UMLで表現された発明の明細書における一考察

平成18年度ソフトウエア委員会　第1部会
（石渡清太*，市原政喜*，内島　裕，川上桂子，川端純市*，川村　武*，塩島利之*，
永田美佐*，長谷川　靖*，原田一男*，福永正也*，松下　正*，三島広規*，吉澤弘司）

目　次

1．はじめに

近年，コンピュータ・システムの開発の手法は，開発システムの複雑化・大規模化等に伴い多様に変化してきている。例えば，開発フェーズが滝の流れのように一方向に進むウォーターフォール型の開発形態から，一定期間毎に分析，設計，実装，テストなどを，実際に実行可能なソフトウエアを用いることで反復的に行い，その動作を確認しつつ評価することによりフィードバックを得ながら開発を進める，いわゆるアジャイル（Agile）と呼ばれる開発形態へと移行しつつある。また，ソフトウエアプログラムを部品化し，部品の再利用を促進することにより開発効率を高めるため，オブジェクト指向技術が一般的に用いられるようになっている。オブジェクト指向言語としては，C＋＋だけではなくJava（登録商標）やC＃が開発され，広く利用されるようになっている。また，アジャイル開発との親和性からRuby等の簡易言語も注目されている。さらに，わが国では，システム化する業務をモデル化する上流工程は自ら行うものの，コーディング等の実装作業は，インド，ベトナム等の工数単価が安価な海外の企業へ外注する企業の割合も増加傾向にある。

このような状況において，発明に相当する新しいアイデアは，抽象度の高い上流設計部分にて生ずることが多い。しかし，このような新しいアイデアを，旧来のようにフローチャートで表現することは少ない。実装作業を行う必要が無いことから，後に述べるUML等を用いてシステムの本質を表現し，フローチャートを作成することなくシステムの完成を見る機会も増大している。

一方，コンピュータ・ソフトウエア関連発明に関する明細書を作成する場合，請求項において，ソフトウエアとハードウエアとの具体的な協働を記述することが発明の成立要件として求められており，単なるコンピュータの利用に過ぎない場合には，法上の発明とは認められない。オブジェクト指向技術を用いる場合であっても例外ではなく，いわゆるオブジェクトを用いていることを請求項上明示しただけでは，法上の発明として認められない。

われわれは，システム開発手法の変化に伴い，抽象度の高い上流設計部分において生じた発明を，特許法上，確実に保護するためには，明細書を記載する上でどのような点に留意すればよいだろうか。平成18年度のソフトウエア委員会第1部会では，オブジェクト指向技術及びシステムの表現方法であるUMLに着目し，単純な事例に基づき考慮すべき問題点について検討した。

* 執筆担当者

2．オブジェクト指向とは？

2.1　従来の手法と何がどう違うのか？

オブジェクト指向技術を用いるシステム設計では，その設計段階において，従来の処理フロー設計を行う代わりに，オブジェクトに着目して設計する。オブジェクト指向設計は，従来の処理フロー設計から派生した構造化プログラミング，データ中心アプローチ（DOA）の延長として位置付けられている。

一般にオブジェクト指向とは，データ中心アプローチのようにデータのみに着目するのではなく，データとデータに対する操作との双方に着目する概念である。オブジェクト指向では，データとデータに対する操作とを一体化し，内部の処理自体は外部から隠蔽することでカプセル化しており，オブジェクトをメッセージ交換により起動させることにより処理を実行する。

例えば簡単な例で比較してみる。学生の出席管理アプリケーションを設計する場合，従来の処理フロー設計では，対象となる学生の出席データを読み込んで更新する，といった処理を順次設計する。それに対して，オブジェクト指向設計では，管理対象である学生そのものを中心にとらえ，「学生」という上位概念に，学生番号，氏名等の識別情報を属性として付与し，出席回数を取り出す，出席回数をインクリメントする等の操作（メソッド）を定義することにより，システム設計を行う。

すなわち，「学生」という概念には，出席回数というデータが隠蔽されており，外部とのインタフェースとして，出席回数を取り出す，出席回数をインクリメントする等の操作（メソッド）が定義されている。設計者は，内部でどのような処理が行われているかは知る必要がなく，メッセージの授受により操作（メソッド）に基づく処理，例えば「学生A」について「出席回数」を取り出す，等の処理を実行する。

