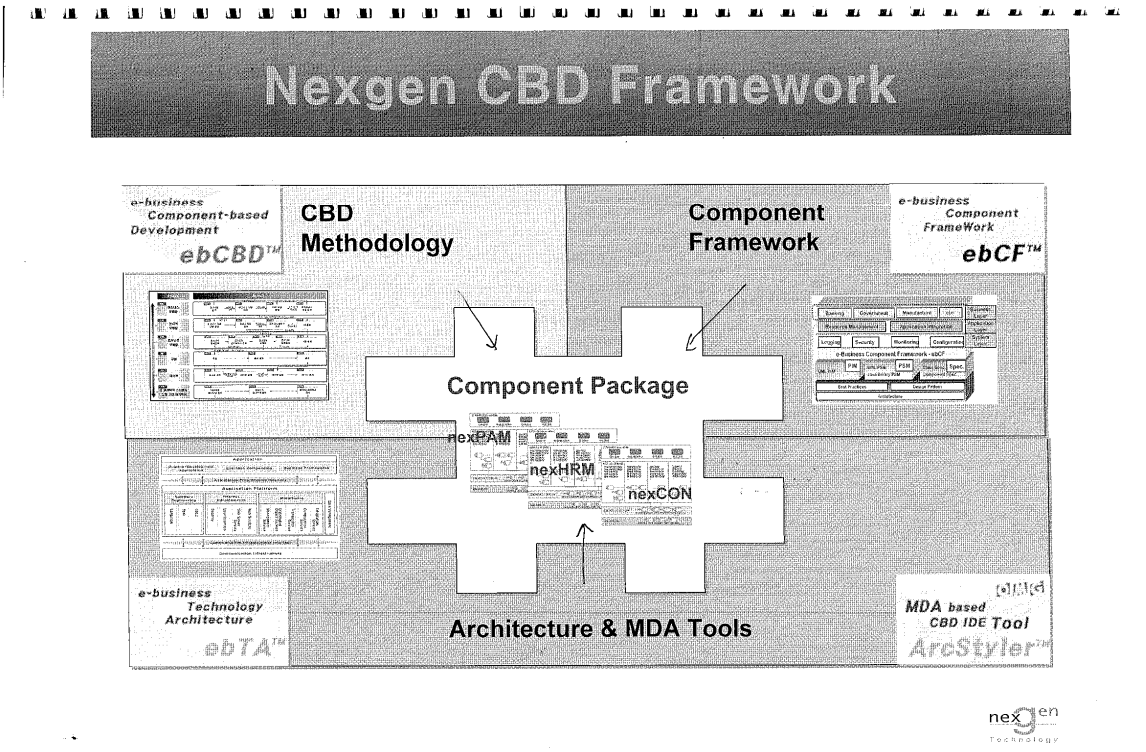


図 3.1-13　Nexgen 社ＣＢＤフレームワークの概念図

3．3　ソリューション

CBD のソリューションセット

・ebTA・・・・・Technical　Architecture

・ebCBD　・・・・Development Methodology

・ebCF・・・・・Component Framework

・ArcStyler　・・MDA Development Tool

コンポーネントパッケージ

・nexPAM　・・・IT Asset Management Component

・nexHRM　・・・Human Resource Component

・nexCON　・・・Construction Management Component

3．4　質疑応答概要

3.4.1 政府のコンポーネントベース開発に対する支援

NexGen 社は、昨年、コンポーネントベースの開発（CBD)として、政府から3つのプロジェクトを受託しており、費用は政府が全額支払っており、政府として積極的にCBDを支援している。最近は政府が発注する仕事の80％がコンポーネントベースとなっている。

3.4.2NexGen 社の海外への取り組み

NexGen 社はまだ、韓国内でのビジネスのみでまだであるが、当然ながら海外進出の機会をねらっており、海外への進出意欲は旺盛である。

3.4.3 日本と韓国のオブジェクト指向への取り組みの差

この夏に日本の経済産業省の情報処理を担当している人間が韓国にきて、意見交換する機会があり、そのときの様子では情報処理を担当している人間でもCBDについてはあまり詳しくなかった状況が韓国の3年前の状況とよく似ている。

3.4.4 日本と韓国のコンピュータ環境の違い

韓国はメインフレームがあまり普及せず、いきなり UNIX での構築が始まった。日本のようにメインフレームでの歴史が無いため、メインフレームへの対応が必要なかったため、移行が早かったのではないかと思われる。

日本では開発する人間が客先に常駐してスクラッチで開発することが多いが、韓国ではパッケージをベースとしてあとはユーザ毎のカスタマイズという形態での開発であるため、客先での開発が少ない。業務優先の考えからいくと、客先で開発するのがよいかもしれないが韓国ではあくまでもパッケージベースであったようである。ソフトウェアの変更に際しても、変更分のソフトウェアをオンラインで供給する形態をとっているそうで、客先に常駐するとか、作業自に客先に出向くことが普通と思われている日本とは大きな違いである。

日本では OS やミドルは日本語化してから使うことが普通であるが、韓国はマーケットが小さいことから、OS やミドルではなかなか韓国語化されない。このため、英語版のソフトをそのまま使ってきたようである。

韓国には「取りあえず、世界標準を受け入れる」という風土がある。これは世界が韓国にはなかなか合わせてくれないのなら、韓国から世界に近づいていこうというのだろう。この辺が韓国のソフトウェア産業が世界を見て仕事をしている土台なのかもしれない。

オブジェクト指向や新技術などが韓国に入ってくるのに時間はかからず、世界がその方向に向かっているのなら韓国もそれを受け入れようというのだろう。

CBD には、工程・品質・ドキュメントが重要であるが、韓国は英語をそのまま受け入れたため、ドキュメントを英語に書くということも違和感はなく、普通に英語で書いてきたようであり、日本語でドキュメントを書くのが普通の日本とは大きく異なっている。それが、海外進出の際には大き

なメリットとなっている。

日本は、OS やミドルといったソフトウェアおよび最新技術の情報なども、一旦日本語化してから普及するのでそれだけで1年くらい遅れてしまう。ソフトの最新バージョンも1年くらい遅れる。普段、最新バージョンの日本語版はいつリリースされるのかとミドルのメーカーに問い合わせていた自分を振り返るとそのようなことがタイムラグとしてきいているのだなと実感させられた。

3.4.5NexGen 社でのコンポーネント開発

NexGen 社のパッケージである「IT PAM」では、ビジネス・コンポーネントが15あり、小さいコンポーネントはたくさんあり、業務に依存しない部品として用意されたものを使ったり、あらたに作って部品とし登録したりしている。

CBD で開発するといっても、ユーザによって業務の違いや要望が異なることから、実際の流用は70%くらいであり、残りの30%はどうしても手が入ってしまうそうである。

NexGen 社としては、コンポーネントだけの単品での販売はしていなく、基本をパッケージで収めて改造分、変更分をコンポーネントとして販売する形態をとっている。

コンポーネントの流通はまだ社内のみであり、米国で行われているような他の企業へのコンポーネントの販売はまでできていないで。韓国内の他の企業でもコンポーネントの社外流通は行われていないそうである。

コンポーネントの流通という考えはまだない。

4. 所感

韓国のプログラマ第1号ということで、メインフレームでの開発が長く米国でも長い間ソフトウェアの仕事をしてきたからか、ソフトウェアのアーキテクチャを重視していることが伺われた。あらかじめアーキテクチャをちゃんとしておかなければ、CBD で開発しても効果を発揮できないとの考えを持っているようである。

他のベンチャー企業との意見交換ではどちらかというと「まず作ってみる」という発想が多く、作ったものを何とか売りたいとの考えが見え見えであるが、この NexGen 社では CEO の発想だろうか、まず社員教育をしっかりやって、コンピュータに関する基礎教育をしっかり受けたものがしっかりした指導者のもとで実践を積んで初めて、自分の発想で開発するという方法をとっているようである。一見普通のことであるが、今の韓国ではスピードを重視するあまりに大学の学生のうちに、もしくは卒業するとすぐに実践経験をつむ暇も無いまま、実用ソフトウェアの開発を行いベンチャーとして活躍することが多いなかでは、堅実で着実な製品開発をやっているように思える。

3.1.5.3　　トライジェン社

報告者：NTTコムウェア株式会社　システム本部SE部　松崎弘人
訪問日：2003年10月24日

1. 会社概要

- 1980年創業の PC 製造会社
- 年間400万台の PC を製造し、世界中に OEM で製造販売している。
- 日本向けとしては1998年から2001年までソーテックの PC は
- 100％TriGem 社製であった。今は、HP などに供給している。

2. プレゼンター　Y.T Lee 会長

韓国のデータ通信サービス会社（DaCom）を創業したときに社長を務めた。日本では国内（NTT）、海外（KDD）と分担してきたが、韓国は、音声（KT）とデータ通信（DaCom）という分担でやってきた。

1984年からUNIX での開発を進めており、1988年のソウルオリンピックはUNIX ベースのシステムで行った。それにあわせて、電子政府も推進してきた。韓国の政府に直言できる韓国のコンピュータ業界のリーダ的存在である。

3. 説明内容

今回は、TriGem 社の会長というより、韓国の情報産業のリーダとしてのプレゼンテーションである。韓国は、現在高学歴者の失業が多いことが政府の重要な課題となっている。韓国では毎年50万人が新たに大学を卒業しているが、十分な就職口がないのが実情である。

韓国のような国が生きていくためにはソフトウェア産業しかないと私は考えており、私は韓国政府に対して次のことを提言している。

「10年で10万人のソフトウェアエンジニアを育成しよう。5年間のシステム開発の実務を経験させて育成する。必要な費用は政府が負担し、外国の企業には無料で韓国のエンジニアを使ってもらうのだ。外国の企業はシステムの企画をして実際の開発は韓国でやってもらう。そうすることによって実システムの開発経験を積んだソフトウェア技術者を育てることができる。」世界中から次世代のシステム開発を韓国でやってほしい。各企業あたり約1000人の高学歴のソフトウェア技術者を無料で使えるのだから、進出企業にも損はないはずだ。

条件は外国の企業が企画・設計し、実際の開発を韓国（ソウル）でやってもらうだけだ。韓国はそれによって技術力のあるエンジニアを育成できる。まだ、韓国政府は10万人の給与を負担する財政的な課題があるためあまり乗り気ではないが、それくらいしなければだめだ。

5. 所感

さすがに韓国の草創期からの情報産業のリーダをつとめ、今も大手の PC ベンダを経営しながら韓国の情報政策に強い影響力を持っている人だけあって、話が自社の一時の利益ではなく韓国という国をどのように建て直すかという観点で熱心にプレゼンテーションしてくれた。熱い思いがよく伝わってきた。

もし、１社あたり１０００人もの高学歴のエンジニアを無償で５年間使わせてもらえるのであれば日本の会社としても一考の余地はあるであろう。コミュニケーションとしても韓国の若手は英語が堪能であり、オブジェクト指向でのパッケージ開発であれば、日本が韓国とパートナーを組む可能性はある。韓国と組むことは日本にとっても利益はあると考えられるが、いまのままでは日本は軽く抜かれるのではと思われるほどのパワーを強く感じた。

3.1.5.4　　ＹＤＣ社

報告者：株式会社ベストワーク　ビジネスシステム統括部　櫻井努
訪問日：2003年10月24日

1．会社概要

- 85年設立
- 資本金8億ウォン（約8千万円）
- 従業員28名
- システムインテグレーションおよび、ソフトウェア、ハードウェアのソリューションの事業展開をおこなっている。NEC、東芝、三井物産、SAPなどと協力関係にあり、海外向けのマーケッティングを事業ビジョンの柱の一つに据えている。特に日本市場に関心を持ち日本向ソリューションを自社保有し、日本に支社を開設している。（豊島区南大塚2－29－9）
- （http://www.ydcdigital.com/japan/index.html）

2．プレゼンテーション内容

事前質問内容の要点を回答書にて配布いただいた。以下にその要点および質疑の内容をまとめた。

3．韓国のメインフレームの状況

YDCの見解では、一般企業では殆どがUNIXシステムを用いたシステムを使用しており、メインフレームはあまり使われていない状況を照会している。

金融機関においてもメインフレームの依存度を減らすべく多くの研究を行ってきたが97年のIMF経済危機によってダウンサイジングの集中的投資を行うこととなった。

現在ではやはりUNIXを中心としたシステムが稼動中もしくは開発中とのことであった。事例として大手銀行ではCBD方法論を用いたシステム、GBS、ほかUNIXでのオープンシステムが中心となっており、また、地銀においては殆どがUNIXオープンシステムとなっている状況を紹介している。

汎用機からUNIXシステムの移行によりコストパフォーマンスは50％向上しているという認識である。

４．システムのライフサイクル

システムのライフサイクルに関する考え方を次の費用対効果、品質、開発方法論という3つの視点から照会している。

4.1 費用的効果

開発費用の捉え方は競争力の強化という将来の利益という考え方で捉えている。

売上の1～3%、利益の10%の投資という捉え方が出来るであろうとしている。競争力の強化が最大の成功要因となることから、システム導入のスピードアップが重要であり、特に銀行はIMF時の影響から経営手法が米国式となり、この影響は大きいとしている。

4.2 品質

今までのシステムとしては、品質より生産性という考え方が主流であった。

ドキュメントは簡素化し、とりあえず開発し保守にて品質の維持確保を行う。初期段階での開発方法論、品質計画などはあまり重んじずシステム運営によって維持していくような傾向であった。しかし、やはり最近では、大手企業中心に品質管が重要性されCMM、SEIなどの導入を行っている。因みにYDC社はCMM2～3のレベルであるとしている。

4.3 開発方法論

CBDは検討される事が多くなってきてはいるが、YDCの見解では「パイロットプロジェクトとしての開発が盛んになってきている」というシステム発展段階として捉えている。

先にも述べたように、新しい技術の取り入れによる競争力の強化というシステム投資に対する考え方を背景として、経営トップの世代交代が頻繁で、トップの考え方でのシステムの再構築が行われることから、特に80年代を境にして4GLによる新しい開発手法が主流になったとしている。また、設計、モデリングとしてはUMLを使用しているが、ユースケース、クラス図、シーケンス図以外のドキュメントフォームは殆んど使用されていない。開発方法論的にはUP(Unified Process)による開発方法はあまり使用されておらず、ウォーターフォールモデルが主流であるとしている。

5. コンポーネントビジネスの状況

韓国訪問他社においてはオブジェクト指向のビジネス事例としてCBD方法論を用いたフレームワークによるインテグレーションが主流になりつつあるという報告事例が照会されているが、YDCにおいては業務の一部としてEJBを利用して開発しているという内容であった。

また、開発コンポーネントの再利用に関してはあまり行われておらず、YDC社内による開発においても再利用は無いと言い切っている。これは先の費用対効果の中で触れたように、新しい技術を採用する事でシステムの競争力を維持するという要因から、システムインテグレーションにおけるコンポーネントの再利用という局面は発生しないという事が理由となる。

6. オブジェクト指向の現状

オブジェクト指向の現状の捉え方としてその教育、設計、開発基盤という3つの視点から照会している。

6.1 オブジェクト技術の教育

JAVA、c＋＋などを中心とし学習されている。大学では開発言語中心の論理的教育が中心となり、実務面ではあまり効果がないと指摘している。在学中、あるいは企業に働きながら専門学校などで実務を学ぶ人が多いとうことであった。

また、IMFによるハードランディングは技術者にも環境の変化をもたらした。技術スタイルは米国式となり、派遣をベースとした作業となった。技術者には終身雇用という形態はなくなっているようである。

YDC社は日本市場が大きなマーケットエリアとなっていることから、日本語の習得においてかなり力を入れている。日本語科の卒業生を採用し教育を行い、開発においても日本文化を取り入れ、日本語が大前提という事であった。また、現地にはオンサイト要員を在駐させている。

一般的に日本語によるオブジェクト指向設計の不相応さが問題の向きになることもあるが、もともとプロダクツにはハングル対応のものが無いという事から、開発においてはハングルを使用しないで実施することが慣例となっているようであった。ただし、日本語のYes，Noの曖昧さはやはり不明瞭であると指摘された。

6.2 オブジェクト指向設計

YDC者の特徴から実務面での捉え方となると思われるが、オブジェクト指向での設計ではクラス図、ユースケース図、シーケンス図が使用され、他UMLドキュメントスタイルはあまり使用されていない状況である。ツールとしてはRationalRose社が使用されている。

6.3 開発基盤

JAVAが中心でありc＋＋、ORACLE、Infomix、等が使用されていると言うことであった。JAVAはビュー関連、c＋＋はサーバロジックを担っているとしている。

ドキュメントはUMLの産出物が使用されているのみであるとしている。

7. 政府支援策

政府の支援策としてコンテンツ開発費用、研究費用などが低利で融資される。しかしながら、審査基準、支援金額共に満足の行くものではなく、多くの会社は利用できない状況にある。

インド、中国、他アジア諸国に対する韓国のIT戦略としても明確になっていないとしている。

YDC社独自の見解では、特に中国は脅威と感じており、単価も従来の65％程度に落ち込んでいる。プログラム設計・製造フェーズにおいてはやはり利益は期待できない。プロダクツの展開（システムソリューション）にシフトしていかざるを得ないとの事であった。

８．所感

YDC社が特に日本市場を意識した会社であることを差し引いても現状の開発現場の本音を聞けたと言う感触であった。

最新の開発方法論であるCBDもYDC社の見解では、やはりまだ一般的な手法には至っていないようであった。NexGen 社での CBD におけるパッケージの流用度は顧客ごとに概ね70%という回答を得たが YDC 社では再利用は無いとの回答であった。(新技術の採用によるシステムの競争力維持)

競争力の強化という将来の利益を先取りするという考え方は、一つの割り切りである。韓国の国民性がはっきりとこの意思を示しているようである。

新しい技術を採用する事によりシステムの競争力を維持するということが韓国での一般的な考え方であるならば、やはりソフトウェアテクノロジーという側面においては韓国の方が一歩先んじているというのが事実であるし、そう感じている。

しかし、企業全体の競争力強化のためのシステムが最大のシステム構築成功要因と捉えるのが本来の姿であり、組織本来が持つビジネスモデルが競争力を維持する源泉である。システムはその骨格を維持する一つの資源である。

競争力の先取りという考え方は、IMF、経済危機からのハードランディングを乗り切った韓国経済が出した一つの結論であることは否定できない。費用対効果をじっくり検討するか、決断を早くし先行して競争力を重視するか(先々の時間を買うという捉え方)、韓国は後者を取っているようである。

BOの捉え方、あるいはDOA、OOAそれぞれを吟味することでシステムへのアプローチを行っていくという考え方よりも、その発展の良し悪しに関わらず、最新の技術を採用することによって必然的にそうなっていた。というのが現状であるような印象を受けている。

中国に対する脅威はYDCにおいては日本と同様で深刻に受け止めているようである。生産性の高い人材獲得のためモンゴルからの人材を注目しているとの考え聞き印象的であった。

プロダクツの展開(システムソリューション)にシフトしていく方向性を打ち出しているが、これこそBOの流通、CBDを活用する分野であろう。

日本企業からの訪問という事で、プレゼンテーション説明の参加メンバも当社の主要メンバから営業メンバまで総員の体制であった。当社システムの売り込みを意識し、製品のデモンストレーションがその半分を占めていたような説明内容であった。

上記内容は製品説明の合間に、当方の本来の質問を折りませて行ったような状態であった。

3.1.6 総括

「3泊4日程度の滞在で本当の事はわからない」

「葦の蕊から天井覗く」の類で、断定的なことはいえないが、いくつかの印象深い聞きかじりを並べ、帰国後にいくつかの本、インターネットのコンテンツから学んだ事項を並べて感想を述べてみたい。

3.1.6.1　　日本流と韓国流

「オブジェクト開発手法は日本に比べて何年進んでいますか？」との私たち訪問団の質問に対してA社社長は「3年程度進んでいると思います」と答えた。
「それは何故そのように感じておられますか？」鋭い私たち仲間からの質問である。
「先日も日本政府調査団がこられていろいろ質問されましたが、その質問は私たちが3年前に悩んだことのある質問だったからです」A社長の自信のある答弁である。
「何故日本より先にすすんだのでしょうか？」これまた別の仲間からの厳しい質問でありなんと答えるだろうか？身を乗り出して聞き耳を立てた。
「日本は進んでいるアメリカの技術資料を、まず日本語に訳します。それの対して韓国の私たちは、そのまま英語のままで使います。まず新しい情報が入ってきたときにそこで遅れさらにバージョンアップなどが発生するたびに遅れます」この答えには英語が必ずしも得意でない私は頭をさげざるを得なかった。

このやりとりを、そばで聞いていた若い韓国人副社長が「社長ちょっと待ってください」と手を上げて発言を求めた。「韓国は昔から外から入ってきたものを、そのまま受け入れるのは早いし上手いのです。でも日本はよく考え、新しい技術が本ものなのか？よく見極め日本流に直して使います。10年たってみたらやっぱり日本の方が進んでいたと言うことになっているのではないでしょうか？」お世辞と慰めが混じった発言であった。

聖徳太子の時代に「科挙，宦官制度が中国から入ってきた時に韓国はそのまま受け入れたが、日本はそのまま受け入れなかった」事から始まり、日本は日本流に修正し活用する事がその後も江戸時代、明治時代と一貫して引き継がれ手いることは、歴史的にも証明されている。日本文化の源は東西の良いところを識別して取り入れているものがたくさんあるが韓国は素直にまず採用するの文化を持っているようである。