この「学生」という上位概念を「クラス」と呼び，「学生」一般に抽象化された情報として定義しておく。それぞれの学生を識別する情報，例えば氏名を属性として付与することにより実行可能なオブジェクトとなる。混乱を避けるために，以下実行段階のオブジェクトをインスタンスと呼ぶ。

2.2　オブジェクト指向プログラム実行時のメモリ領域

上述したように，インスタンスは属性が付与された段階で生成され，それまではクラスとして定義情報が記憶されているにすぎない。換言すれば，クラスの状態では処理を実行することができない。あえて例えるならば，クラスがコンピュータで，属性情報がワープロソフトであると考えればわかりやすいだろう。つまり，コンピュータはソフトウエアをインストールしない限り何の処理も実行することができない，ただの箱である。そして，ワープロソフトをインストールした段階で，ワードプロセッサとしての機能を果たすことができるのである。この例では，ワープロソフトの実行が終了してもワープロソフトはインストールされた状態のままである。しかし，オブジェクト指向プログラムの実行時には，例えるならば実行終了時にワープロソフトがアンインストールされ，ただの箱に戻ってしまう。すなわち，実行時以外は実行モジュールが存在しない状態となる点で，ワープロソフトの例とは相違しているのである。

実際には，アプリケーションの実行段階では，クラスの定義情報がメモリへロードされている。クラスの定義情報にはメソッドの定義も含まれている。クラスの定義情報がロードされているのはメモリ上の静的領域であり，1つのクラスに対して1つの定義情報がロードされている。メモリロードのタイミングは特に限定されるものではなく，アプリケーションの起動時に一括ロードしても良いし，コマンドが実行される都度，逐次ロードしても良い。

「インスタンスの生成」と呼ばれる処理は，具体的には，生成対象となるインスタンスが実行時に用いるメモリ領域の割当処理となる。つまり，属性情報であるインスタンス変数を付与された場合，インスタンス変数は，生成するインスタンスが使用すべきクラスの定義情報が記憶されているメモリの静的領域の始点アドレスを示すポインタとして機能する。したがって，インスタンス変数が付与されることにより，対応付けられたクラスの定義情報に属性情報が付与されたインスタンスが，メモリの動的領域に生成される。このようにすることで，クラスの定義情報の一部であるメソッドに基づく処理をインスタンス単位で実行することができる。なお，実行終了時には，該インスタンスは動的領域から削除される。

このように，オブジェクト指向プログラム実行時には，インスタンスが生成されている間だけメモリ上に該インスタンスが存在しており，メモリダンプ等を取得することにより存在を確認することができる。一方，

インスタンスが生成されていない間は，メモリダンプ等を取得した場合であっても，該処理の存在を確認することができない。従来の手続き型言語で作成されたプログラム（コンパイラ型）では，ロードモジュールが常時メモリ上に存在しているのと比べて，大きく相違している。

2.3　オブジェクト指向プログラムの表記方法

オブジェクト指向プログラムの考え方は，従来のフローチャートで記載することができるのであろうか？答えは「Yes」でもあり，「No」でもある。しかし，フローチャートで記載する場合には，考え方自体を従来の処理フローへと変換する必要があり，特に発明者にとっては現実的な記載方法ではない。そこで，直接的にオブジェクト指向プログラムの考え方を表記するために標準的に定義付けられた表記方法が，1990年代の中ごろから普及し始めたUML（Unified Model Language）である。UMLには9種類のダイアグラム（図）が存在するが，ここでは，以下の事例の説明等でも用いる代表的なダイアグラムにつき，簡単に説明する。

(1) クラス図

クラスに含まれる属性，操作を明確にし，クラス間の関係を表す図である。FIG. 1は，上述した「学生」クラスを示すクラス図である。

FIG. 1に示すように，クラス名「学生」に対して，属性「回数」及び変数（メソッド）が2つ定義されている。インスタンスとして機能させる場合には，「学生」の名前等の識別情報をインスタンス変数として付与して用いる。クラス間の関係は，クラス図同士を結ぶ線の種類で区別する。

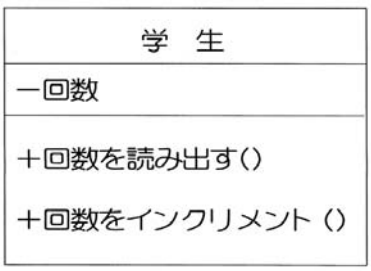


FIG. 1

(2) シーケンス図

時間軸に注目して，クラスにインスタンス変数をどのタイミングで付与するか，すなわち，どのタイミングでインスタンスが生成されているのか等を示す図である。FIG. 2では，インスタンスA，B，Cそれぞれについて，どのタイミングで生成されているかを四角形で表している。インスタンス相互間には，付与すべきメッセージ，あるいは呼び出す変数（メソッド）を明示することによりインスタンス相互間の関係も容易に把握することができる。

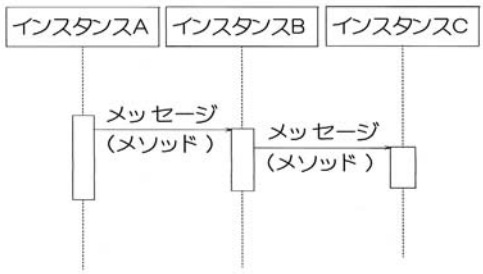


FIG. 2

(3) アクティビティ図

インスタンスのアクション（動作）の変化を示す図である。フローチャートと類似しているので，従来型のシステム開発に慣れている開発者にも理解しやすい。FIG. 3に示すように，条件分岐も含めて，どのようなアクション（動作）がどの順番で実行されるのかが明示される。

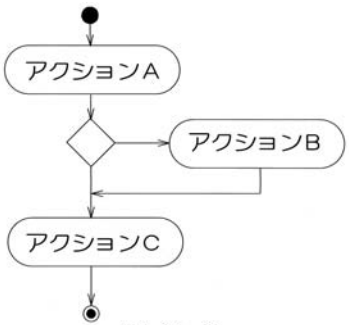


FIG. 3

3．事例の紹介

3.1　事例説明

UMLを用いた参考事例として，ネットワーク上で商品の販売などを行うバーチャルショップ（サイバーショップ等ともいう）を構成するシステムを用いる。以下，本事例で用いるシステムの概要について説明する。

本システムは，複数のクライアント端末，カタログサーバ，および複数のショップサーバがネットワークに接続されて構築されている。ネットワークは，公衆回線網，インターネット等の既存のネットワークであ

る。

複数のショップサーバのホームページ（ウェブサイト）によってバーチャルショップが形成される。複数のショップは，互いに同じ商品または同じ商品群を取り扱っている。もっとも，一のショップと他のショップとが，互いに異なる商品群を取り扱っていても良い。

カタログサーバは，複数のショップが取り扱うすべての商品に関するデータベースを備えている。カタログサーバはまた，商品名（検索条件）が与えられたときに，その商品名の商品，その商品名に属する個別商品（以下，これらを関連商品という）等に関するデータをデータベースにおいて検索して出力する機能（プログラム）を備えている。

カタログサーバに設けられたデータベース（商品データベース）は，商品の一般名称（例えば，ウイスキー，ワイン，ビール，日本酒など），複数の商品のグループ名称（例えば，アルコール），個々の商品名（例えば，ABC ビール 720ml，DEF ビール 350ml など）等のように階層化されたデータ構造となっている。データベースはまた，個々の商品について，その商品の価格，その商品を取り扱っているショップ名，その商品の在庫情報，その他のデータを格納している。

カタログサーバのメモリには，ショップ名／アドレス対応表のテーブルが設けられている。この対応表にはショップ（名）ごとに，そのショップのホームページ（ウェブサイト）の URL が記述されている。

次に，ユーザがクライアント端末からカタログサーバにアクセスして所望の商品を取り扱っているショップを知り，そのショップに商品を注文する動作について，FIG. 4 を参照して説明する。

クライアント端末において，ユーザはカタログサーバの URL を入力する。クライアント端末はカタログサーバにアクセスする（ステップ S1）。カタログサーバは商品検索欄（フィールド，ボックス）を表すデータ（HTML ファイル）をクライアント端末に送信する（ステップ S11）。クライアント端末は，FIG. 5 に示す商品検索画面を表示する（ステップ S2）。

ユーザは商品検索画面において希望する商品名を入力する。FIG. 5 に示す例では，「DEF ビール 350ml」が入力されている。ユーザが［検索］ボタンをクリックすると，商品名を表すデータを含む検索要求がクライアント端末からカタログサーバに送信される。