したがってインターネットの採用も早い、オブジェクト設計の採用も早いと言うことになる。オブジェクト設計のような技術はじっくり見極めながら活用する方が良い。最初の導入の早さなど気にしてもしょうがない。負け惜しみのところもあろうが、この日本流発想法は自然と日本人が身につけた貴重な習慣のひとつである。

しかし参考にすべきところは素直に学んだ方が良い。この「日本人はいちいち日本語に訳す」と言う話を、友人にしたところ、実は私も韓国の製鉄所を見せてもらったが、現場の作業員が使うコンピュータの端末機の表示がすべて英語になっていたので「韓国語には直さないで大丈夫なのですか？」と質問したら「画面に現れる文字はタイトル程度であるから、50文字程度覚えればすむこと」とあっさり片付けられた、と話してくれた。

韓国のIT市場は日本の1/3程度であると思われるので、いちいちハングルに直していてはビジネスにならない事もあろうが、パンフレット、画面、マニュアルなど、すべての資料を日本語に直していては、ドッグイヤーと呼ばれる流れの速いITの世界の競争から遅れることも、事実であろう。

日本独特の技術になってきた例をあげてみよう。

日本でオブジェクト設計を採用する場合でも、上流設計にはＵＭＬでなく、ＤＯＡ（data oriented approach）を使い成功している企業が存在する。

韓国で「DOAを知っていますか？」とベンチャー企業の若い社長10人に聞いてみたが誰も知らなかった。「あれDOAは日本独特の手法なのか？」と錯覚を起こしたくらいであった。

実はDOAは世界的な手法ではあるが、日本では椿正明先生や佐藤正美先生のご苦心と努力で体系化され進んだ手法である。日本的工夫でオブジェクト設計手法が進化することを願ってやまない。

3.1.6.2　　生産性と品質

「日本の企業から注文をもらっているが、もっと安く出来ないか？と注文をつけられているので、もっと安くする方法を考えている」とK社の社長が胸を張って以下のように続けた。

「日本語の出来る韓国人を仲介に使い、モンゴル人、ベトナム人を採用し現地に開発センターを設置し、プログラムを作成する。もっとも北朝鮮人まで使えればリソースには事欠かない。私の会社に仕事くれませんか？」ビジネスに熱心なK社の社長の目が＄マークに見えた。

実は韓国滞在中に数社訪問させていただいたが、すべて「生産性、生産性」「コストダウン」しか言われなかった。日本でもしばしばベンダーで起こる現象であるが「品質第一」と言う単語は、システムの世界では何故か出てこない。

仮に100％完全な仕様書が出来ていたとしても、安い労働力、未熟なプログラマを使って良いプログラムを作成することは至難の業である。

品質の定義がはっきりしないで生産を続けること自体が問題であるが、この安いプログラマを活用して作成されたプログラムのメンテナンスをさせられる日本のユーザ企業内エンジニアはたまったものではない。

「安値競争だけ」から早く「品質競争」にシステムの世界が変わることを願ってやまない。

3.1.6.3　　人材育成

韓国の情報産業の大御所である李龍兌（Lee yong- teh）韓国情報産業連合会会長にお会いする機会に恵まれたときの話である。

「韓国では若い大学卒業のエンジニアが就職難に困っている。それならソフトウェア工学を学びに年間1万人を10年間くらいアメリカに出そう、と産業省に言ったが、ちっとも予算化してくれなかった。やっと最近100人だすことになった。

日本では若い人の就職難は無いのですか？と日本の大企業の社長に聞いたら、そのような

悩みは無いと言われた。日本は良い国ですね」とさらりと言われた。

実は一週間くらいたった時の11月の新聞に「大卒は 50%、高卒は 70%の人がまだ就職がきまっていない」と乗っていた。また最近の失業者の率は、日本の方が高い（韓国 3%、日本4%）。李会長はそのことを知っていて、日本の大企業の社長が自社のことしか考えていないことを皮肉ったのかもしれない。

ちなみに韓国の短期留学・研究まで含めた総留学者数は99年には20万220人であった。しかし2000年にはこれが25万4184人、2001年には27万7798人と増え、2002年には34万3842人になっている。

一般留学と小学生などの早期留学などをすべて含めた2002年の留学経費総額は45億8000万ドルと政府は発表した。これは108億ドルの貿易黒字の42%に相当する額になる。留学ばかりではない移民も増えている。2001年の韓国からの移民者数は13953人、であり90年代はほぼ同水準で推移している。

参考だが、日本人の海外派遣留学者総数は76000 人・平成12年度ユネスコ文化統計年鑑調べである。韓国の20万人は企業からの派遣まで含めた数であり正しい比較にはならないが、韓国人留学生は相当な数であることには違いない。この若者が学んで帰ったものは相当な知的資産の源にはなると思われる。

3.1.6.4 IMF と大改革

「IMFの時は大変でした。私もある韓国銀行から放り出されてこの会社に来ましたがあのときは、国を挙げての大幅改革が必要だった時期なので、仕方無かったです」とR専務が淡々と話された。誰にあっても「あのIMF危機からの脱出は国家挙げての大事業であった」と話されたことは印象的でした。

IMF危機脱出は以下のように行われたようである。

97年1月金泳三（キム・ヨンサム）政権は続出するストや株価下落に見舞われていた。

1月23日中堅財閥である韓宝グループ（財閥資産14位）の中核企業である韓宝鉄鋼工業が不渡りを出したのに引き続き資産規模26位の三美グループの主力2社も海外特殊鋼メーカーへの大型投資を機に資金繰りがつかず法廷管理を裁判所に申請破綻した。

4月には焼酎メーカーの真露グループ（韓国財閥19位）が負債比率3619%に達し破綻した。無理な拡大路線をとった財閥（漢拏は負債比率1986%、ニューコアは1224%）が金利負担に耐え切れず相次いで破綻し始めた。7月には韓国第二位の自動車メーカーで財閥8位の起亜自動車の経営危機が表面化した。11月には中堅財閥のヘテとニューコアが経ち続けに破綻。無理な多角化が原因の資金繰り悪化であった。12月には証券業界8位の高麗証券が破綻。続いて4位の東西証券もソウル地裁に法廷処理を申請して破綻した。韓国政府がIMFによる緊急支援で合意した2週間後の12月18日大統領選挙で金大中氏が当選した。年初1ドル843ウォンだった為替レートは24日には1965ウォンに達した。国際格付会社ムーディーズの評価は9月からわずか3カ月で6段階国家信用度を下げた。企業倒産件数は12月には

3200件と従来の3倍の数になってしまった。
98年2月に誕生した金大中氏政権は経済危機の克服を最優先課題に掲げ経済構造改革を3年間で完了させると宣言した。

その内容は

① 金融機関の健全性回復
② 不良企業の整理と企業支配構造・財務構造の改善
③ 公企業の経営革新と民営化推進
④ 法と原則に基づく労使問題の解決

であった。

金融監督委員会がまとめた不良債権は4月に時点で、ざっと10兆円であった。金融監督委員会は33あった銀行を「優良行」「「条件付存続可能行」「再生不能行」の三つに峻別し33あった銀行を最終的には18にまで絞り込み、さらに証券、保険、ノンバンクなどを含めて2068あった金融機関のうち実に631社が整理されることになった。銀行員は97年末の11万3000人から68000人に40％削減され、役員の数は501から183人に 1/3 にまで落とされた。
2002年までに投入された公的資金は15兆円に達した。これはGDPの30％に達する大規模なものであった。

1998年1月9日財閥改革5項目案が示された。

① 連結財務諸表の作成と経営情報の公開
② グループ内企業間の相互債務保証の撤廃
③ 財務体質の改善（負債比率を200%以下に引き下げる）
④ 主力事業の選定と事業集中
⑤ 経営陣の責任の明確化

であった。「金融と産業の一体再生」を目指し「企業の構造改革を、銀行を通して進める」金大中大統領の原則が打ち出され財閥はこれに従った。

99年8月にはこれに加え

① 系列金融機関の支配構造の改善
② グループ内企業同士の相互出資規制の強化
③ グループ内企業同士の不正取引の根絶

と言う3原則が示された。

この結果を受け製造業の負債比率は画期的に低下した。

97年末には396％あった負債比率は2002年6月末には136%にまで下がった。IMF危機からの脱出は大胆な構造改革によって成功したが終身雇用・年功序列システムは崩れ猛烈な社内競争を通じての人員合理化や経営幹部の若返りが進んだ。

・ 5％ルール　四半期ごとに社員の評価がつけられ、業績の悪い下位5％の社員は退職勧告され、6ヵ月の転職備期間を経て退職する。一方上位5％の優秀社員に対しては破格の待遇で年俸を上回るボーナスが支給されるほかストックオプションを与えられるケ

ースもある。海外留学や国内留学で費用は会社持ちで夜間大学に通う場合もある。

- 主要財閥企業の場合40代で役員、50代前半で副社長か社長と言う「若手幹部登用制度」が定着している。韓国の大手紙、中央日報が実施した百大企業の課長以上436人の79%が「終身雇用の考えをあきらめた」と答えている。
- こうして退社した社員はベンチャー企業設立に動き新興企業が数多く誕生した。この中にはITベンチャーも数多い。1999年の新規法人数/倒産法人数は12倍にまで膨れ上がった。韓国政府は新しい経済システムの成長源として浮上してきた「デジタル経済」を後押しし、」既存産業に代わる韓国経済の柱として期待している。

3.1.6.5 ＩＴに対する国家戦略

「わが社の1/3はパッケージの開発に勢力を注いでおります。その資金5000万円はすべて政府から出資されました」とA社長は得意げに話された。

「利益が出たら返しなさい」とまず全額出資してくれた内容はそれほどのレベルであるとは思えなかったが、国が世界を目指したパッケージの開発に力を注いでいるのは事実であろう。

1998年1月経済危機の最中に就任した金大中大統領が韓国の継続的発展には情報化が重要であることを強調し「知識基盤国家の建設」を提唱した。

この結果各家庭にインターネット高速回線が敷設され、ITになれ親しむようになったことは事実であろう。

その後もノムヒョウン大統領によってIT化ビジョンが進められている。日本のe―ジャパン構想は政府官庁や国民生活への寄与を目指したIT化推進であるが、IT産業を世界レベルに押し上げそれによって国家基盤を強化することにはなっていない。この力のかけ方の差はIT推進の差に今後どのように影響が出てくるのであろうか？

3.1.6.6 言語差の問題

韓国の町並みを歩いて気づくのは漢字の少なさである。漢字の使用禁止を出しそれを推進している韓国と漢字が相当に使える中国のどちらを日本企業が選択しシステム開発に関する注文を発注するようになるのかは、その他の要因にもよるので一概には言えないが「年配者には日本語をしゃべれる人がいる」のに反し「若者はほとんど日本語は語れない」ことは微妙に影響を出してくるに違いない。

「一番近い国韓国」と日本の IT 産業のかかわりは微妙である。この2国に加えて英語の国インドを交えてのシステム開発業務移管の推移は興味深い課題である。

ちなみに韓国の給与水準は上がって日本の 2～3割安い程度になってきている。仮にシステム開発におけるプログラム開発作業の割合が全作業の3割であったとすれば、その3割が安くなる可能性がMAXとなる。つまり9%程度安くなることに対して、中間SE（通訳）が必要、習慣、法律、文化の差などの影響を超えての日本へのビジネスは相当に厳しいものになる。

結局は安さで無く、良き機能、良き品質が勝負のキイポイントになってくるのではあるまい

か？韓国ソフトウェアが「良き品質である」と評価を取ってくる時期が早く来ることを願ってやまない。

「参考」

(A)なぜ日本は韓国にさきを越されたのか？　韓国IT研究会　日刊工業新聞社
　　2001年6月発行　1400円
(B)韓国はなぜ改革ができたのか？　玉置直司　日本経済新聞社　1500円
　　2003年4月発行
(C)韓国が日本伊追いつけない18の理由　百瀬格　文春文庫　514円
　　2001年4月発行
(D)韓国IT革命の勝利　河信基　宝島社新書　700円

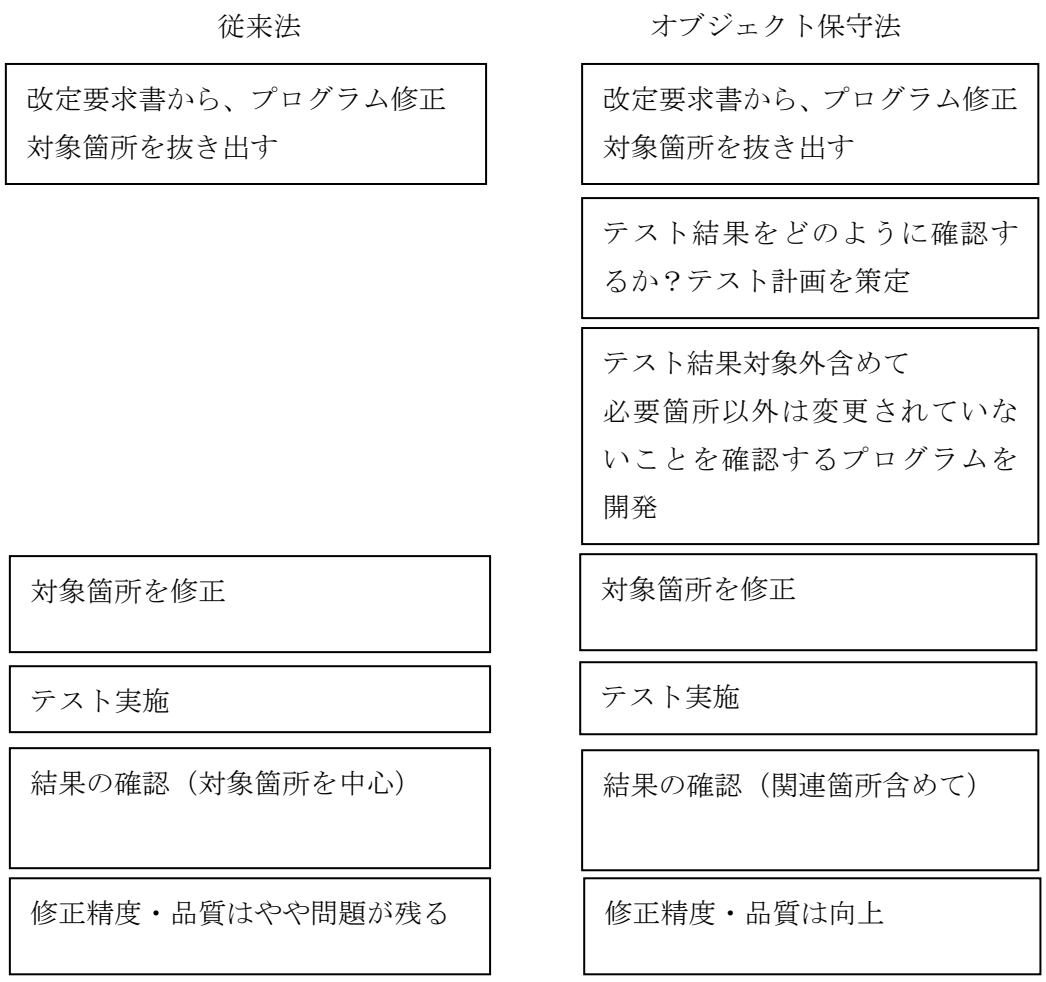
3.2　オブジェクト指向設計に関する保守問題

オブジェクト指向で開発したシステムは、保守性が高いと言われている。実際にオブジェクト指向技術に関する大規模システムを開発した企業の調査を行い、どの程度保守性が向上したかを報告する。

以下は、A社の大規模（200万LOC/c++）情報系システム保守問題について報告する。印象的であったのは以下の事項であった。

3.2.1 特徴と対策

① 継承機能を活用することで、部品を再利用し生産性をあげる事が出来る。しかし思わぬ箇所に影響を与える事があるので、対象箇所、必要箇所以外への影響は無いことを確認する方法を確立しておく必要がある。従来の保守方法でも、多かれ少なかれこの問題は存在しうるのではあるが、オブジェクト開発においては、特に component が数多いこと、影響範囲も大きいことから、特に注意が必要である。



② UMLを活用することにより、設計対象範囲、設計箇所、関連箇所などが明確になる。結果としてオブジェクト保守は精度が向上する。
③ Rosetta series などのTESTツールが活用でき生産性・品質が向上する。
④ 保守作業に必要なドキュメントを十分に残しておかねばならない。
⑤ 後継者の育成に十分に配慮すること

3.2.2 オブジェクト指向設計によるシステム開発に関する保守事例

オブジェクト指向設計システム開発について経験豊かなB社のNリーダからお聞きした感想を以下に述べる。

3.2.2.1 向上する事例

B社では、DOA（T字形ER手法）を分析・設計に利用し、実装段階でオブジェクト指向技術（以下、OO技術）を使っている。

OO技術は、カプセル化のために使うのではなく、部品化の為に使っている。部品化はMVCモデルのM，V，Cの全域が対象となっており、V，Cに関しては、既に240以上の部品（メインルーチン、画面出力、DB入出力を含む部品）が揃っている。

組立型のシステム開発ではコード量が大幅に削減されるため、保守性が向上する。
また、上記部品は、表示項目等がXMLのパラメータとして与えられているため、画面の表示項目の修正はJavaを修正する必要がない。この点は、保守性の点で大きな効果が得られる。

3.2.2.2 悪化させる事例

UMLを使ったシステム開発はコンサルタントが支援した一部のシステムは、成功しているが、多くの事例がQCDの悪化を招いている。当社でも3つほど失敗事例がある。

N氏の観察では、

① OO技術者はコードの重複を嫌い、コードを共通化するためにプログラムを分割する傾向がある。
② Java は関数型の言語である為、第 4 世代言語（パラメータ系言語）に比べてコード量が多いが、オブジェクト指向言語は部品化の機能が豊富であり、①の作業を進めると、多くの細かいプログラムがたくさんできる。
③ これらのプログラムは、複数のプログラムから呼ばれる（共通化するために分割しているので）ため、1ヶ所の修正が多くのプログラムに波及する。
④ 部品化が進み、プログラムの階層が深く（サブルーチンが次々とサブルーチンを呼んでいる状況）なれば、修正に対する影響が多くのプログラムに波及する。
⑤ この現象は、開発が進んだ段階と保守の段階で発生する。
⑥ 修正範囲は局所化できるが、影響範囲が広く、テスト作業の工数が膨らむ。しかも、テスト範囲の調査が難しい。

⑦　結局、開発、保守ともコストが増加する。

それどこか、本番化すると怖くて触れないという状況が発生する。

以上が私の体験した3つプロジェクトの共通点である。開発は異なる2社にお願いしたものである。

一方、うまく設計できたプロジェクトでは保守性が向上したという声も聞かれる。

ただ、開発生産性、保守性共に数字で効果を公表している会社は知らない。

3.2.2.3　　N氏の意見

開発生産性、保守生産性に関して、私は次のように思っている。

QCDの改善は、部品化が重要なファクタであって、オブジェクト指向技術は部品化を実現する為の技術である。

ここでいう部品は、開発プロジェクトとは別のチームで開発されたものでなければならない。

一方、世間では、変更箇所の局所化がメリットと言われているが、影響分析やテスト工数が議論されていないように感じる。

なお、XPのテストファーストはOOの欠点を自動テストで補ったモノと考えることもできる。

3.2.3 総括

以上保守問題を要約すると次のように集約される

① オブジェクト指向設計は基本的には保守生産性を向上させる仕組みを持っている。

② この良さを生かすには開発方法の標準化を確実に実施する必要がある。

- 保守作業を行うために必要なドキュメントを確実に残すこと
- コンポネントの構造化を行い、その深さを制限すること
- プログラムあるいはコンポネントを修正した場合その影響を確認する仕組みを十分に準備して行うこと

4 UML 例題作成

4.1　損保業務のモデル化

損害保険会社業務の UML モデリングを行う。当業務をモデリングの対象とする主旨は、

① オブジェクト指向を用いたモデリングが下流工程だけではなく、上流工程（ビジネスモデリング）でも利用できることを実証する。

② これまでベンダー主体のモデリング実績はあるがユーザの立場での試みは無かった為、ユーザが作成した場合にどのようなモデリングになるのかを検討してみる。

③ UML を利用して実際にモデル図を書いてみる。その結果が正解か否かは不明確であり、読者のご意見を求めたい。

である。

当モデリングは下記の順番で実施した。

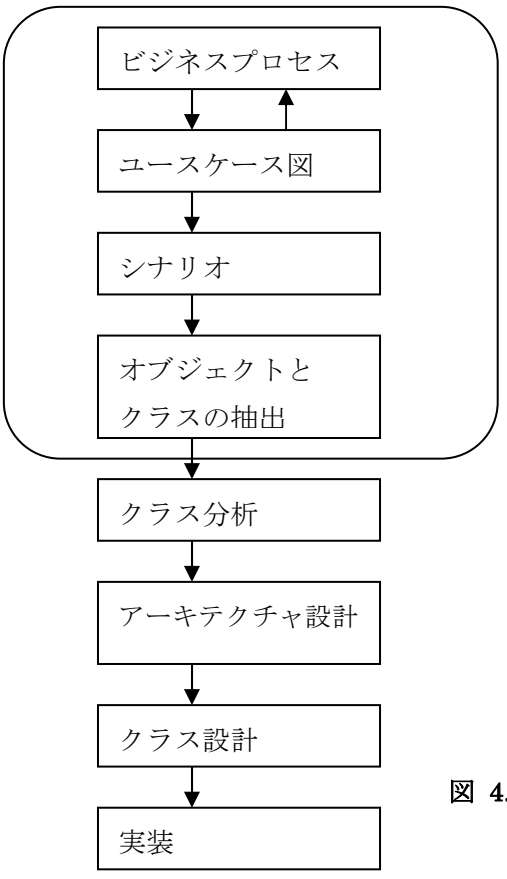
① 損害保険業界の全体ビジネス・ドメインを定義する

② 対象プロセスの詳細ドメインを定義する

③ 該当業務のユースケース図を作成する

④ ユースケース単位にシナリオを作成する

⑤ シナリオからクラス図・シーケンス図を作成する



←今回のモデリングの対象

図 4.1-1

4.1.1 損害保険業界の全体ビジネス・ドメインを定義する

図 4.1-2 はUMLで規定されているものではないが、このようなドメイン図を描くことによって、損保業界業務を知らないメンバでも全体像の概要・大まかな流れの把握が可能となり、認識を共有化でき、UML 図を作成する上で有効となる。

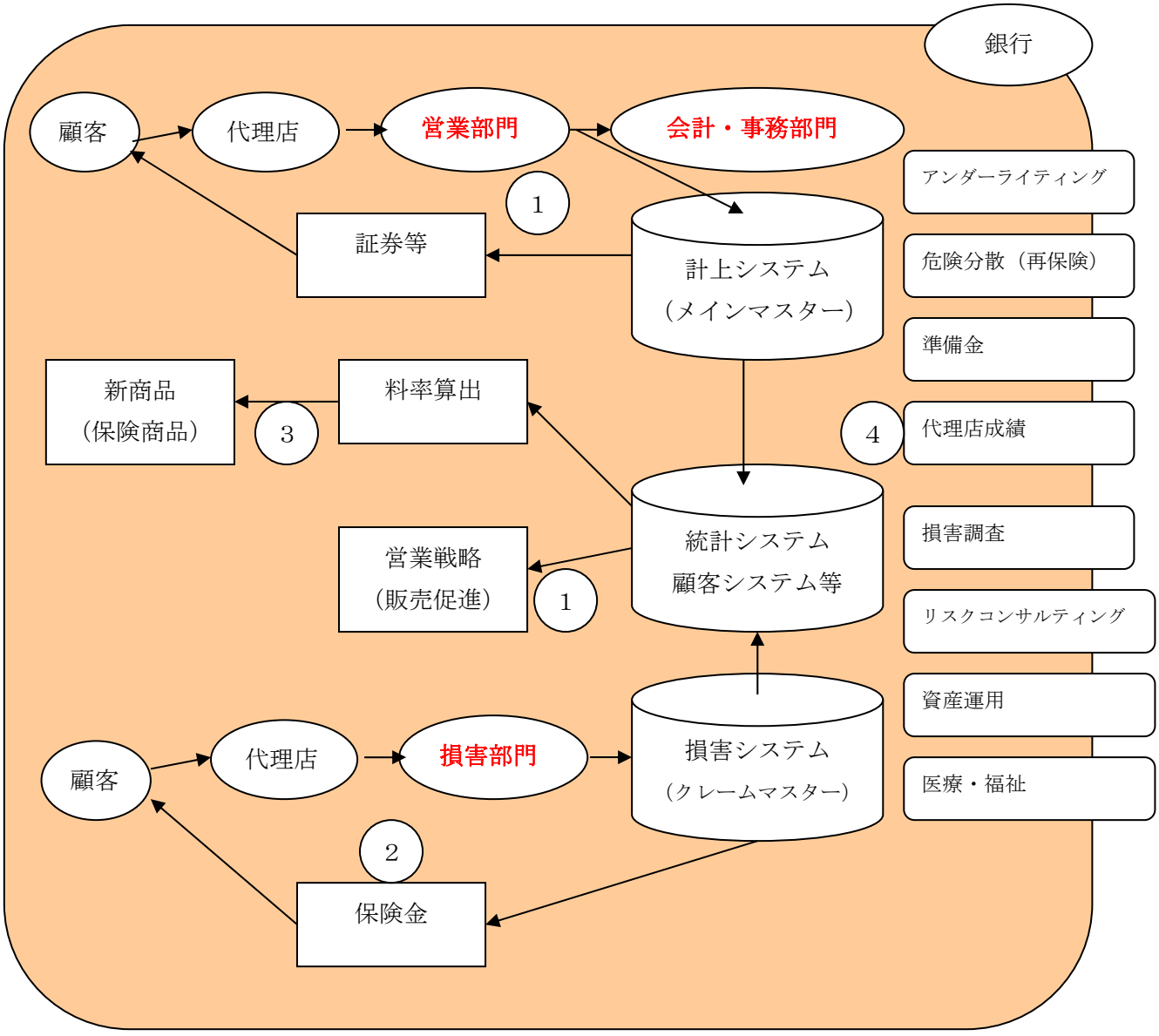


図 4.1-2　保険業界全体ドメイン

図 4.1-2 中丸数字について概要を記す。

①は販売プロセスを表した流れである。「顧客（お客様）が保険商品を申し込む」動作がトリガとなり、代理店を介して申込書・保険料が保険会社に渡る（営業部門、会計・事務部門は保険会社内の部門である）。計上システムで必要な処理がされ、証券等が顧

客に渡る流れである。過去の契約情報等を分析・戦略をたて販売を促進するという流れも販売プロセス内の大きなドメインの一つである。
②は事故が起きてしまったときのプロセスを表した流れとなる。販売プロセス同様基本的には代理店を介して保険会社(損害部門)にその情報が渡り、調査等の必要な処理を経て保険金が支払われる。
③は保険という商品の開発を表す概要プロセスとなる。損害保険には、販売時において、その原価が未確定であるという性質がある。保険料率は、過去の統計に基づいて、将来を予測することで算出する。
④は①～③以外を表す。アンダーライティングとは保険引受にあたっての各種判断(引受可否、担保条件、料率、引受金額などの判断)のことをいう。前述のとおり、保険は過去の統計、すなわち大数の法則の応用である。少数の巨大リスク(宇宙ロケット、ジャンボジェット、石油コンビナート、超高層ビルなどに対する保険)契約は、事故が生じると、巨額の保険金支払が予測されるので、一定額を手元に残し(保有)、それを超える責任を国内外の他保険会社に転嫁することにより、危険の分散をはかる。これを再保険という。また、事故を未然に防ぐ為のリスクコンサルティングや資産運用等も損保業界の重要な業務となる。

4.1.2 対象プロセスの詳細ドメインを定義する

今回は図 4.1-2 の内「①販売プロセス」についてもう少し詳細のドメインを定義することにする。販売プロセスの詳細ドメインは図 4.1-3 のようになる。図 4.1-3 中の数字(①～⑦)は業務の流れを表す。

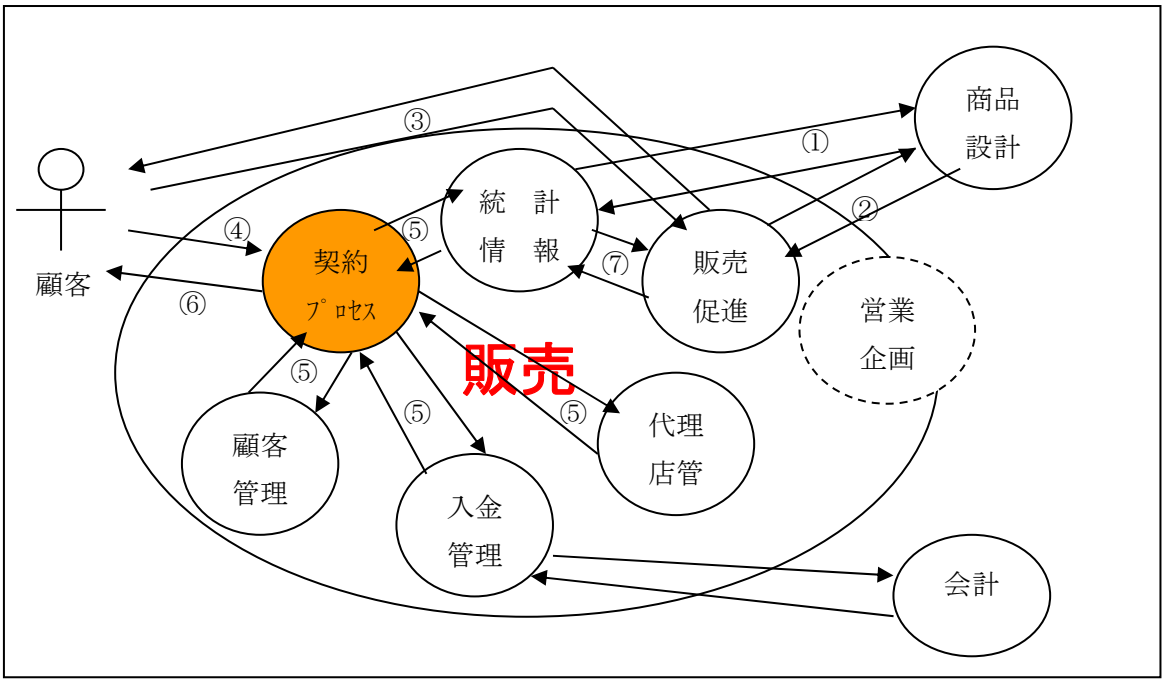


図 4.1-3　販売プロセス詳細ドメイン

過去の統計情報から保険商品設計がされ(①)、新商品についての販売促進方法を検討する(②)。テレビCMなどの宣伝を行い顧客(お客様)に告知する(③)。④は顧客(お客様)が保険を申し込む流れである。契約処理がされると、⑤の各管理がされ、契約者に対して証券が発行される(⑥)。保険販売後の動向によっては再度販売促進についての検討がされる(⑦)。

今回は更に「契約管理プロセス」について詳細化を検討した。「契約から保険契約が満期を向かえるまでの流れが図 4.1-4 となる。

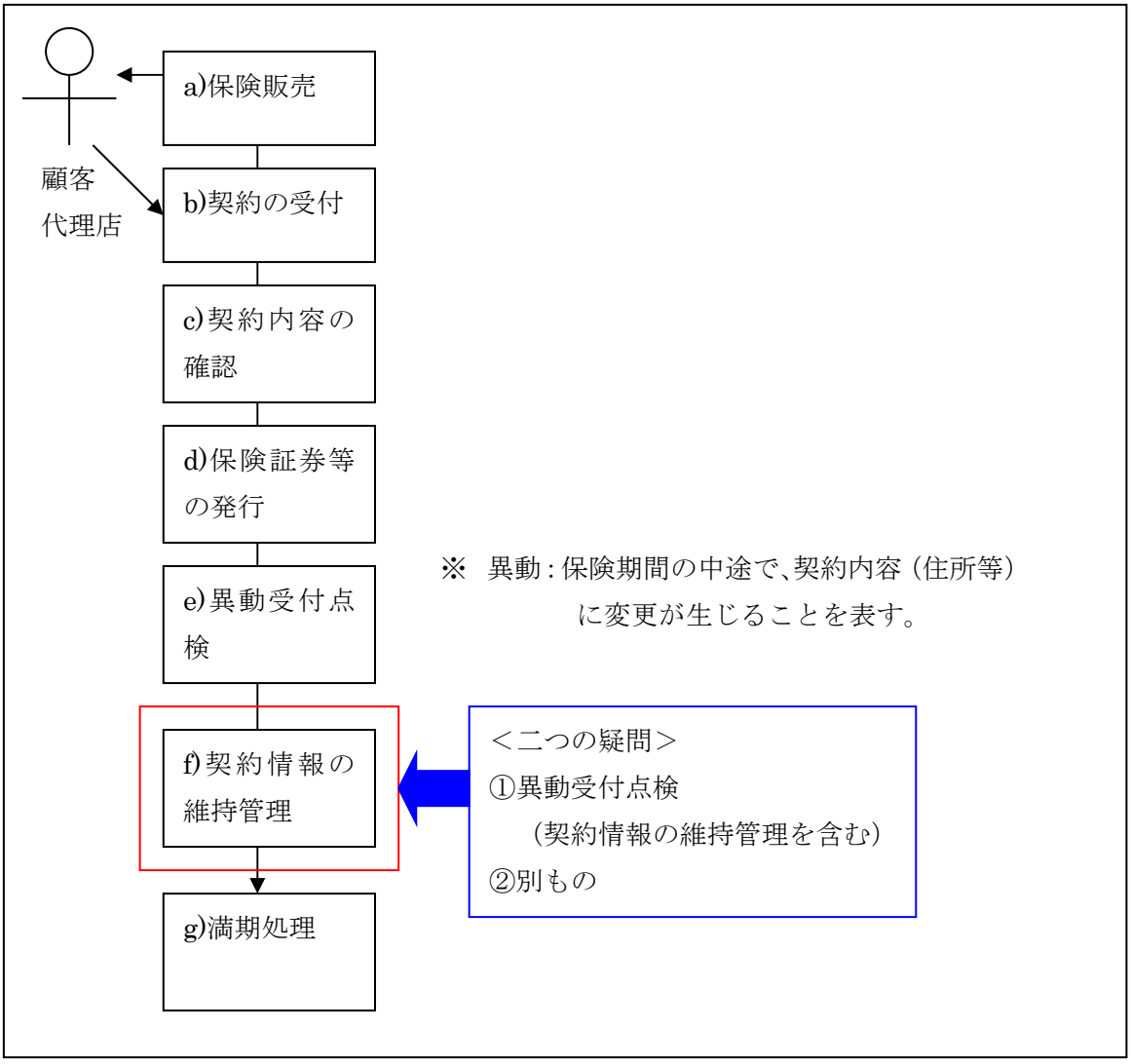


図 4.1-4 契約プロセスの流れ

図 4.1-4 の各 BOX は機能を表すが、当図を作成するにあたって様々な論議が交わされたの

で紹介する。
（メンバ A）
「f)契約情報の維持管理」は機能なのか？ビジネスプロセスとは、各プロセスで扱うデータがあるのに「f」契約情報の維持管理」にはそれがないので違和感がある。
（メンバ B）
今の問題点は、「f)契約情報の維持管理」を契約プロセスとして入れるべきなのかどうかの判断ではないだろうか。
（メンバ C）
「g)満期処理」を行うために、事前作業として契約の維持管理が必要なので、消す必要はないのでは。
（メンバ D）
粒度の捉え方の問題である。他の機能（BOX）と比べてレベルが違うと思う。
（メンバ E）
そもそもこの図を作成する目的は「プロセスの全体の流れを把握する」為である。漏れが無いようにするのだから、消さないでおいて良いのでは。
（メンバ F）
このような議論は日常茶飯事に起こること。我々はその指標・レベルを見極めたい為に行っている。

4.1.3 該当業務のユースケース図を作成する

図 4.1-4 の「契約申込み」から「保険証券の発行」までについてユースケース図で表す。

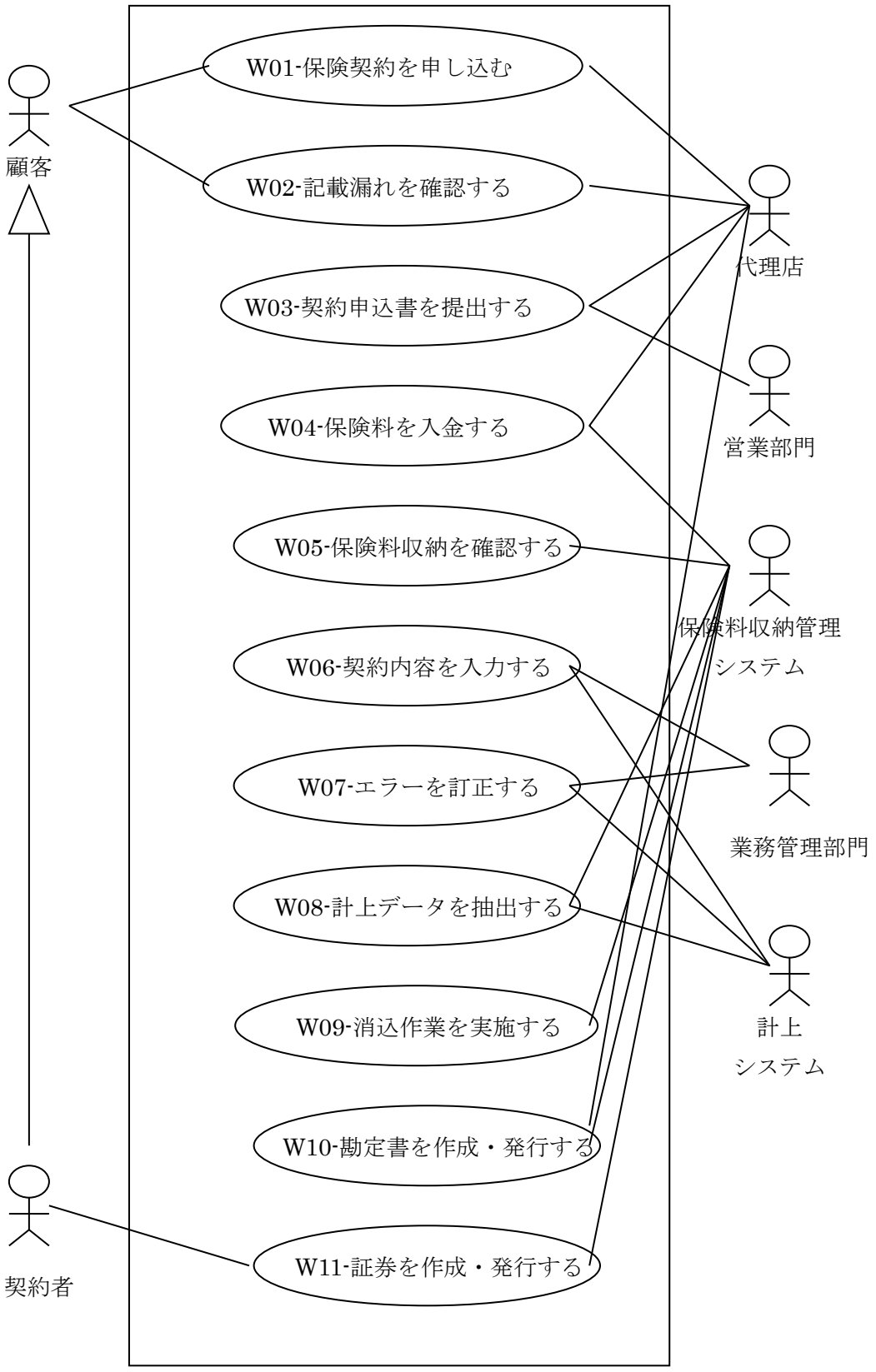


図 4.1-5　ユースケース図

次に UML「クラス図」を一部記述する。クラス図によって、静的なオブジェクト（クラス）構造とクラス間の関係を表すことができる。システム実装時クラス図には「クラス名」「属性」「操作（メソッド）」の詳細を記述する必要があるが、当段階（上流工程）では「操作」は省略する。

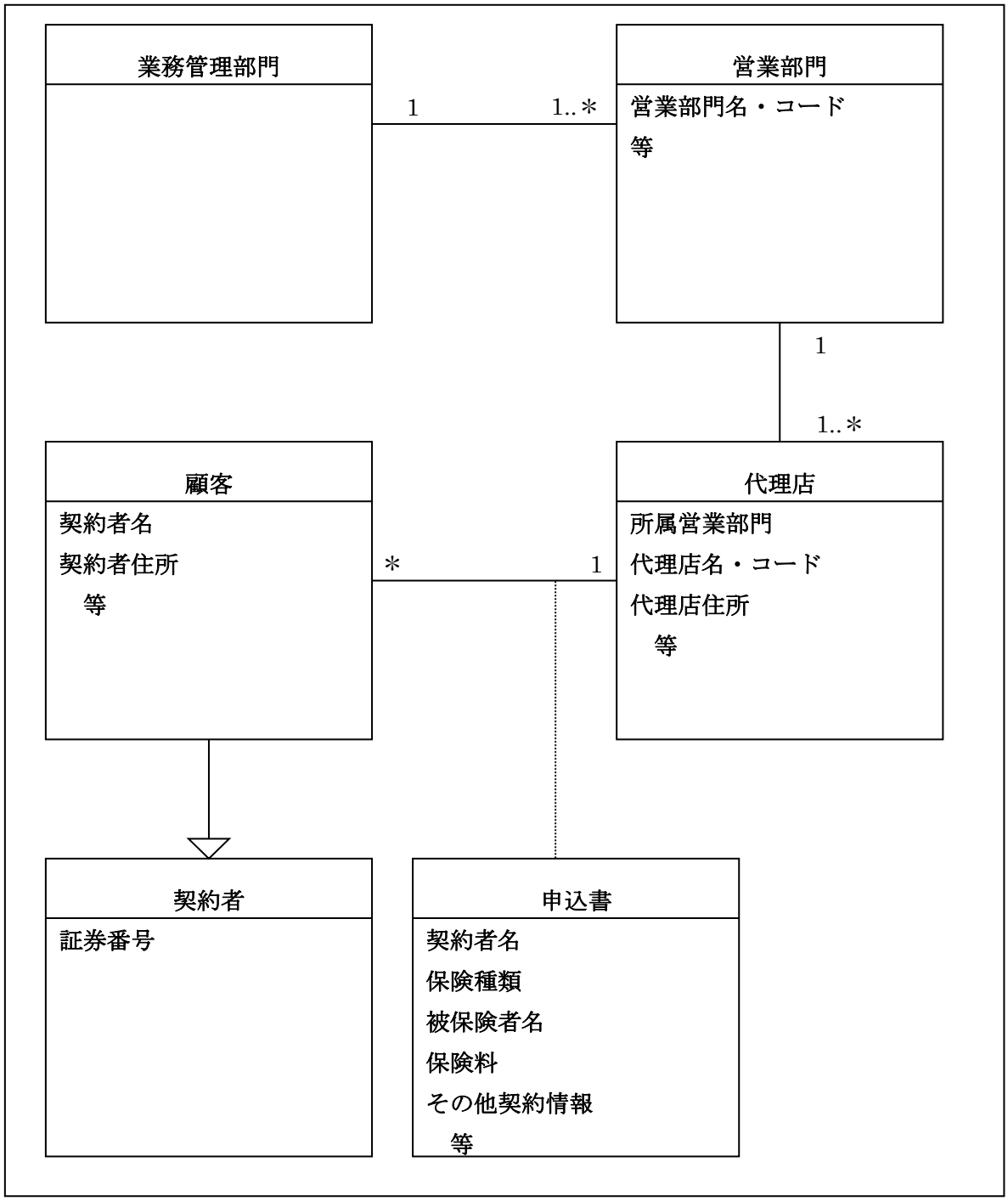


図 4.1-6　クラス図

オブジェクト間のメッセージのやりとりを時間順に表す振舞い図「シーケンス図」は下図のようになる。アクタとシステムがどのように相互作用するのかを時間を追って表すことができる。