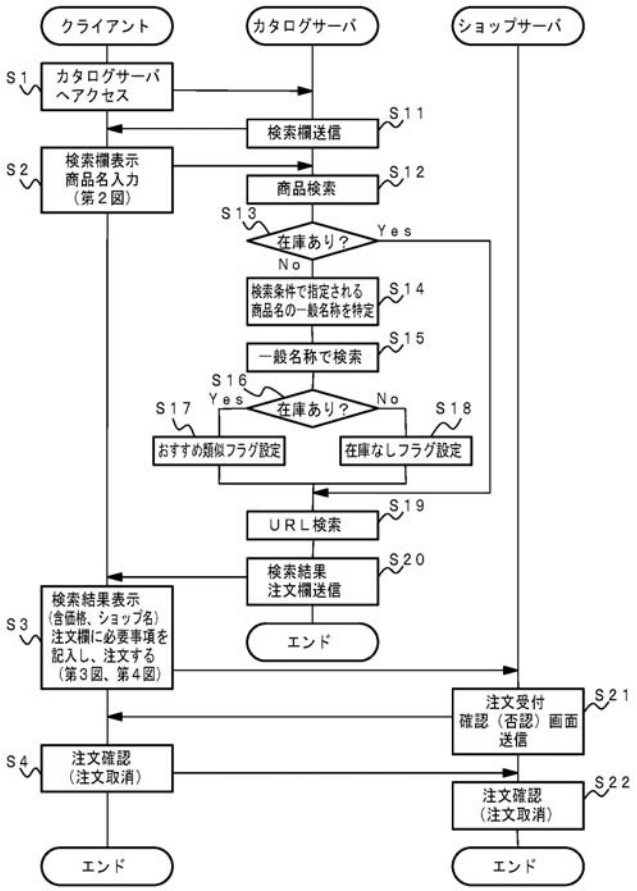


FIG. 4

カタログサーバはクライアント端末からの検索要求を受信すると，検索要求に含まれる商品名をキーにしてデータベースを検索する（ステップ S12）。そして，カタログサーバは検索対象の商品の在庫があるかどうかを判定する（ステップ S13）。

FIG. 5

検索対象の商品在庫が存在する場合は，カタログサーバは，検索した商品名およびそれに関するデータ（商品の価格，取り扱いショップ名）を得る。そして，URL 検索の処理（ステップ S19）に移行する。

検索対象の商品の在庫が存在しない場合は，カタログサーバは，検索対象の商品に関連する商品（関連商品）を検索する処理を実行する。関連商品を検索する処理を類似商品検索という。類似商品検索において，

カタログサーバは，検索条件で指定された商品名の一般名称を特定し（ステップS14），特定した一般名称をキーとしてデータベースを検索する（ステップS15）。例えば，検索対象の商品名が「DEFビール350ml」である場合，「DEFビール350ml」の一般名称は「ビール」である。したがって，この場合，カタログサーバは，「DEFビール350ml」の一般名称「ビール」をキーとしてデータベースを検索する。そして，カタログサーバは類似商品検索対象の関連商品の在庫があるかどうかを判定する（ステップS16）。

関連商品の在庫が存在する場合，カタログサーバは，検索した関連商品の商品名およびそれに関するデータ（商品の価格，取り扱いショップ名）を得る。また，類似商品検索で関連商品が検索されたことを示すフラグ（おすすめ類似フラグ）をメモリに設定する（ステップS17）。そして，URL検索の処理（ステップS19）に移行する。

関連商品の在庫が存在しない場合は，カタログサーバは，類似商品検索で関連商品についても検索できなかったことを示すフラグ（在庫なしフラグ）をメモリに設定する（ステップS18）。そして，URL検索の処理（ステップS19）に移行する。

カタログサーバは，ユーザから検索要求された商品またはそれに関連する関連商品を検索できた場合は，ショップ名／アドレス対応表にもとづいて，検索した商品を取り扱っているショップのURLを検索する（ステップS19）。そして，カタログサーバは，検索により得られた商品名等のデータ，取り扱いショップのURL，および注文欄（フィールド，ボックス）（送信ボタン等を含む）を表すHTMLファイルを作成してクライアント端末に送信する（ステップS20）。なお，類似商品検索で関連商品についても検索できなかった場合は，ユーザの求めている商品およびその関連商品を検索できなかったことを表すHTMLファイルを作成してクライアント端末に送信する。

クライアント端末はカタログサーバから送信されたHTMLファイルに基づいて，FIG. 6またはFIG. 7に示すような，検索結果（商品（関連商品）一覧表）および注文欄を表示する（ステップS3）。FIG. 6はユーザから検索要求された商品を検索できた場合の画面表示例を示し，FIG. 7はユーザから検索要求された商品を検索できなかったが関連商品を検索できた場合の画面表示例を示している。