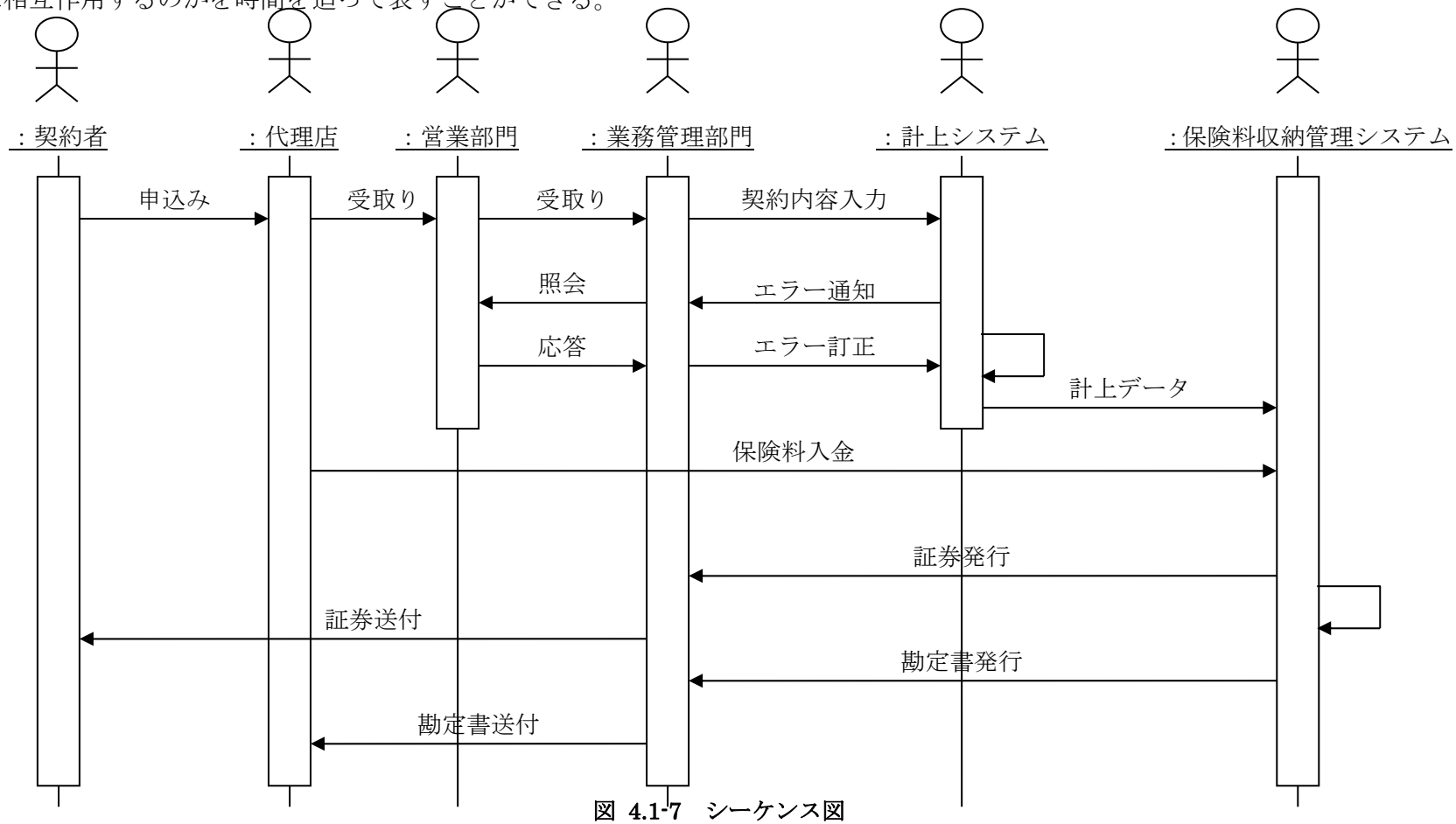


図 4.1-7　シーケンス図

4.1.4 総評

今回モデリングを実施した所感を述べる。

まず、UML で規定されている図を用いる前に、検討メンバ間の認識を共有化するために図1～3のような簡易図が有効であることが感じられた。その際に議論には参加しない立場のメンバ（書記）が一人いることにより、振り返りを行い更なる共有化が図れることもわかった。

この「全体ドメイン把握」の為の手段については標準化されたものがない。しかし、むしろこの時点では標準にこだわらず、メンバ間で利用できるものを好きに使うことにより柔軟な議論が進められるのではないか。

次に UML 図だが、振舞いを表す「ユースケース図」、構造を表す「クラス図」を利用することにより、業務内容を動的・静的ともに把握することができるようになった。つまり、システム開発において上流工程でもUML は有効な手段として利用できるのではないだろうか。OMG が認定している世界標準の図法であるという観点でも利用すべきと考える。

ただし、課題もある。今回は損保業界業務の中でも極一部の業務部分についてのモデリングをしかも簡易的におこなったが、実業務で利用できるレベルのモデリングをするとなるとかなりの時間を費やす必要があると思われる。この時間をユーザがいかに確保できるかが第一の問題である。

また実装レベルまで含めた下流工程での利用も視野に入れた場合、再利用可能なクラスを作成する為の粒度（目安）が不明確である。再利用ができて初めて「上流工程でのモデリング作成時間」を費やす価値があると考えられる。

これらの問題点は当プロジェクト発足以来の課題となっており、今回は改めてその課題を痛感したことになった。

5 最新技術の動向調査

5.1 フレームワークの技術動向

5.1.1 フレームワーク解説

5.1.1.1 フレームワーク（Framework）

フレームワークとは枠組みを意味する言葉である。コンピュータ業界においては、開発の手順やアプリケーション設計の方法、ドキュメント作成のためのルールやソフトウェア開発のための雛型を指していたり、特定業務向けもしくは特定業務に依存しない半完成品のアプリケーション製品のことを指していたりする。

このように、フレームワークという一言でも様々な意味を持っており、コンピュータ業界に従事する技術者でさえ、その言葉からイメージするものは各人各様である。

5.1.1.2 フレームワークの定義

一般的にフレームワークの意味するものを分類整理すると、広義なものと狭義なものに分けられる。

広義なものは、「開発フレームワーク」 または、「設計フレームワーク」と呼ばれ、開発の手順やアプリケーション設計の方法、ドキュメント作成ルールなどソフトウェア開発を推進するための枠組みを指している。（以降　開発フレームワークと称す。）

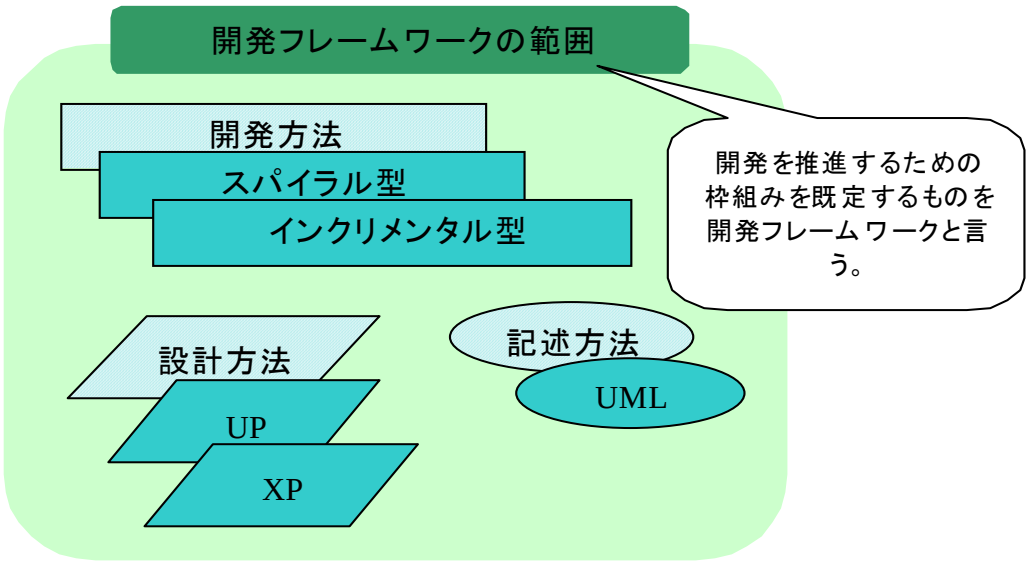


図 5.1-1　開発フレームワークの範囲

狭義なものは、「アプリケーションフレームワーク」と呼ばれ、特定業務向けもしくは特定業務に依存しない、ソフトウェアの基本的な枠組みを規定した半完成品のアプリケーション製品を指す。（以降 アプリケーションフレームワークと称す。）

本項では、このアプリケーションフレームワークを中心に報告する。

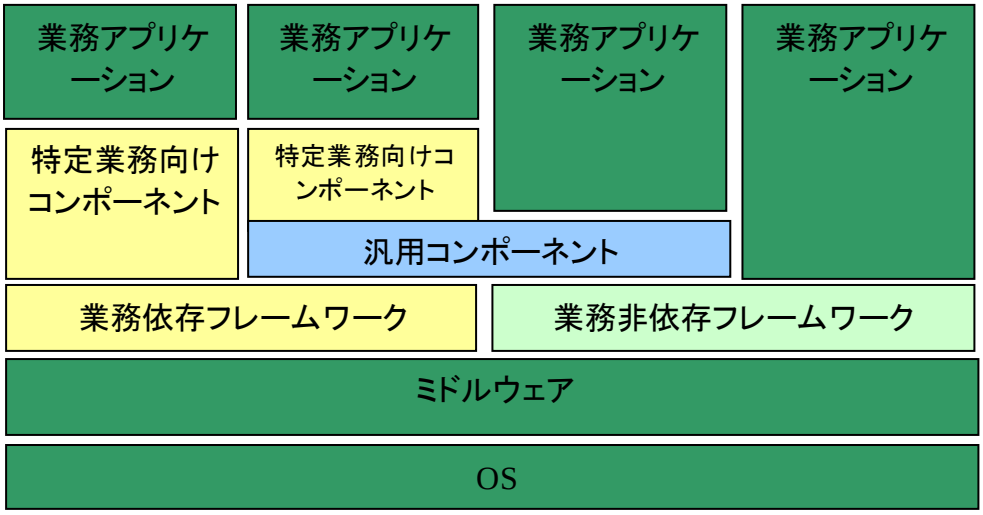


図 5.1-2　アプリケーションフレームワークの範囲

5.1.2 アプリケーションフレームワークの構造

5.1.2.1　　MVC モデル

現在、アプリケーションフレームワーク（業務依存型、業務非依存型含む）は、ソフトウェアの基本構造に MVC（Model View Controller）モデルを採用したものが主流である。

MVC モデルは、主に Web を前提とした技術で、ビジネスロジック（Model）、画面表示（View）、画面からの入力情報とビジネスロジックの処理結果をコントロールするコントローラ（Controller）の 3 層のアプリケーションにて構成される。従来のクライアントサーバ型モデルと比較し、プログラム毎の責務の明確化、メンテナンス性、拡張性の向上、コンポーネント化の促進による再利用性の向上などが期待できる。

5.1.2.2　　MVC モデルを用いたアプリケーションフレームワークの処理概要

MVC モデルを用いたアプリケーションフレームワークの処理は以下の通りである。

① ブラウザからの入力情報をリクエストとして受付。

② コントローラが、ブラウザからのリクエスト内容から、適当なビジネスロジックを呼出す。

③ 呼出されたビジネスロジックは、必要に応じて汎用コンポーネントやデータベース

との I/O を行い、レスポンスをコントローラに返す。

④ コントローラは、ビジネスロジックからのレスポンスに応じて表示する画面を選択。

⑤ Web アプリケーションサーバが、コントローラで選択された画面をブラウザに送信。

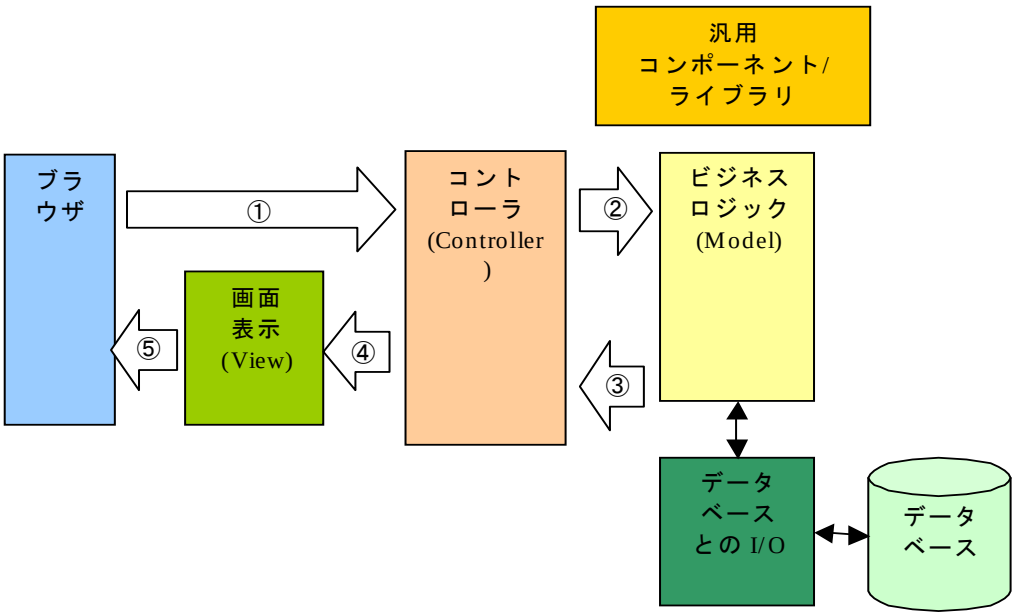


図 5.1-3　ＭＶＣモデルを適用したアプリケーションフレームワークの処理

コントローラは、アプリケーション全体の旗振り役である。①、③の入力情報と XML 等で予め記述された環境設定情報を参照し、これに該当する画面やビジネスロジックを選択し呼出すような仕組みとなっている。このような仕組みとすることで、特定の業務に依存しない汎用的なモジュールとすることができ、実際のフレームワーク製品でも提供されている。

画面表示、ビジネスロジック、データベースI/Oは、アプリケーション毎の独自性を排除することが難しい部分である。この部分をいかに汎用コンポーネント化してリリースできるかがアプリケーションフレームワークとしての重要なポイントとなる。

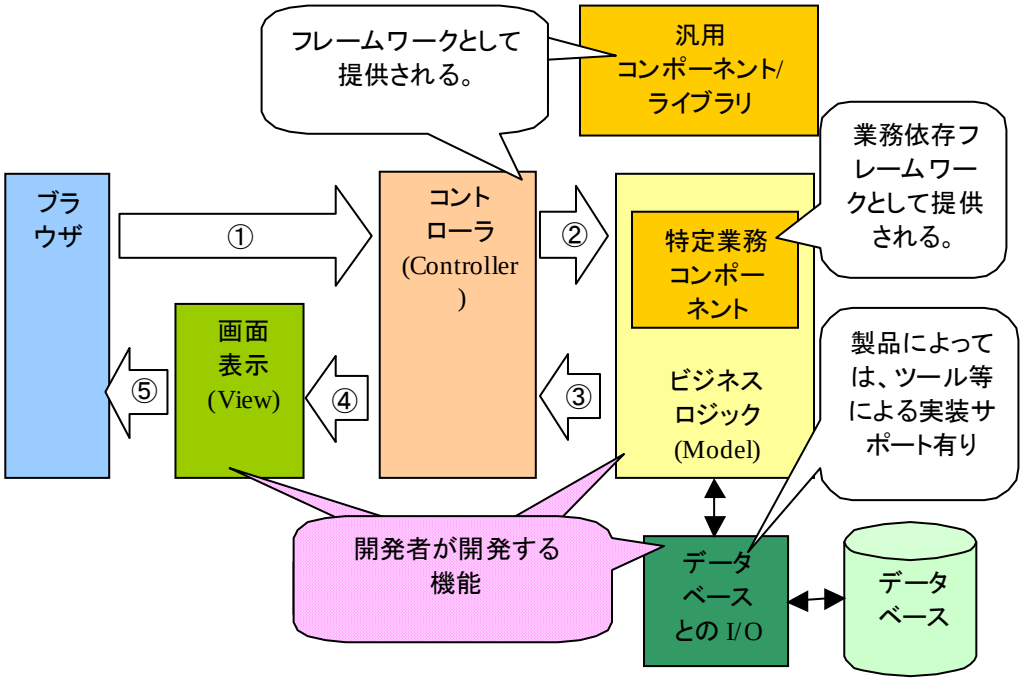
5.1.2.3　　アプリケーションフレームワークを用いた開発

① 要素技術

アプリケーションフレームワークは、半完成品として提供されており、未完成機能を開発者が補完し業務アプリケーションを開発していくこととなる。未完成機能の実装には、Java 及びその関連技術（Servlet、JSP、EJB、JavaBeans など）を用いるのが一般的であるが、アプリケーションフレームワークに.Net Framework を採用した場合は、Microsoft 系の技術で構成されることとなる。

② 開発範囲

アプリケーションフレームワークは、過渡期の製品であるため、提供される機能やツールは製品毎にかなりバラツキがあるが、汎用モジュールとして作りやすいコントローラや汎用コンポーネント,ライブラリは、ほとんどの製品で提供されている。開発者は、これらを利用し 画面、ビジネスロジック、データベース I/O の設計・開発を進めることとなる。(図 5.1-4 参照)。



*上図での示す提供機能は製品により提供範囲が異なる。

図 5.1-4 MVCモデルのアプリケーションフレームワークで提供される機能

③　アプリケーションフレームワーク導入時の課題

実際の開発においてアプリケーションフレームワークを導入する場合、導入教育にある程度の時間を必要とする。技術者の適応力により差異はあるが、筆者の経験では、Jakarta プロジェクトの Struts のようなアプリケーションフレームワークを導入する場合、経験年数 2~3 年程の Java プログラミング技術者で、製品の設計思想と実装までの手順を理解するのに 2 週間から 1 ヶ月程度の時間が必要であった。

また、技術者によっては、設計思想を理解出来ない場合があり、この場合、業務分担の工夫やプロジェクトメンバの入替え等の考慮が必要となる。

ビジネスロジック、画面、データベース I/O における設計では、開発するアプリケーションの出来を左右するものとなるため、優秀な技術者をアサインする必要があった。

設計の自由度（アプリケーションで実現できる機能の自由度とも言える）が高い製品ほど技術者の能力に依存し、設計の自由度が低い製品ほど技術者の能力に依存しないといえる。筆者の考えでは、Jakarta プロジェクトの Struts は、前者に該当すると考えている。

現時点では、万能な製品は存在しない。アプリケーションフレームワークの導入効果として何を目指すのか考え明確化した上で、製品選択をする必要があるだろう。

④　データベースとの接点

アプリケーションフレームワーク製品によっては、データベース I/O 部分の設計と実装をサポートする仕組み（ex…Jakarta プロジェクトの Torque）が提供されている。Jakarta プロジェクトの Torque を例にすると、テーブル定義情報を環境設定ファイルに設定するだけで、データベース I/O に関する実装コードを生成してくれる。このような仕組みを利用することで、データベース I/O に関する開発負荷の軽減できること、実装中に SQL の記述が無くなるため、プログラムコードの分かりにくさが排除でき、メンテナンス性の向上が期待できる等のメリットが享受できる。

実際の開発では、RDB 製品独自の機能を使用しなければならない場面や、レガシーシステムとの接続について考慮しなければならない場面がある。そのような場面では、データベース接続用のドライバ（ex…Java の場合 JDBC が該当）が RDB 製品に対応していなければ接続は難しいため、事前の考慮が必要となる。

⑤　業務依存型フレームワーク

業務依存フレームワークにおいては、ビジネスロジック内の 特定業務向けの処理をコンポーネントパッケージとして提供している。このパッケージを利用して、各企業の独自アプリケーションを構築していくこととなる。

製品の目指すビジネスモデルと企業がアプリケーション開発を考えているビジネスモデルとの適合度が高ければ、汎用フレームワークを適用する場合と比較し、効率の

良い開発が可能となる。

業務依存フレームワーク導入時の考慮点としては、

(1) 製品の目指すビジネスモデルと、企業がアプリケーション開発を考えているビジネスモデルとの間で十分な Fit＆Gap 調査が必要であること

(2) 当該製品が特定ベンダーの商品であるためベンダーの事情によりサポートの打ち切りや製品のバージョンアップ等が発生する可能性があること

(3) 設計が製品の思想に縛られるため、設計の自由度が無くなる傾向にある

の3点である。

(1)の調査の結果、ギャップが大きい場合は、

- 企業側のビジネスモデルを製品にあわせる
- 製品を大幅に改造して企業側のビジネスモデルにあわせる
- 当該製品を採用しない

等の対応が考えられるが、製品を大幅に改造する案は、製品のバージョンアップに追随出来なくなる、ベンダーへの保守費用の支払いなどネガティブな要素が多くお勧めしない。また、フィットする製品が存在した場合も、(3)で述べたように、設計の自由度が無くなることを理解した上で、製品選定をする必要があると考える。

5.1.2.4 市場に流通するアプリケーションフレームワーク製品

アプリケーションフレームワーク製品は、Jakarta プロジェクト(Web サーバ Apache を開発しているApache ソフトウェア財団で運営しているプロジェクト)で提供しているオープンソース系のフレームワーク(Struts)と、各ベンダーで開発されたプロダクトに分類できる。

現在のオープンソース系のプロダクトは、業務非依存フレームワークに分類でき、技術雑誌の付録やインターネットからフルスペックのソフトウェアを気軽にダウンロードすることができる。

また、利用方法や詳細な情報については、雑誌や解説書、講習会にて参加することで得ることができる。書籍については、一般の書店でも普通に目にすることができるため容易に入手することが可能であろう。

各ベンダーのプロダクトについては、業務依存型、業務非依存型、両者の特徴をミックスした型の様々なものが提供されている。

ホームページからのダウンロードやメディアの貸出しによって評価版のみ無償で入手可能である。利用方法や詳細情報については、各ベンダーの営業やベンダー主催のセミナーに参加することで得ることができる。

アプリケーションフレームワーク製品の種類と機能の比較情報について、コンポーネントスクウェア社のホームページ(http://www.c-sq.com/data/framework/table.html)に一覧表が掲載されている。

5.1.3 アプリケーションフレームワーク導入の現状と効果

アプリケーションフレームワークの活用により、ソフトウェア構造が規定されることは前述の通りである。これにより、プログラムの責務を明確にする設計が強要され、プログラムのコンポーネント化が促進される。プログラムのコンポーネント化により、優秀な技術者の設計と実装の遺産が後進の技術者へ継承され、その結果、

① 開発するアプリケーションの品質向上
② 効率的な開発の実現
③ 開発・維持の低コスト化

が効果として表れてくるものと考えられている。

実際に前述のような効果が得られているのかどうか、理想と現実のギャップがどれほどあるのか、現状の導入実績とその効果を分析してみた。

5.1.3.1 フレームワークの認知度と使用実績（導入数）

昨年度 ビジネスオブジェクト検討プロジェクトでは、JUAS 会員と訪問企業に対して「オブジェクト指向開発の貴社における定着度合い」についてアンケートを行った。このアンケート結果によると全 76 社中 90%の企業がソフトウェア開発に Java を使用し、39%の企業がフレームワークを持っているとの回答であった。このときのアンケートでは、フレームワークの定義を明確にしていなかったため、前述のどのフレームワークに属するのか、また、どの程度開発に適用しているか不明であるが、約 30 社がフレームワークを導入していることとなり、フレームワークの認知度は高いと言える。

使用実績については、インターネットを検索し調査したが、ベンダー提供のアプリケーションフレームワークにおいては、どの製品も、概ね数十程度のプロジェクト適用実績があった。また、Jakarta プロジェクトのアプリケーションフレームワーク Struts のについては、「@IT Java Solution 第 7 回読者調査結果 ～Jakarta Project の成果はここまで使われている！」の調査結果に Struts の利用状況が掲載されていたのでこちらを参照して欲しい。

（http://www.atmarkit.co.jp/fjava/survey/survey0212/survey0212.html）。

5.1.3.2 生産性向上、工期短縮、開発コスト削減

アプリケーションフレームワークを活用しソフトウェアを開発した場合、アプリケーションフレームワーク製品にて提供される汎用コンポーネントやツールの利用、過去に製作したコンポーネントの再利用によって、生産性が向上し、工期短縮、開発コスト削減というメリットを得ることができる。

昨年度のビジネスオブジェクト検討プロジェクトでヒアリングした B 社では、自社フレームワークを自社システムに適用した場合に、COBOL で開発した場合と比較し生産性が 2.6 倍、同 VB で開発した場合と比較し 1.8 倍に向上したという報告を頂いている。同社では、業務ロジックのコンポーネント化、画面部品のパターン化（同社は、複数の異なる業務にお

いても操作性を統一している)、開発プロセスの標準化、開発ツールの自社開発とこれらの再利用により、技術者のテクニックに依存する部分を最小限にし開発効率を向上させた結果と述べていた。

これは非常に高い成果だと考える。逆に、同社の取り組みのように出来ない場合、例えば、アプリケーションフレームワークを最初に適用するプロジェクトの場合やコンポーネント化した業務ロジックを再利用できない場合、パターン化した画面部品が再利用できない場合については、前述のような生産性は得られないと考えることができる。

5.1.3.3 コンポーネントの再利用と保守性

アプリケーションフレームワークは、プログラムのコンポーネント化に適した基盤である。保守性については、このコンポーネントの設計が重要な鍵となる。

昨年度のビジネスオブジェクト検討プロジェクトでヒアリングしたA社では、同系列の業務システム間で共用できるコンポーネントを製作し、同系列の複数の業務システムで再利用することで、保守性の向上につなげている。

同社では、データチェックなどの基本部品とデータ項目部品が 50%、画面表示と画面制御部品が 75%、メインフレーム間通信や障害監視部品で 95%の再利用を果たしている。これに伴い、プログラムメンテナンス時の修正範囲や修正ステップ数が減少し、修正時間が減少傾向にあるとの報告を頂いた。

一方、再利用化が促進すると、プログラム修正をした場合の影響範囲が広がるために修正後の検証負荷が増大するとの報告を別の企業から頂いている。これについては、現時点では詳しく調査しておらず、どのような設計で、どのくらい検証負荷が増大するのか追加調査が必要と考える。

5.1.3.4 技術者のスキル補完と品質

アプリケーションフレームワークのソフトウェア構造の強要とコンポーネント化の促進は、技術者のスキル補完と品質向上にも貢献している。

ソフトウェア開発という分野は、工学的手法が確立されていない分野である。ソフトウェア開発技術者は、10人いれば10人とも違う作法でソフトウェアを開発し、各々の技術力や理解力にも格差がある。このような格差を最小限にできれば、開発するソフトウェアの品質向上につながるはずである。

先程も例にあげたB社では、優秀な技術者を集めチームを形成し、このチームがフレームワーク及びコンポーネントを開発し開発チームに提供している。前述の通り、業務ロジックに至るまで、コンポーネント化が促進されているため、一般の技術者のコーディング範囲は、従来のものと比較し少なくなっているとのことである。また、完成後のバグもかなり少ないと述べていた。

5.1.3.5　　レガシーシステムからの移行と共存

レガシーシステムからWeb系システムへの移行では、

① 業務プロセス及びシステムを全面刷新する場合

② 業務プロセスとシステムを部分的に刷新する場合

③ 業務プロセスはそのままでシステムを全面移行する場合

④ 業務プロセスはそのままでシステムは部分的に移行する場合

という4つのケースが考えられる。

アプリケーションフレームワークを適用において、特に②、④のケースは、レガシーシステムとの情報授受を意識しなければならない。アプリケーションフレームワーク製品には、レガシーシステムとの連携のための専用コンポーネントは用意されていないため、MQによるメッセージ連携やJDBCを利用したデータベースとの連携、ファイル入出力による連携を駆使し開発をする必要がある。

③のケースでは、現在、IBM iSeries400 や Windows アプリケーションを J2EE 環境へ移行するための支援ツールが出てきており、移行の選択肢が増えてきている状況である。

5.1.3.6　　フレームワークに関する動向

近年、最も注目されているフレームワークが、Jakartaプロジェクトの Struts である。Strutsは、オープンソースと呼ばれる、ソースコードを無償で公開するライセンス形態を取っており、不特定多数の技術者が自由にソースコードの修正と拡張を行える環境が整っている。このような中で、現在 自力のあるIT企業が、Strutsの機能拡張や自社製品へStrutsサポート機能を追加するといった取り組みを行っている。

日本IBM社では、「Extension for Struts」というStrutsの機能拡張版をオープンソースとして提供。野村総合研究所社では、「T-Struts」と称し Struts の機能拡張を積極的に行いホームページ上（http://t-struts.sourceforge.jp/）で公開。また、Borland 社では、自社のJava 統合開発環境の J Builder に Struts による開発をサポートする機能を追加している。このように、自力のある企業がオープンソースの利点を生かして、Struts の機能拡張や自社製品への機能追加に注力してきており、更なる熟成と今後の普及拡大が期待される状況となっている。

5.2　デザインパターンの技術動向

5.2.1 デザインパターン概説

5.2.1.1　　デザインパターン（Design Pattern）

プログラマがソフトウェア仕様書に基づき、最初からコーディングしていく場合、プログラマ個人の性格、経験、能力により千差万別の構造をもったプログラムが出来上がってしまう。このため、良いプログラムを作ろうと思うと必然的にスーパーマンのようなプログラマが何人も必要となる。このような事は、実際には非現実的である。

その為、経験豊富でスーパーマンのような能力を持ったプログラマの経験や手法を概念的にまとめたものをデザインパターンとして広く使えるようにしようと考えられた。

代表的なデザインパターンとしては EJB(Enterprise Java Beans)があり、さまざまな機能が作られている。

デザインパターンを使うことにより、以下のような長所がある。

① 過去の豊富な経験をデザインパターンという形で蓄積し、それを後に続く者が活用することができる。

② パターンを使うことにより、プログラムのアーキテクチャが決められ、プログラムの品質や構造が一定のレベルにあることを期待できる。

③ パターンを使うことにより、機能ブロックの名称やコンポーネントの名称が共通化され、エンジニアのコミュニケーションが用意になり、携わる多くのプログラマ間での認識の調整が容易になる。

パターンの考えでは、プログラミングのスタイルと問題解決法などの情報や手法を標準化することになる。プログラムにおけるパターン化は効果的、効率的な手法であり、プログラムの品質の向上につながるものである。

5.2.1.2　　デザインパターンの種類

デザインパターンとしては、大きく次のように分類される。

① 生成に関するパターン

② 振る舞いに関するパターン

③ 構造に関するパターン

④ システムパターン

5.2.1.2.1　生成に関するパターン

オブジェクト指向プログラミングにおける最も基本的なパターンである。
システムにおけるオブジェクトの生成をつかさどる。

以下の特徴を持つ。

① 汎用的なインスタンスの生成
コードの特定クラスの種類を識別することなく、システム上でオブジェクトの生成を可能にする。

② 簡易性
一部のパターンはオブジェクトの生成処理を簡易化することから、呼び出し側ではオブジェクトのインスタンスを生成するために複雑なコードを書く必要がなくなる。

③ 生成の制約
一部のパターンはシステム上で生成できるオブジェクトの種類や数を強制的に制約する。

代表的なパターンとして以下のものがある。

① Abstract Factory
② Builder
③ Factory Method
④ Prototype
⑤ Singleton

生成に関するパターンのうち、Abstract Factory パターンについて説明する。

Abstract Factory パターンは、異なるリソースや OS を使用しなくてはいけないとき(ウィンドウ機能、ファイルシステム、他システムとの通信など)に有効な方法である。

Abstract Factory パターンを使うと異なるリソースや他システムとの通信などにおいて、新しい要求が発生するつど、再コーディングを行わずに、リソースを使用できるような柔軟なアプリケーションを作成できる。

Abstract Factory は具象クラスを指定せずに、関連オブジェクトまたは従属オブジェクトの集合を生成するための規約を提供するものであり、次の場合に使用する。

- クライアントが製品を生成する方法に依存しない場合
- アプリケーションが複数の製品の集合で構成される場合
- 相互に互換性を保つためにオブジェクトをセットとして生成しなければならない場合
- クラスの集合を提供する場合に、クラスの実装ではなく規約と関係だけを公開したい場合

Abstract Factory パターンによりアプリケーション全体の柔軟性が向上する。ただし、Abstract Factory パターンでは汎用インタフェースを適切に定義しないと目的の Concrete Product の生成が難しくなったり生成できなくなったりする。

5.2.1.2.2 振る舞いに関するパターン

振る舞いに関するパターンはシステム全体のフロー制御に関連している。

何らかの出方法で1つのシステムにおける制御をまとめると、システムと効率と保守性が向上する。振る舞いに関するパターンは実証された慣例的手法の本質を引き出し、わかりやすい経験則に変える。

代表的なパターンとして以下のものがある。

- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Visitor
- Template Method

振る舞いに関するパターンのうち Chain of Responsibility について説明する。

Chain of Responsibility パターンはシステム内でチェーンを確立し、メッセージを最初に受けとったレベルで処理するか、処理可能なオブジェクトに直接送ることを可能にすることを目的としている。

Chain of Responsibility パターンは、システムにおいて動作が発生した場合に、イベントまたはメッセージとして表す場合に使うもので、単純な例では、オブジェクトがイベントに応じて処理を行い、イベントに応じたテキストを表示する場合などに使用する。

5.2.1.2.3 構造に関するパターン

構造に関するパターンはアプリケーションの要素を分解し、組み合わせるための効果的な方法を表す。構造に関するパターンがアプリケーションに与える影響は様々あり、たとえば~~である~~、

代表的な構造に関するパターンとして以下のものがある。

- Adapter
- Bridge
- Composite
- Decorator
- Façade

・ Flyweight
・ HOPP (Half Object Plus Protocol)
・ Proxy

構造に関するパターンのうち Adapter について説明する。

Adapter パターンは、2つのインタフェース間の仲介を行うもので、一方のクラスのインタフェースを変換してもう一方のクラスで使用可能とするものである。

Adapter パターンは次のような場合に使用する。

・ オブジェクトを、そのオブジェクトのインタフェースとは異なるインタフェースを使う環境で利用したい場合
・ 複数のソース間でインタフェース変換を行う必要がある場合
・ 1つのオブジェクトをクラスグループの1つの仲介者として機能させたいが、実行時までどのクラスが利用するのかわからない場合

あるクラスを、新しい環境に一致するように再コーディングすることなく新しいフレームワークで使用したい場合に、Adapter クラスを設計して変換装置として機能させる。Adapter パターンを使うことによって本来互換性の無い複数のオブジェクトが相互にやり取りできるようになる。

5.2.1.2.4 システムパターン

システムパターンはアプリケーションを最も抽象的なアーキテクチャのレベルで囲むものであり、アプリケーション内の主な処理手順だけでなく、複数のアプリケーション側にも適用する。

代表的なシステムパターンとして以下のものがある。

・ MVC (Model View Controller)
・ Session
・ Worker Thread
・ Callback
・ Successive Update
・ Router
・ Transaction

システムパターンのうち、MVC パターンについて説明する。

コンポーネントまたはサブシステムをモデル（Model）、ビュー（View）、コントローラ（Controller）という3つの論理的な部品に分割し、それぞれの変更やカスタマイズを容易にするものである。この MVC パターンは柔軟で保守しやすいコンポーネントを作成したいときに役に立つ。

MVC パターンは柔軟性が高く、さまざまな状況に対応できる要素を作成するための優れた方法である。

MVC パターンの最大の課題は正確な基本表現を定義することであり、実装する場合にはアプリケーションに固有の制限を課すことのないようにしなくてはいけない。

MVC パターンは次の特性の一部が当てはまる場合に有効である。

- コンポーネントおまたはサブシステムを複数の異なる方法で表示できる。システムの内部表現は、画面上の表現とは異なる場合
- 異なる種類の振る舞いが予測される場合。複数の起動元から同じコンポーネントに対して呼び出すが、振る舞いが異なる場合
- コンポーネント使用中に振る舞いまたは表現が変わる場合
- コンポーネントを再利用することが、機能が必要となる場合

モデル・・・要素の状態であり、状態を変更する手段

ビュー・・・要素の視覚的または非視覚的な表現

コントローラ・・・要素の制御機能であり、ビューに対する操作をモデルに対する変化に関連付ける

5.2.1.3 Java プログラミング言語におけるパターン

5.2.1.3.1 イベント処理

イベント処理はシステム内の状態変化について2つ以上のオブジェクト間での通信を可能とする仕組みである。イベントベースのシステムではあるオブジェクトがイベントの発生元として動作し、システムの状態の変化を表すイベントオブジェクトを生成する。その後、1つ以上の受信者に何らかのメソッドを呼び出すことによりイベントを渡す。

イベント処理で使われるパターンとしては以下のものがある。

- Observer パターン
- Adapter パターン
- Factory Method パターン
- Composite パターン
- Chain of Responsibility パターン
- Command パターン

5.2.1.3.2 JavaBeans

JavaBeans は Java プログラミング言語に標準化されたモデルを提供し、コンポーネントとしてクラスを開発できるようにするものである。各コンポーネントはデータや振る舞いを表す方法が標準化されているため、開発者間で簡単に共有できる。Java Beans のようなコンポーネントモデルは、ある開発者のコードを別に国の企業で働いている開発者でも再利用できるようになる。

JavaBeans でのパターンとしては以下のものがある。

- Factory Method パターン
- Singleton パターン
- Adapter パターン
- Observer パターン

5.2.1.3.3 GUI

GUI では、コンポーネント、コンテナ、レイアウトマネージャの概念が中心になる。コンポーネントは、ボタン、ラベル、テキストフィールドなどの GUI 要素を現す。コンテナは Java のグラフィックスを組織化する役割を持つ。レイアウトマネージャはグラフィカルではなく、コンテナ内のコンポーネントのサイズや一を決定する作業に特化したオブジェクトである。

5.2.1.3.4 コレクション

コレクションのフレームワークにより、自分でコードを書くことなしにコレクションの動的なサイズ変更などの機能を利用できる。コレクションのパターンは Java に最初に導入されて以来、継続して変更が加えられ、利用されている。

5.2.1.3.5 入出力

Java における入出力 API の主な目的はストーリムをかのにするためである。ファイルへ直接もしくはネットワークを介しての書き込み、読み出しに使われる。

5.2.1.3.6 リフレクション

リフレクション API により、実行時にクラスとオブジェクトの情報を調べることができる。この機能は内部検査処理と呼ばれ、プログラムの実行中に新しいクラスを動的に挿入したいときに使用する。リフレクションを使うとクラスをその名前だけで動的にロードできる。

5.2.1.4 分散技術

5.2.1.4.1 JNDI

JNDI(Java Naming and Directory Interface)は、Java コードからルックアップサービスへアクセスするために標準化された方法を提供する目的で開発された。

JNDI により、ネームサービスとディレクトリサービスのすべての範囲でアクセスを標準化できる。

5.2.1.4.2 JDBC

データベースにアクセスする場合のインタフェースである。JDBC は SQL データベースにアクセスするための汎用フレームワークであり、データベースと接続するためのさまざまなモジュール上に統一したインタフェースを提供する。JDBC により、特定の DBMS に依存しない方法でデータベースのデータを操作できる。

5.2.1.4.3 RMI

RMI は、リモートメソッドの呼び出しによる通信を行う。RMI を使うことにより、アプリケーションは基本的に同じアドレス空間に存在しないオブジェクトに対してメソッドを実行できる。Java クライアントはオブジェクトに対して単純なメソッドの呼び出しを実行するだけで利用できる。

5.2.1.4.4 CORBA

CORBA（ Common Object Request Broker Architecture ）は分散オブジェクトの通信アーキテクチャである。アプリケーションがリモートメソッドを呼び出すことにより他のアプリケーションにサービスを要求する方式である。

5.2.1.5 J2EE

J2EE は分散アーキテクチャの開発と配備をサポートするために定義されたものである。J2EE では、コンポーネント、コンテナ、コネクタという3つの概念が中心になっている。J2EE では、協調して動作する Java コンポーネントのセットとしてエンタープライズアーキテクチャを構成する。以下の特徴を持つ。

- 柔軟性、拡張性
- 標準化が容易
- アプリケーションが実現するサービスセットの開発が容易

J2EE では、次のコンポーネントが中心となっている。

- EJB

  J2EE で定義されたエンタープライズ向け機能を持つ Beans 管理のための仕組みであり、ビジネスロジックに相当する機能を実現する。J2EE アーキテクチャの中心を構成する。EJB は、次の3種類に分類される。

  セションビーン
  エンティティビーン
  メッセージドリブンビーン

- JSP、サーブレット

  HTML ページを作成する環境を提供するもので、JSP やサーブレットから EJB を呼び出すという形態となる。

5.2.1.6　　その他

参考図書:デザインパターンによる Java 実践プログラミング(アスキー)

5.3 ＭＤＡ（Model Driven Architecture）

5.3.1 MDA とは

MDAとは、UML, CORBA 等の仕様を定めているオブジェクト指向関係の標準化団体OMGが、2001 年 3 月に正式に提唱した理念であり、今後、OMGの活動の中心になっていくものと考えられる。

その基本的な発想は、UML等の実装プラットフォームに依存しないモデルを、できる限り実装に依存しないモデルのまま詳細化していき、最終的に実装に繋げるための全体的なアーキテクチュアを定めようというものである。

MDAは、ビジネス・ロジックを実装プラット・フォームから独立させることにより、20年間持続するソフトウェア・アーキテクチュアを実現し、その結果として、既存システムと新規システムの速やかな統合、システム保守コストの削減を実現するとされている。

その背景としては、近年、CORBA, XML/SOAP, J2EE, .NET 等の様々な実装プラットフォームが生まれ、変遷していく中で、ビジネス・ロジック自体に変化がなくとも、業務システムをこれらの実装環境の変遷に追随させるために、多くのコスト・時間を要しているという現実があげられる。