FIG. 6

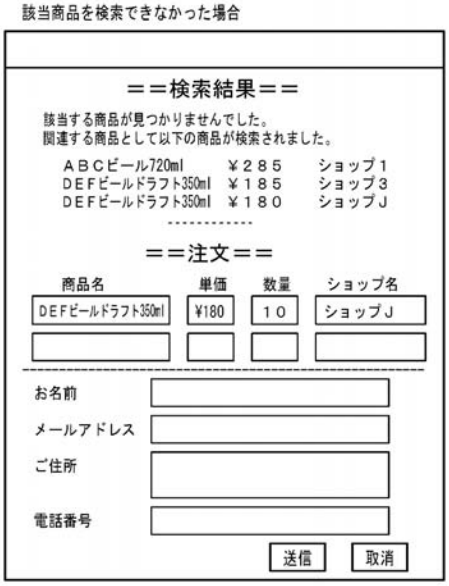


FIG. 7

ユーザはこの画面を見て，購入を希望する商品名と数量（個数）を入力する。単価およびショップ名は自動的に表示される。ユーザはまた，自分の氏名，メールアドレス，住所，電話番号を入力する。入力されたデータは画面に表示されるので，ユーザはこれを確認して「送信」ボタンをクリックする（ステップS3）。

「送信」ボタンのクリックに応答してクライアント端末は，ユーザによって入力された注文欄のデータを，注文された商品を取り扱っているショップのショップサーバに送信する（ステップS3）。なお，このショップのURLは上述のようにカタログサーバからのHTMLファイル内に記述されている。

クライアント端末からの注文データを受信したショ

ップサーバは，注文確認用の画面を生成するHTMLファイルを作成して該当するクライアント端末に送信する（ステップS21）。この確認画面は先にユーザが入力した注文データを含む。

クライアント端末に表示される注文確認画面を見て，ユーザが，その画面に含まれている「確認」ボタンをクリックすると（ステップS4），その旨がショップサーバに送信され，注文（取引）が成立する（ステップS22）。ユーザは注文確認画面が表示されたときに，その画面に含まれる「取消」ボタンをクリックすることにより注文を取り消すこともできる。

上記の構成によれば，ユーザが商品を取り扱っているバーチャルショップにアクセスして注文することができ，商品の購入が容易となる。

3.2　受任から明細書作成までの検討過程

(1) 受任時の図面

上述の電子商取引システムをオブジェクト指向プログラミングで実現した発明の資料として，依頼主から最初に配置図とクラス図が提示された。以下これらの図面について説明する。

(1－1) 配置図

FIG. 8に配置図を示す。配置図は一般には，システム実行時に稼動している各モジュールの物理的な配置を表現するために使用されるものであり，各ハードウエア上でどのようなコンポーネント（ソースファイルやヘルプファイル等）を配置するかを定義する。FIG. 8より，本システムのシステム構成が把握できる。同図に示すように，本システムは，カタログサーバ，クライアント（クライアント端末）及びショップサーバで構成される。カタログサーバにおいて，商品データベースとショップ名／アドレス対応表が配置される。クライアント（クライアント端末）はカタログサーバ及びショップサーバそれぞれに接続される。

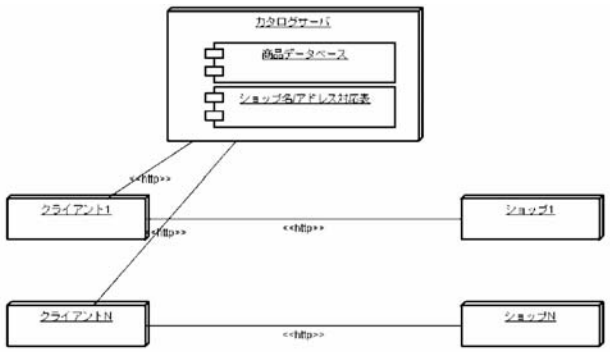


FIG. 8　配置図

(1－2) クラス図

FIG. 9に，本システムをオブジェクト指向プログラミングで実現する場合のクラス図を示す。前述のようにクラス図からは，各クラスの属性，操作の内容，及びクラス間の関係のようなクラスの静的な性質を把握することができる。FIG. 9において，カタログサーバに関するクラスとして商品クラス及びショップクラスが，ショップサーバに関するクラスとして注文クラスが，クライアントに関してクライアントクラスがそれぞれ定義されている。