5.3.2 全体概要

MDAは、図 5.3-1 に示すように、2種類のモデルの存在を前提にしている。

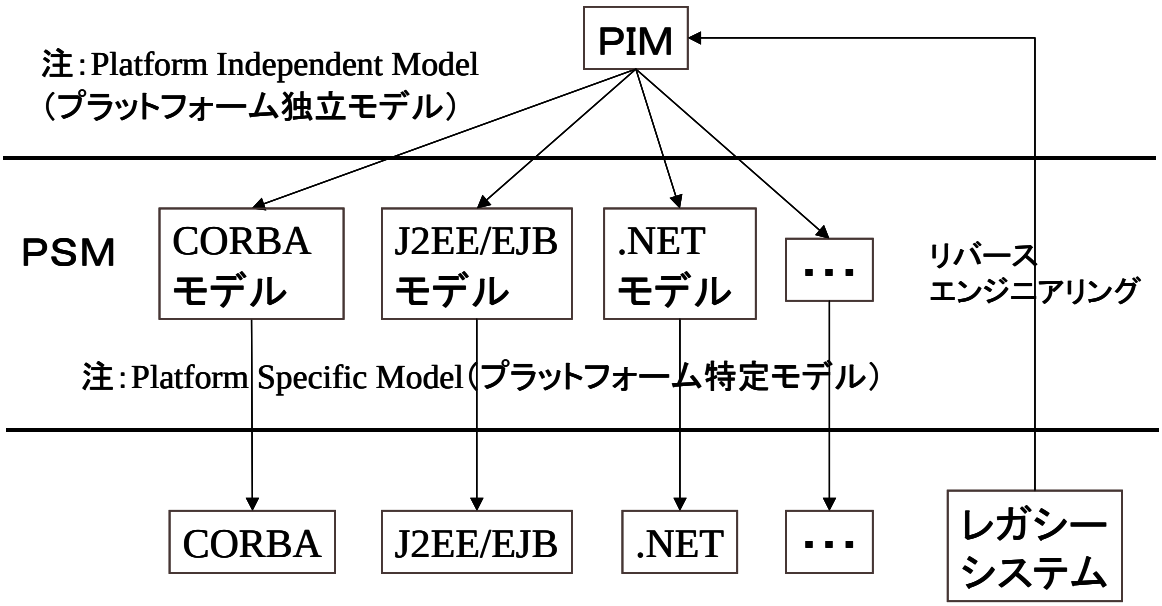


図 5.3-1　ＭＤＡの概要

1つは、ビジネス・ロジックの実装プラットフォームから独立したモデル化を実現させるためのPIM（Platform Independent Model，プラットフォーム独立モデル）であり、もう1つが、特定のプラットフォームに依存したPSM（Platform Specific Model，プラットフォーム特定モデル）である。

PIM で表されたビジネス・ロジックは、あらかじめ用意された変換ルールにより、各プラットフォームの PSM に変換される。そこから、各プラットフォーム上の実装システムが生成される。また、レガシーシステムは、リバースエンジニアリングにより PIM モデルを作成する。そこから、最新の実装プラットフォームの PSM に変換し、さらに実装システムを生成する。容易に最新のプラットフォームへの変換が可能とされる。さらに、システムの開発・維持において、プラットフォームの変更による影響が局所化および極小化されることになる。

また、メタレベルでMDAの全体概要を示すと、図 5.3-2 のようになる。図 5.3-2 からも分かるように、MDAは、PIM から各 PSM への変換だけでなく、異なる PIM 間、異なる PSM 間の変換も意識されている。

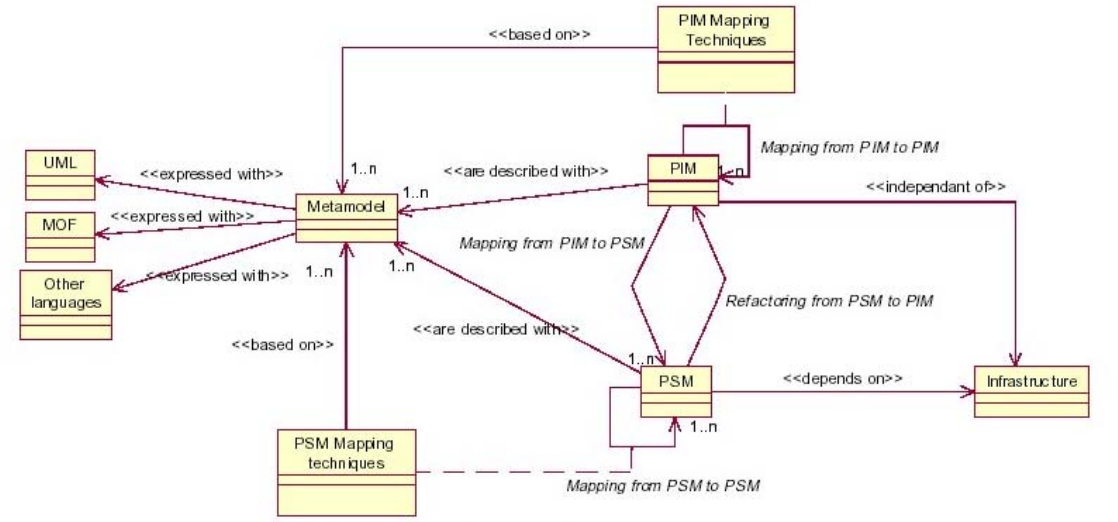


Figure 8. *MDA Metamodel Description*　　出典：OMG ormsc/2001-07-01

図 5.3-2　ＭＤＡのメタモデル

5.3.3 MOF

各々の PIM，PSM 間の変換を実現するためには、変換ルールが必要である。そのためには、各々の PIM, PSM が同じ手法で規定されて全体として一つのメタモデルを構成している必要がある。このメタモデルを記述するメタ・メタモデルとして、図 5.3-2 では、MOF, UML,その他の言語が例示されているが、その中心となるものが MOF である。

MOF(Meta Object Facility)は、1997 年に OMG により定められた、メタモデルを規定するためのメタ・メタモデルである。MOF は、CORBA 等との親和性が高く、PIM だけでなく PSM を規

定するのにも適したメタ・メタモデルであると言われている。

すべての PIM，PSM のメタモデルは、この MOF により規定されることにより、共通の土俵に載り、それらの間の変換ルールも MOF により規定されるようになる。例えば、最も中心的な PIM の1つである UML のメタモデルは図 5.3-3 のようになり、また、PSM の1つである CORBA のコンポーネントのメタモデルは図 5.3-4 のようになる。

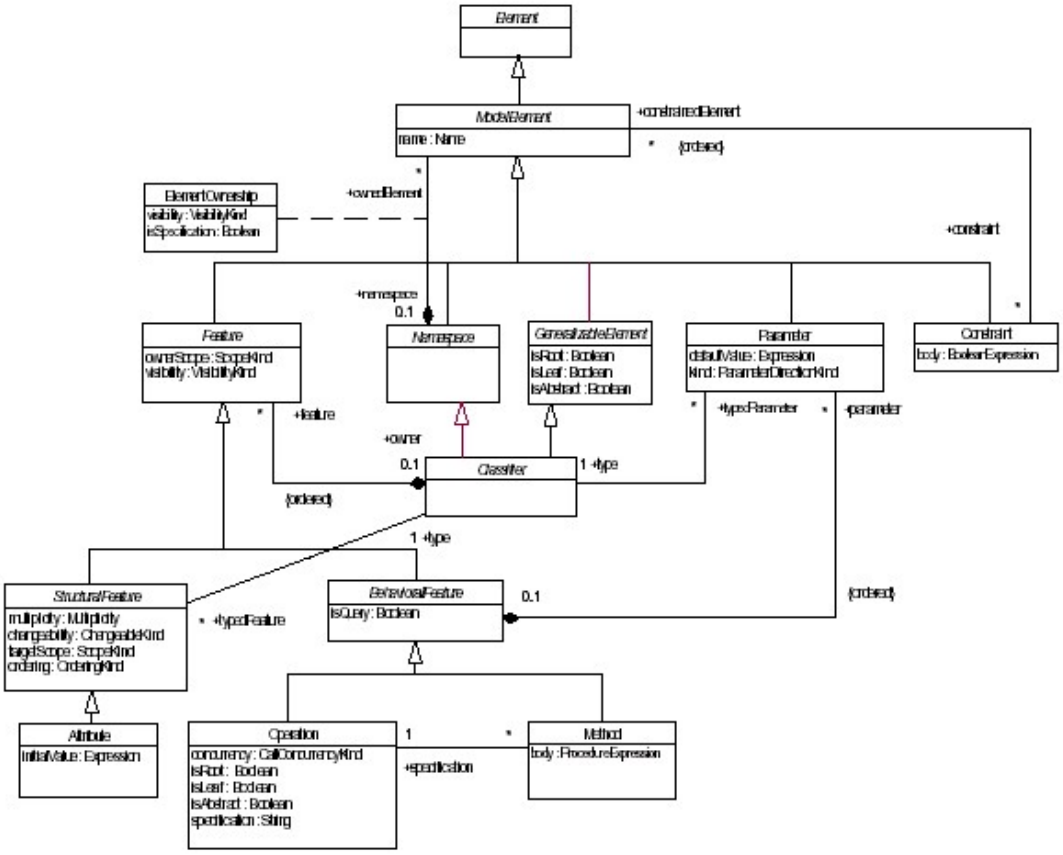


Figure 2-5 Core Package - Backbone

出典：OMG Unified Modeling Language Specification version 1.5

図 5.3-3　UMLの骨格のメタモデル

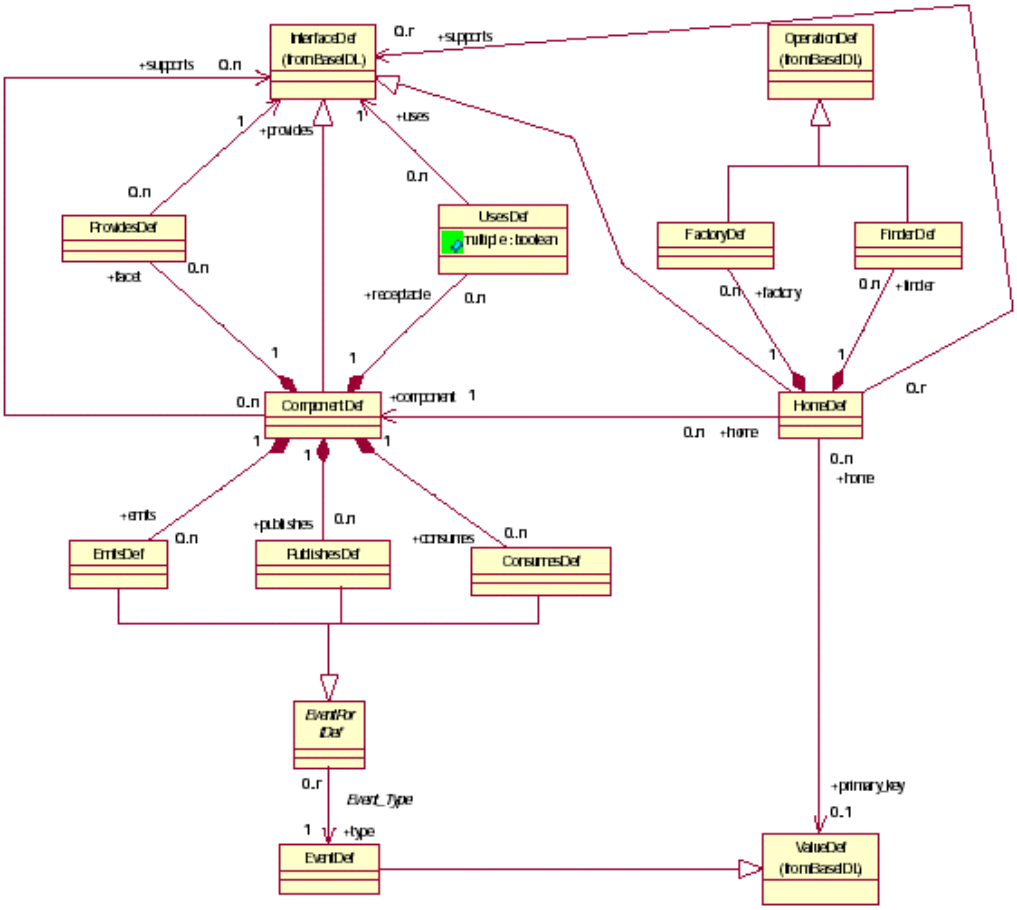


Figure 8-12 ComponentIDL Package - Main Diagram

出典:OMG CORBA Components version 3.0

図 5.3-4　ＣＯＲＢＡコンポーネントのメタモデル

5.3.4 UML

MDA は、PIM，PSM を規定するモデリング機能を規定していないが、PIM に関しては、その中心となるものが、UML であることは疑いのないところである。ただし、現状において、UML は完全にはMOF により規定されてはいない。これは、UML が、もともと、必ずしもフォーマルな定義がなされていたものではなく、OMT 法、OOSE 法、Booch 法が融合され実務的に発展してきた経緯によるものと思われる。その後、OMGにより、必要な精密化・拡張がなされてきたが、現状最新の version1.5 においても、MOF により定義されているのは一部にすぎない。最近正式にリリースされた UML version 2.0 において、UML は初めて MOF により完全に規定されることになる。

5.3.5 UML Profile

PIM モデルは、プラットフォーム独立とはいえ、特定のプラットフォームに依存した PIM モデルに変換され、その実装環境でのコードを実際に生成しうるだけの詳細性・厳密性が求められるモデルである。

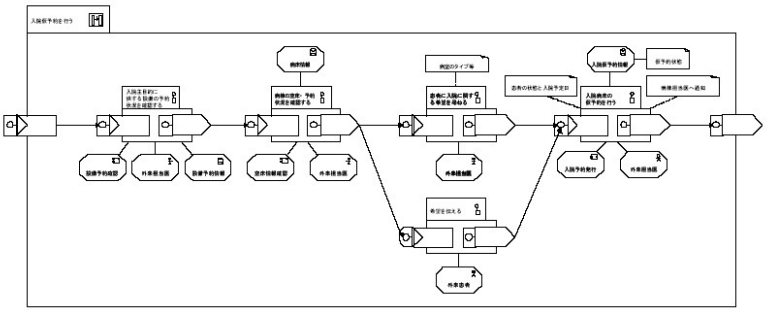
即ち、ユースケース図、クラス図、イベント・シーケンス図では十分ではなく、イベント・シーケンス図だけでは表現きれない各クラスの状態遷移図や、イベントを受けて実効される各クラスのメソッドの内容、その事前条件・事後条件等、一切合切を含んだものを意味する。

さらに、例えば、分散コンポーネント環境を意識すれば、PIM は、J2EE、.NET、CORBA などに関わらず、その PSM に変換されうるだけの情報量を持たなければならない。そのために、UML が、PIM を規定するうえでの必要な機能すべてを具備しているわけではない。一例を挙げれば、メタクラスとして、UML の class という概念では足らず、Process Component, Entity 等のサブ・メタクラスが必要となってくる。

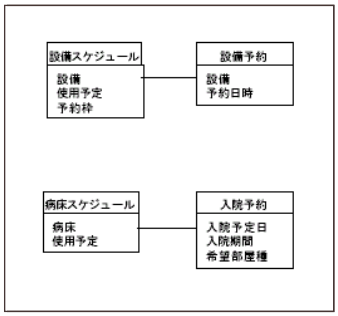
このような UML に対する必要な機能拡張は、本来的には、MOF により規定されるべきものであるが、現状、UML が完全には MOF で規定されていないこともあり、MOF によることなく UML 自身に備わっているプロファイルという機能により、必要な機能拡張を行う努力がなされている。ちなみに、上の例では、プロファイル機能により<<Process Component>>, <<Entity>>というステレオタイプを付加することにより、MOF による UML メタモデルを変更することなく実現される。

その一例が UML profile for EDOC である。EDOC とは、Enterprise Distributed Object Computing の略で、企業レベルでの分散オブジェクト環境を意味する。具体的には、UML profile for EDOC では、PIM として ECA(Enterprise Collaboration Architecture)を採用している。ECA はオープンな分散処理システムの枠組みを規定する国際標準である RM-ODP (Reference Model for Open Distributed Processing)に基づいており、エンタープライズ・ビュー、インフォメーション・ビュー、コンピュテーショナル・ビューの3つの視点から PIM を構成している。

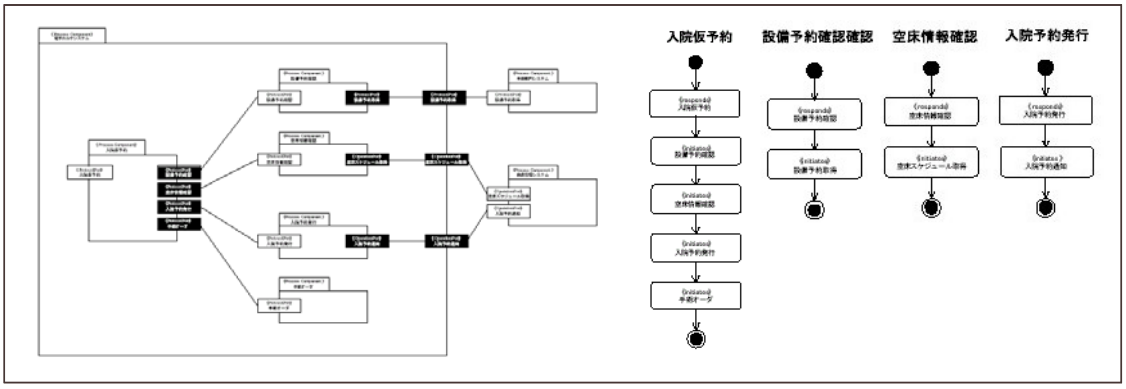
エンタープライズ・ビューでは、企業レベルでの業務プロセスを描き、インフォメーション・ビューで個々の業務プロセスの静的構造や動的振る舞いを描き、最後に、コンピュテーショナル・ビューにおいて、分散オブジェクト環境における構造が導入されている。この内、インフォメーション・ビューは、通常よく使われる範囲のUML近いが、エンタープライズ・ビューは企業レベルでの業務記述のために、また、コンピュテーショナル・ビューは、分散オブジェクト環境での構造の記述のために、拡張されており、特に、エンタープライズ・ビューにおいては、通常のUMLとは大きく趣を異にする。図 5.3-5 にその一例を挙げる。



＜エンタープライズ・ビューにおけるビジネス・プロセス・モデル＞



＜インフォメーション・ビューのモデル＞



＜コンピュテーショナル・ビューのモデル＞

出典:コンポーネント標準化による電子カルテ開発に関する研究・分担研修報告書

図 5.3-5　ＵＭＬProfile for EDOC によるモデルの例

さらに、UML profile for EDOC では、これらの拡張された PIM の情報を用いて、同様に、UML profile for Java and EJB により規定された Java/EJB の PSM や UML profile for CORBA により規定された CORBA の PSM への変換が可能になるとされる。そのメタモデルの概略を図4．2－6に示す。

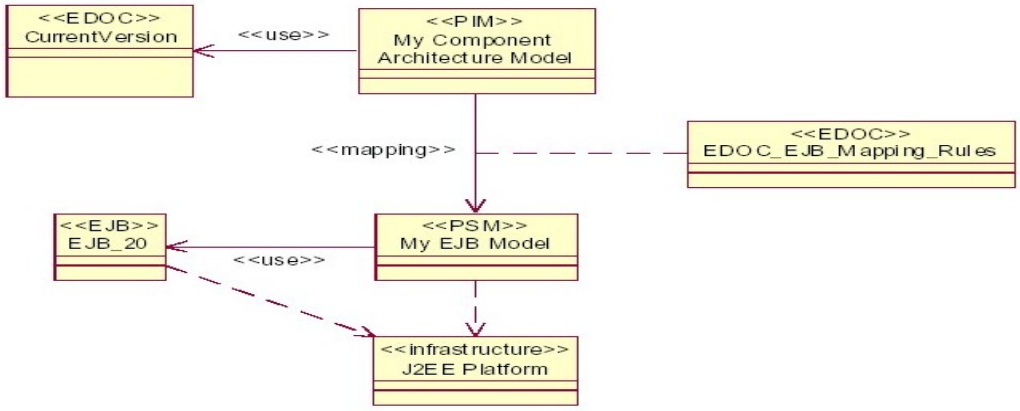


Figure 9.　*MDA Metamodel Example*　　出 典 ：ＯＭＧ　ormsc/2001-07-01

図 5.3-6　ＥＤＯＣによるＰＩＭおよびJava／EJB での PSM のメタモデルの概略

5.3.6 MDA ツール

現在、MDA ツールを標榜する各種ツールが、ArcStyler を代表として存在する。しかしながら、これらは UML をサポートする CASE ツールであり、さらに、複数のプラットフォームに対して実装システムの生成機能を有することは事実としても、上に述べたように意味での MDA をサポートしているかどうかは疑わしい。

また、これらのツールは、あくまで実装システムの生成を意識している点が重要である。

MDAのそもそもの狙いである、既存システムをリバース・エンジニアリングすることで PIM を作成し、そこから改めて新たなプラットフォーム上のシステムを生成すると言ったプラットフォームの変遷に柔軟に対応できるというメリットは、サポートされていないようである。

5.3.7 ユーザとして留意すべき点

OMG が謳っているように、MDA を活用する事によって、実装プラットフォームから独立し20年間持続するソフトウェア・アーキテクチュアが実現される。更にその上に、ビジネスロジックが表現され、レガシィー・システムや XML/SOAP, J2EE, .NET 等の様々な実装プラットフォームに惑わされることなく、既存システムと新規システムの速やかな統合、将来に渡ってのシステム保守コストの削減が実現されれば、ユーザにとってのメリットは計り知れない。

ただし、現状は、UML profile for EDOC を見ても、その道のりは必ずしも平坦ではないように思われる。

UML profile for EDOC が依拠している PIM モデルである ECA は、国際標準 RM-ODP に基づくものであるにせよ、これが、今後世界において実質的に標準的な地位を確立し、20年間持続するソフトウェア・アーキテクチュアであるかは定かでない。

また、ECA は、RM-ODP に基づいているためか、あるいは、CORBA や Java/EJB 等の PSM への変換を意識するためか、これらを包含する抽象化されたソフトウェア・アーキテクチュアではあっても、ユーザが想起するような実装プラットフォームから独立したソフトウァエ・アーキテクチュアであるかどうかも疑念が残る。

少なくとも、通常のUMLになじんでいる者にとっては、エンタープライズビューにおけるビジネス・プロセス・モデル等は違和感を覚えるものである。さらに、ECA は、UML profile for EDOC が依拠している PIM の一つであり、今後様々なPIMが乱立する可能性もある。

このような状況は、UML version 2.0 により大きく改善されるかもしれない。ただし、すでにUMLversion 1.5 の仕様書は 700 余ページに及び、version 2.0 においてはさらに拡充される。その全体から見れば、通常ユーザが使用しているユースケース図, クラス図, イベント・シーケンス図などは極々わずかでしかない。UML が如何に MDA を意識して拡張され完全なものになり、それにより、真の意味での MDA が実現されうる環境が整ったとしても、PIM の作成のためには、通常ユーザが業務および業務システムの概要を理解するために使用しているUMLの極一部だけでは到底不足であり、膨大な情報を UML に従い登録する必要があることを忘れてはならない。

ユーザとしては、PIMの作成のために必要となる労力との比較においても MDA により実現される効果が色褪せぬために、使いやすいPIMが主流になるよう、今後の動向を十分注視し、かつ、その実現のために積極的な発言をしていくことが必要である。

少なくとの通常のUMLに馴染んだものからすれば、違和感を伴うものであり、プロファイルの整備により地道に進んでいる。

ただし、現状、MDAは、ビジネスロジックを実装プラットフォームから独立させることにより、20年間持続するソフトウェア・アーキテクチュアを実現し、その結果として、既存システムと新規システムの速やかな統合、システム保守コストの削減を実現するとされている。

その背景としては、近年、CORBA, XML/SOAP, J2EE, .NET 等の様々な実装プラットフォームが生まれ、変遷していく中で、ビジネスロジック自体に変化がなくとも、業務システムをこれらの実装環境の変遷に追随させるために、多くのコスト・時間を要しているという現実があげられる。

現状では、MDA は、UML for EDOC に見られように、プロファイルの整備により地道に進んでいる。また、それとは別に、複数の実装環境に対応した MDA ツールと称する UML ツールも普及しつつある。さらに、近々正式にリリースされるであろう MDA を強く意識した UML version 2.0 により状況は大きく進展するかもしれない。

ただし、MDA は壮大なビジョンである。すでに UMLversion 1.5 の仕様書は 700 余ページに及び、version 2.0 においてはさらに拡充される。その全体から見れば、通常、我々ユーザが使用しているユースケース図, クラス図, イベント・シーケンス図などは極々わずかでしかない。
つまり、UML が如何に MDA を意識して拡張され完全なものになり、それにより、真の意味での MDA が実現されうる環境が整ったとしても、PIM の作成のためには、通常我々ユーザが業務および業務システムの概要を理解するために使用しているUMLの極一部だけでは到底不足であり、膨大な情報を UML に従い登録する必要があることを忘れてはならない。

PIM を作成するために必要となるであろう労力との比較においても、MDA により実現される効果が色褪せないものでありうるかどうかは、今後の動向を十分に見極める必要がある。

5.4　ＣＢＤ（コンポーネントを中心とした設計手法）

コンポーネントを中心としたシステム開発技法が注目を浴びている。いわゆる、CBD（Component Based Development）である。コンポーネントの再利用を前提とした開発方法論としては、I. Jacobson の「Software Reuse」で解説されている RSEB(Reuse-Driven Software Engineering Business)が CBD の前身的位置付けに相当する。

CBD を理解するためには、この RSEB を理解する方が近道である。これについては、後述するとして、まずコンポーネントとは何なのかを明確にしておきたい。

5.4.1　コンポーネントとは何か

R. Orfali らの定義によると、コンポーネントを「The Essential Distributed Objects Survival Guide」で以下のように定義している。このガイドは、分散オブジェクトの技術解説書として書かれている。その為、分散オブジェクトとの関係でコンポーネントを解説しており、技術的にはボトムアップのコンポーネント定義である。

> コンポーネントはネットワーク、アプリケーション、言語、ツール、オペレーティングシステムの境界を超えてプラグアンドプレイできるスタンドアロンオブジェクトである。分散オブジェクトの定義からして、そのパッケージング方法をみれば、分散オブジェクトはコンポーネントであることがわかる。分散オブジェクトシステムでは処理と配布の単位はコンポーネントである。分散オブジェクト基盤はコンポーネントがより自律性を持ち自己管理を行ない強調動作することができる環境を整えなければならない。

一方、RSEB を提唱した I. Jacobson の「Software Reuse」では、コンポーネントを次のように定義している。

> コンポーネントは再利用を前提に設計、ドキュメント化、パッケージ化された高品質のタイプ、クラス、その他の UML の生産物である。コンポーネントは結合力があり安定したインタフェースを持っている。コンポーネントはインタフェース、サブシステム、ユースケース、タイプ、アクタ、クラスのいずれかであり、属性タイプが含まれている。コンポーネントはまたテンプレートやテストケース仕様など(UML 以外の)他の生産物も含んでいる。

ここでは、コンポーネントをベースとして企業のインフォメーションシステムを構築する方法に

ついて説明している。従って、R. Orfali の言う分散オブジェクトというプログラムだけでなく、その分散オブジェクトに関してのドキュメントやテストケース仕様といった再利用時に必要なマテリアルも含めてコンポーネントと呼んでいる。

言い換えると、コンポーネントとは再利用可能である。また自己完結しており、特定のアプリケーションに依存しない。それ自体が独自の振る舞いを持ったオブジェクトある。つまり、カプセル化、インタフェースのインヘリタンス、ポリモーフィズムをサポートするという点において真の意味でのオブジェクトである、と言うことができる。

R. Orfali らの「Client/Server Survival Guide (3rd) 」では、コンポーネントの性質・特徴について以下のものを挙げている。

- 商品として流通する単位であること
- それ自身はアプリケーションではない
- どのような組合せで使われるか事前に予測できない
- きちんと設計されたインタフェースを公開していること
- ツールによる操作が可能であること
- イベント通知機能を持つこと
- コンフィグレーション/プロパティ管理機能を備えていること
- スクリプト言語による操作が行えること
- メタデータとイントロスペクションが用意されていること
- 使い易いこと

また、コンポーネント指向の開発方法論としてカタリシス法(Catalysis)がある。カタリシスとは「触媒作用」あるいは「触媒反応」の意味である。カタリシス法は、ICON Computing 社の Desmond D'Souza と Trireme 社の Alan Cameron Wills が提唱している「コンポーネント指向のオブジェクト指向開発方法論」である。

カタリシス法の特徴は、コンポーネント指向とフレームワークを取り入れている事である。これにより、エンタープライズレベルのシステムにまで対応出来るようになる。

また、システムの実装が分散オブジェクト技術の CORBA や DCOM を採用している場合では、設計から実装へとシームレスにつなぐことができる。

さらに、UML ベースの表記を採用しているので、ダイヤグラムは UML の知識があれば、理解することが出来る。

5.4.2 RSEB における開発体制

I. Jacobson の「Software Reuse」で解説されている RSEB (Reuse-Driven Software Engineering Business) が CBD の前身的位置付けに相当することは先に述べた。RSEB は

OOSE(Object-Oriented Software Engineering)をコンポーネントビジネス用に拡張したものである。これは、コンポーネントの再利用を前提とした開発方法論となっている。
RSEB では、目的に応じて開発手法を AFE、CSE、ASE の3つに分類している。

(1) AFE(Application Family Engineering)

システム全体のレイヤ構造に関する設計手法である。サブシステムの特定、使用するアプリケーションの選択など、システム全体のアーキテクチャに関する開発を行う。オブジェクト指向を熟知していることが前提となり、最も高いスキルが要求される。また、ソフトウェア工学全体とターゲットとなる業務に対する広範で充実したスキルが要求される。AFE における主要な開発要員は以下のとおりである。

- リードアーキテクト
- ディストーリビューションエンジニア
- スーパオーディネートユースケースエンジニア
- スーパオーディネートサブシステムエンジニア
- GUI コーディネータ
- テスタ
- ソフトウェアエンジニアリングビジネスマネージャ

(2) CSE(Component Service Engineering)

コンポーネント開発の手法を決めており、どのように再利用可能なコンポーネントを開発すべきかを述べている。オブジェクト指向を熟知していることが前提となり、中程度のスキルが要求される。CSE における主要な開発要員は以下のとおりである。

- サブシステムエンジニア
- ユースケースエンジニア
- ユースケースデザイナ
- 再利用プロセスエンジニア
- 再利用サポート環境エンジニア
- コンポーネントシステムライブラリアン
- テスタ

再利用プロセスエンジニア、再利用サポート環境エンジニア、コンポーネントシステムライブラリアンは開発されたコンポーネントを再利用できるような環境の整備を行なう。

(3) ASE(Application Service Engineering)

個々のアプリケーション開発の手法である。AFE で決められたアーキテクチャに則り CSE で開発されたコンポーネントを利用してアプリケーションの開発を行う。高いスキルは要求されない。ASE における主要な開発要員は以下のとおりである。

- アーキテクト

- サブシステムエンジニア
- ユースケースエンジニア
- ユースケースデザイナ
- アプリケーションエンジニア
- テスタ
- 製造者
- コンポーネントシステムトレーナ
- コンポーネントシステムサポータ
- 顧客
- エンドユーザ

アプリケーション開発におけるコンポーネントシステムトレーナとコンポーネントシステムサポータはコンポーネントをアプリケーションに組み込むための情報を提供するエンジニアである。クックブックの執筆や講義といった形でコンポーネントの使い方に関する情報の提供を行なう。

RSEB の考え方で重要な点は、再利用のための専任のエンジニアが用意されていることと、アーキテクチャを設計する部隊、コンポーネントを開発する部隊、アプリケーションを開発する部隊の役割が明確になっていることである。

5.4.3 カタルシス法によるコンポーネント設計

カタルシス法は、Alan Wills と Desmond D’ Souza によって提唱されたコンポーネント指向の開発方法論である。Desmond D’Souza が書いた「Objects, Components, and Frameworks with UML(the Catalysis Approach)」が原典となっている。

カタルシス法では、UML でのユースケースを「アクション」と呼び、オブジェクトの相互作用を意味するものとして定義している。また、クラスに相当するものを「タイプ」と呼び、アクションと混在して表記する。カタリシス法によるモデル例を次に示す。

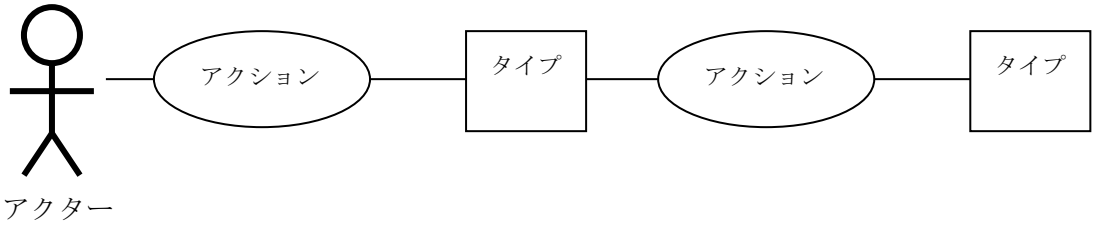


図 5.4-1

また、カタルシス法ではコンポーネントを次のように表記する。

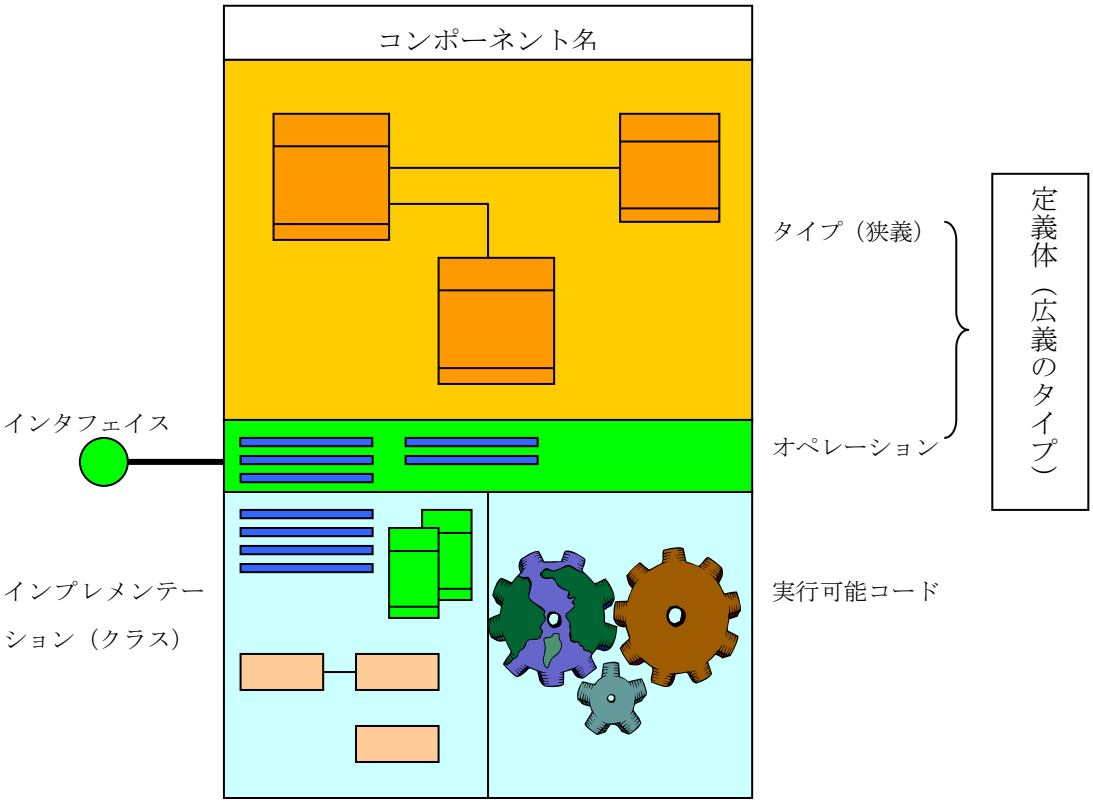


図 5.4-2

各コンポーネントは、少なくとも 2 つ以上のパートから構成されなければならない。

- インプリメンテーション（デザインされたクラス群、モジュール、実行単位、他）
- タイプとして定義されたコンポーネント・インタフェース

.

カタルシス法には 3 つの構成要素がある。

- コラボレーション
- タイプ
- リファイン

コラボレーションでオブジェクトのグループの振る舞いを整理し、タイプはオブジェクトの外部に対しての振る舞いを決め、リファインで振る舞いなどを抽象化してゆく。

その後、プラグイン可能なフレームワークを作成してゆく。

先にも触れたが、カタリシスではオブジェクトの抽象レベルをタイプと呼んでいる。また、タイプ間にある機能をアクションと呼んでいる。アクションはUMLで言うところのユースケースにあたる。例えば「預金」と「利息」の間には、金額と期間と利率によって利息を計算するというアクションがあり、この種類の関係をコラボレーションと言う。これを図で示すと次のようになる。

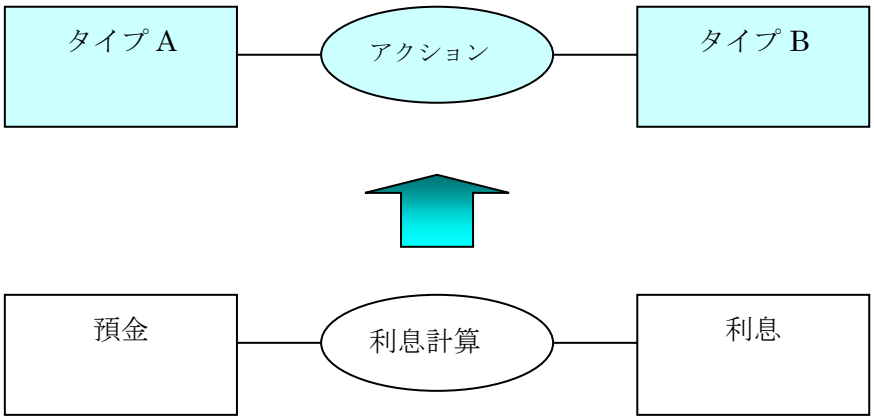


図 5.4-3

タイプとは、オブジェクトやグループ化されたオブジェクトの抽象レベルを言いう。タイプの表記は UML のクラス図を拡張している。上から順番に「タイプ名」、「属性、またはモデルの図」、「このタイプが持っているオペレーション」を記述する。オペレーションとは、タイプの外部から起動される振る舞いである。

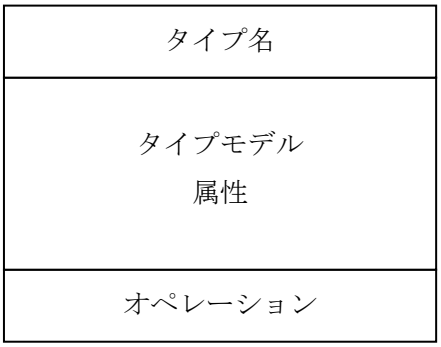


図 5.4-4

リファインとは「洗練」を意味し、抽象化のレイヤを指している。これにより、具像化したものから抽象化したものへとマッピングすることを、明確に表現できる。次の例では、「登録」「変更」「削除」を「申し込み」というアクションにリファインしている。

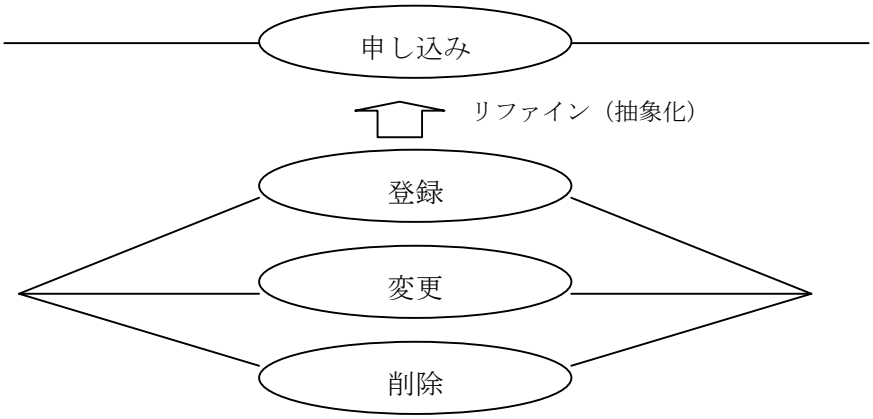


図 5.4-5

これまでに述べてきた、カタルシス法で言うコラボレーション、タイプ、リファイン、及びフレームワークの全体構成を次に示す。

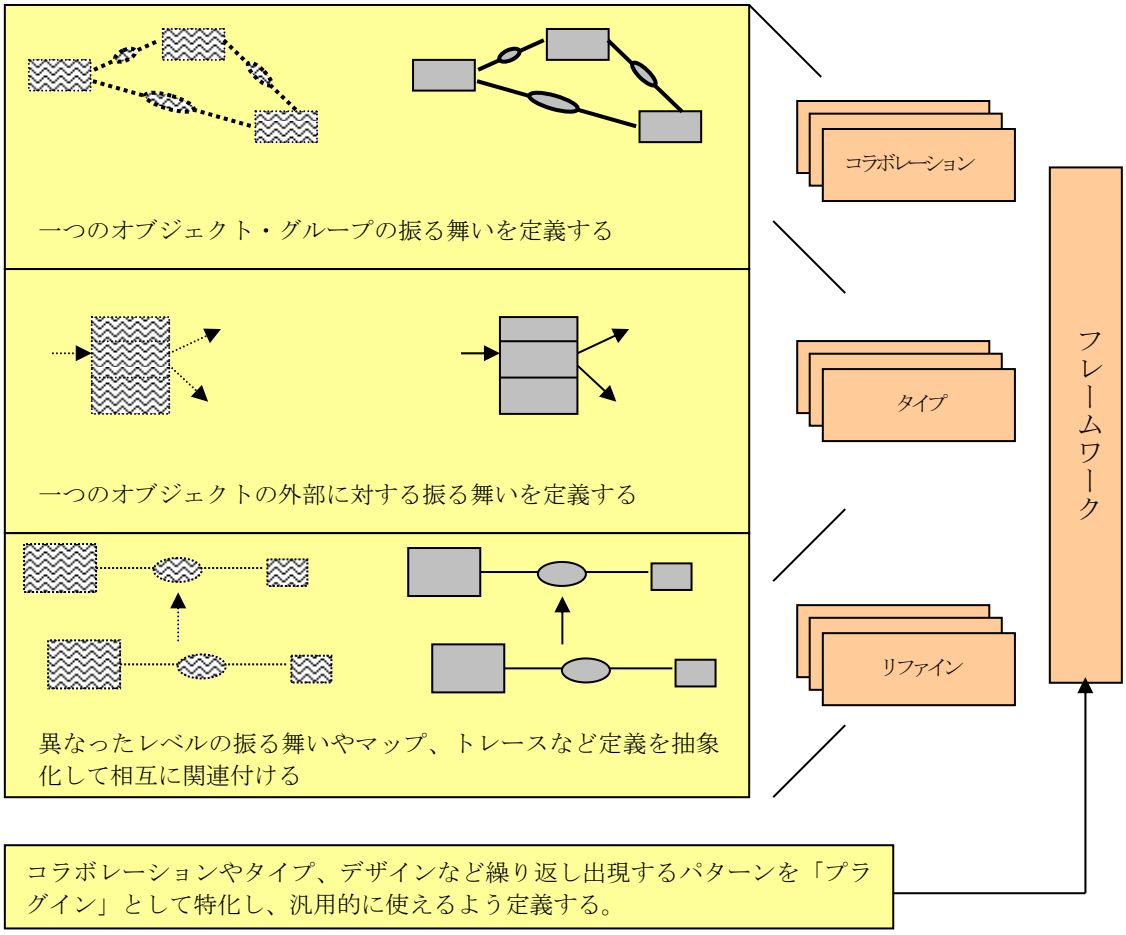


図 5.4-6

カタルシス法による開発プロセスも提示されている。カタリシス法による開発過程は、図のような三つのモデリングプロセスに分類される。

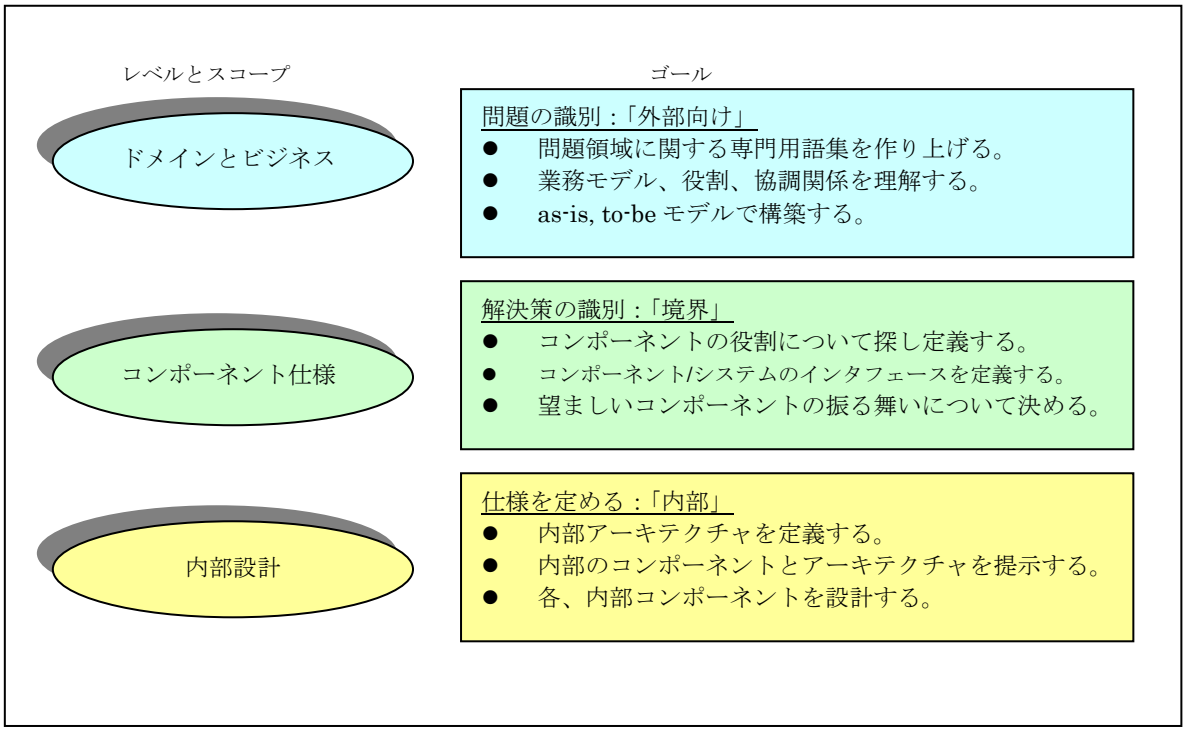


図 5.4-7

ドメインとビジネスは、最初のプロセスである。ここでは、ビジネスプロセスなどの業務の流れを明確にし、システム化の対象となるコンテキストを絞り込み、それと関連する要素を分析していく。

次のプロセスがコンポーネント仕様となる。最後がコンテキストの内部を分析・設計していく内部設計である。

これらの 3 つのプロセスを具体的な例で示す。例題として、オンライン・ショッピングを利用した Web からの商品注文を考える事とする。

① 最初のプロセスは、ビジネスプロセス全体を整理する。このことで、全体の流れを明確にできる。顧客は注文を行なう。これをユースケース図で表わすと、次のようになる。

② ビジネスプロセス全体を表現する。このプロセス全体におけるシステム化の範囲を明確にする。この例では、太枠点線で囲った部分がシステム化対象となる。顧客管理は、別システム(別アクター)とする。

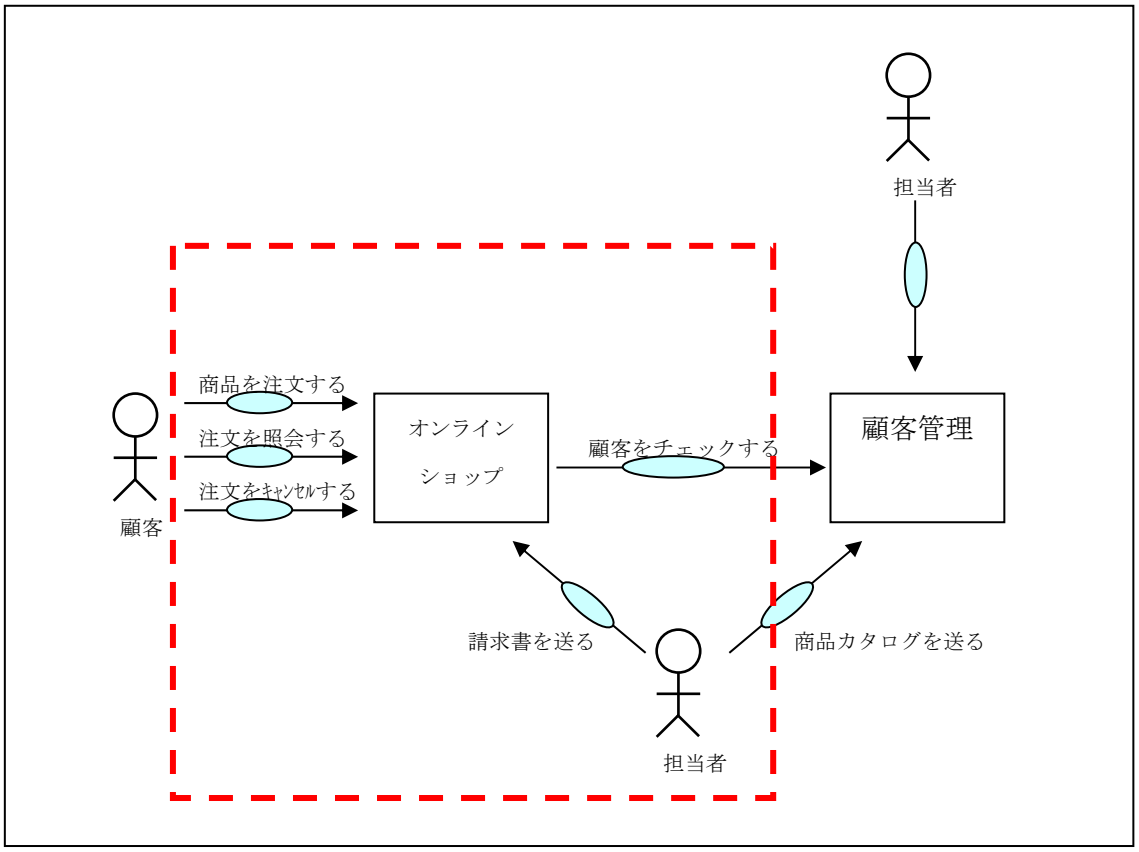


図 5.4-8

③ カタルシス法では、システムをコンポーネントと考えるので、システムの入出力に注目する。つまり、システムの入力に対して、コンポーネントが「振る舞い」と言うコンテキストを持つわけである。このコンテキストを正確に洗い出すために UML のシーケンス図を利用する。

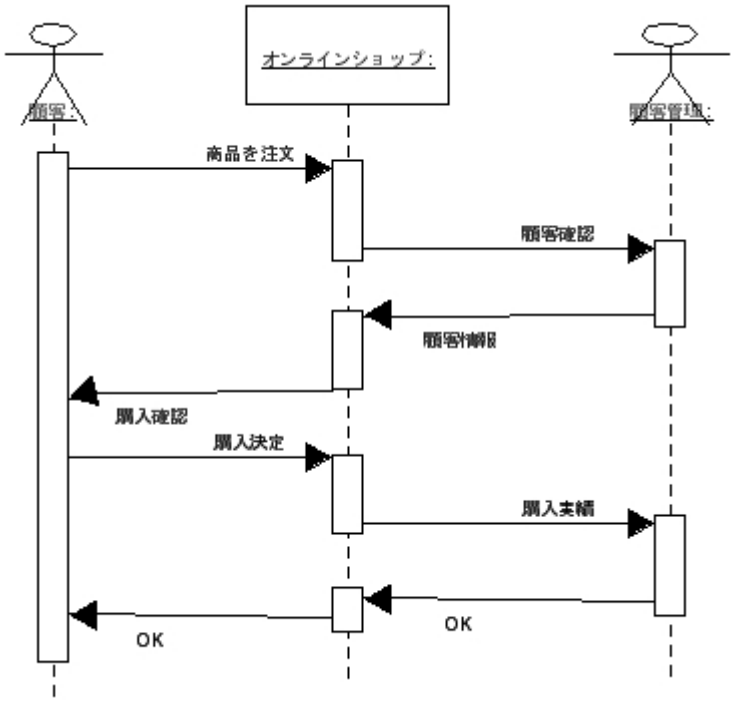


図 5.4-9

④　このシーケンスチャートを元に、オンラインショップのコンテキストを決定する。

オンラインショップ
商品を注文（） 顧客情報確認（） 購入確認（） 購入実績送信

図 5.4-10

内部設計では、このコンテキストの中を詳細化して行く。これには、制約(Constraint)を用いる。制約を記述する言語は米 IBM 社の Insurance division でビジネスモデリング用の言語として開発された OCL (Object Constraint Language)を用いる。

OCL は UML の一部で、オペレーションのシンタックスは次のとおり。

タイプ名	オペレーション	説明
Pre	条件	Pre とは Pre-condition のこと。事前条件を指す。このオペレーションを起動できる条件を意味している。
Post	状態	Post とは、Post-condition のことで事後条件を指す。オペレーションを起動した結果の状態をある。

表 5.4-1

内部設計におけるオペレーションの定義例を以下に示す.

オンラインショップ： 商品を注文する（）
Pre:ログインされていること
　　注文情報が入力されていること
Post:顧客情報を顧客管理に問い合わせる

オンラインショップ：顧客情報の確認
Pre:顧客のコードが特定されていること
Post:正しい顧客コードであることが確認される。
・
・
・

⑤　コンテキストを完成させる。

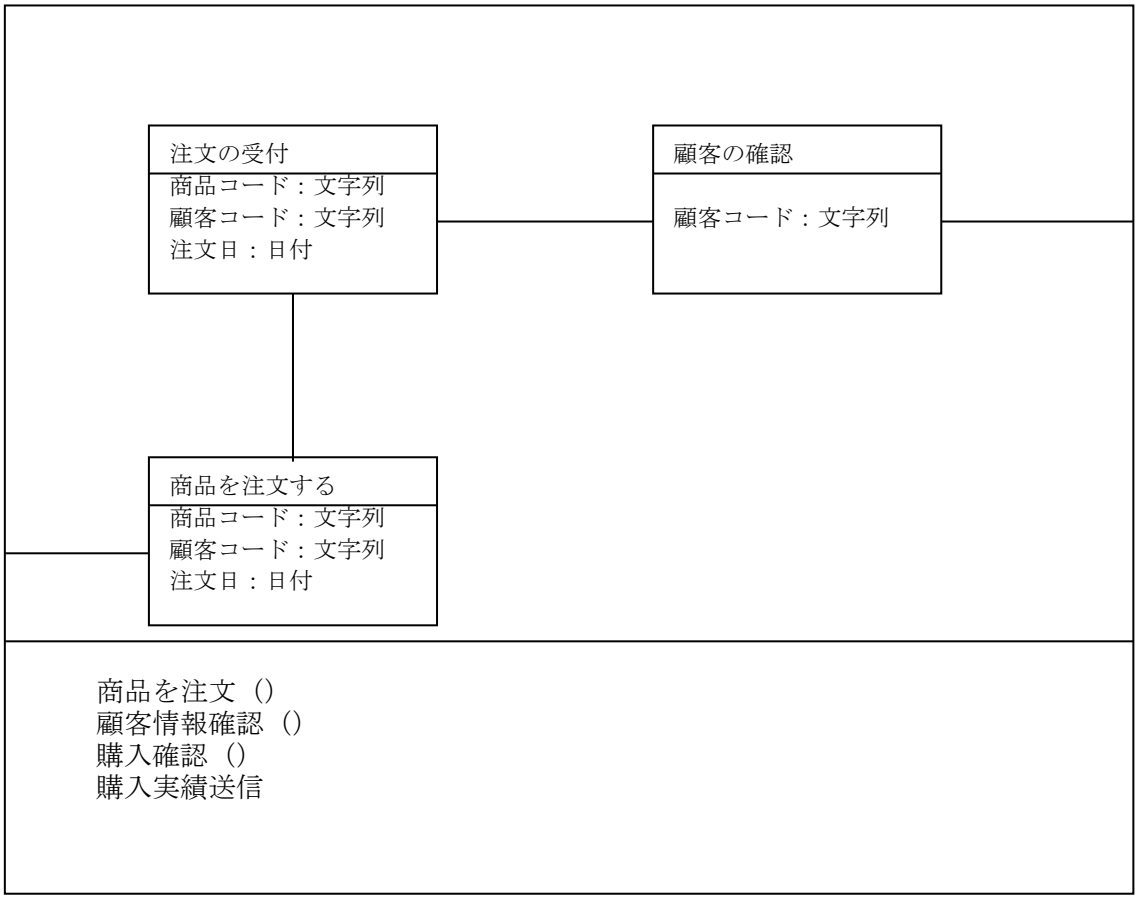


図 5.4-11

⑥　次にタイプモデルと状態モデルを作る。外部とのインタフェースを持つタイプには、外枠との間に実線を引く。

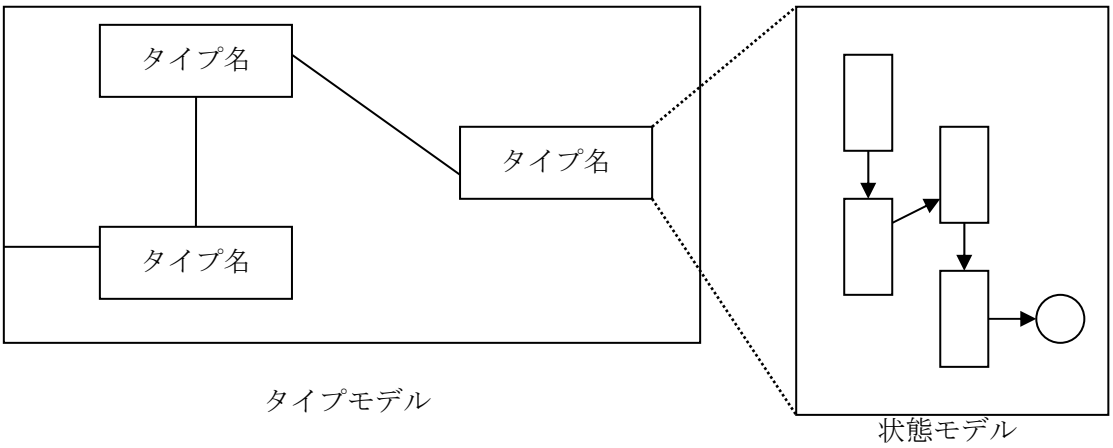


図 5.4-12

⑦　コンポーネントの設計

タイプモデルは抽象化されたモデルであるため、どのようなアーキテクチャ上に実装するかといったことは考慮していない。

実装については、コンポーネントの設計のフェーズで始めて検討を行なう。このことをリアライゼーション（具現化）と呼んでいる。

リファインとリアライゼーションは、システムを最適化するためにしばしば繰り返される。これは統一プロセス（UP）でいうイテレーションに似ている。

⑧　モデルフレームワーク

タイプモデル作成する時に、似たようなパターンが発見できる場合がある。カタリシスのモデルフレームワークは、モデリングにパターンを発見することで、効率の良いモデリングを行うことを狙っている。

参考文献：http://www.catalysis.org/、“Components, and Frameworks with UML”（Desmond D’Souza）

6 あとがき

当研究会も2年目に入り、多くの参加メンバにも恵まれ、いよいよ佳境に入ってきた。
全員がユースケース、クラス図、などの単語にも慣れて一段と議論に熱が入る。
「中学生にも分かるオブジェクト指向設計ガイド」を最終的に作成することが、このグループの究極の目標ではあるが、まず基本を知らねばならい。就業後、月2回集まって、夜遅くまで議論を重ね理解を進めてきた。ともに考え、悩んだ仲間としての感想を述べたい。

「誰が作っても同じようなクラス設計が出来ないものか？」を1年目の課題に取り上げたが、継続審議になり今年も議論を詰めてきた。本編を熟読いただくとそのあたりの苦労、努力が分かっていただけるものと思う。
議論が行き詰まってきたときは、
「何故今これが問題になっていますか？」
「何故答えに差が出てくるのでしょうか？」
「汎用的にするためには、どのようなアイデアが必要ですか？」などとリーダが問いかけた結果が本編に随所に顔を出している。読み取っていただければ有難い。

さて、当協会はユーザのための協会である。
「ユーザ業務の改革のために、UMLは役に立つのか？」をいつも念頭に入れて議論をしているが、ユースケース図から始めると、業務改革のきっかけを見つけにくいことが徐々に判明してきた。
ビジネスモデルの改革のためには、ユースケース図の前に、もう少し業務分析用のドキュメントが必要なことが判明しそれは何なのか？の議論が激しく行われた。
本編の中に「情報の流れ」と「物の流れ」を分けて書くことが業務改革には有効であり、そのサンプルがかかれているが、これに「金の流れ」を別に書いてみることでそれぞれの問題が見えてくる。
サンプルとして取り上げた「ワイナリー問題」もこのように、問題解決のための作成ドキュメントのあり方として眺めて見ると、ユースケース図以前のプロセスでの分析資料の重要性が見えてくる。これは非常に興味ある課題であった。
これらの指摘は、別のプロジェクトである「ユーザによるビジネスシステム定義プロジェクト」（通称RFPプロジェクト）にも反映させることが出来た。
問題分析があって、さまざまな開発手法を利用した解決策に結びついてゆくが、出発点は問題分析である。
保険業務の分析もその一環で行われ「IDEF手法」も学ぶことが出来た。

そのような有意義な成果の残せた2年目であり、自画自賛かも知れないが、当プロジェクトの内容の充実は図れたと思う。
これからオブジェクト指向設計を進めようとしている企業の方達は、是非一度「ワイナリー問題」を一貫して演習問題として取り上げ、当ガイドを元に、実際に解決策を自らが書いてみることをお勧めする。私達が悩んだのと同じ課題に遭遇し、解決策も有効に参照していただけるに違いない。そしてその結果をJUASに是非寄せていただきたいと願っている。そうすればますますこの報告書は充実してくるに違いない。

このプロジェクトには、いろいろな意味での才能のある方が多い。

「日本語は設計書に書く言語として何が問題か？」

「貴方の書いた意味は、このようなことか？」などが、頻繁に議論して出てきたときに「象は鼻が長い」の主語は何か？と言うことが議論になった。

おかげで三上章の同じ題の本を読まされることになり、日本語は主語述語が明快でなくてもなんとなく意識は通じる素晴らしい言語である反面、SEの文章の書き方を考えさせられた。単にオブジェクト指向設計技術を議論するだけでなく幅広いものの見方、考え方が出てくるのが、企業内で狭い範囲の方と議論していることとの大きな差である。
是非人間的にも、技術的にも成長したい方の、次年度へのこのプロジェクトの参加を期待したい。
さて次年度は、いよいよ最後の年に入る。
「ユーザのための実装問題も含めたオブジェクト指向設計技術」の研究成果が楽しみである。
ユーザ協会ならでの検討成果を期待してやまない。

以上

JUAS専務理事　細川　泰秀

平成 15 年度　ビジネスオブジェクト検討プロジェクト
「ユーザが考えるビジネスオブジェクト研究報告書」
～これからのＩＴ投資を考えるユーザのために～

発行日：平成 16 年 4 月
発行所：社団法人　日本情報システム・ユーザ協会
〒103-0001　中央区日本橋小伝馬町 15-17 ASK 日本橋ビル 5 階
℡03-3249-4101　Fax03-5645-8493
URL http://www.juas.or.jp/

この事業は、オートレースの補助金を受けて実施したものです。