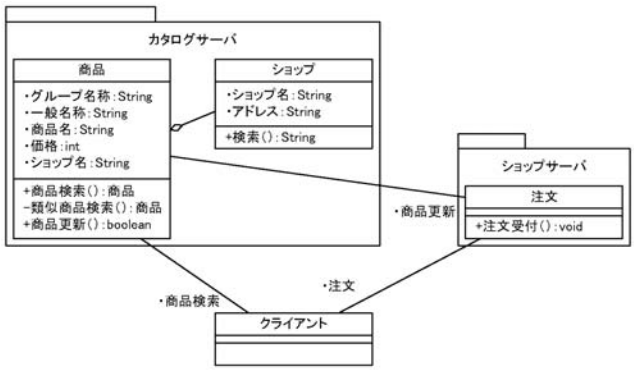


FIG. 9　クラス図

(1－2－1) カタログサーバ

カタログサーバに関して商品クラス及びショップクラスが定義されている。

a. 商品クラス

商品クラスの属性として，グループ名称，一般名称，商品名，価格，ショップ名が定義されている。また，商品クラスに対する操作として，商品検索，類似商品検索，商品更新が定義されている。商品検索は，クライアントからの検索条件に基づき，商品データベースから商品を検索するメソッドである。類似商品検索は，検索条件に一致する商品が商品データベースから検索されなかったときに，検索条件で指定される商品に類似する商品を検索するメソッドである。商品更新は，ショップサーバからの情報に基づいて商品データベースを更新するメソッドである。

b. ショップクラス

ショップクラスの属性として，ショップ名，アドレス（ホームページのURL）が定義されている。ショップクラスの操作として，検索が定義されている。ショップクラスは，ショップ名から，そのショップのアドレスを検索し，そのアドレスを戻り値として返す。なお，ショップクラスと商品クラスとは集約の関係に

ある。

（1－2－2）ショップサーバ

ショップサーバに関して注文クラスが定義されている。注文クラスの操作として，クライアントからの商品の注文を受け付ける注文受付が定義されている。

（1－2－3）クライアント

クライアントに関してクライアントクラスが定義される。クラインアントクラスには，図示していないが，カタログサーバに対して商品検索を要求する操作や，ショップサーバに対して商品の注文を実行する操作が定義されている。

（1－3）問題点

FIG. 8の配置図からは本システムのシステム構成が理解でき，また，FIG. 9のクラス図からは各クラスの属性，操作，クラス間の関係等，クラスの静的な性質が理解できる。しかしながら，これらの図のみでは，商品データベースがどのように構成されるのか，また，カタログサーバにおいて商品検索，類似商品検索等がどのように処理されるのか，ショップサーバにおいてどのように注文を受け付けるのか等，具体的なオブジェクトの処理内容までを把握することはできない。そこで，これらの情報が把握できる更なる図面が必要であると考えられる。

（2）追加時の図面

上で述べたように受け付け時に得られた図面では十分な明細書を作成することはできない。従って特許法第36条の実施可能要件及び開示要件，また特許法第29条第1項柱書の要件に違反しない明細書を作成するために必要十分な図面の追加を行う必要がある。

UMLでは上記クラス図，配置図の他にオブジェクト図（オブジェクトとオブジェクト間の関係をモデル化するための図。クラス図を実体化したもの），コンポーネント図（ソフトウエアモジュールの構成を表現する），ユースケース図（対象システムが提供するサービス・機能を表現する振る舞い図），シーケンス図（オブジェクト間のやりとりを時系列で表現する），コラボレーション図（オブジェクト間のやりとりを構造的に表現する），ステートチャート図（オブジェクトの内部状態の変化を表現する），アクティビティ図（業務の手順やワークフローを表現する）の7つのダイアグラムが定義されている。

従来のコンピュータ・ソフトウエア関連発明の明細書では，システムの構成を示すブロック図，データの構造を示す図（レコードやテーブル）と処理の流れを示すフローチャートを添付するのが一般的であった。これに対しUMLで実施の形態を説明する場合には，上記7つのダイアグラムの特性を考慮した上で，以下のような図を追加すべきと考える。

（2－1）追加すべき図（テーブル）

FIG. 10に本例で用いるテーブル例を示す。

商品

グループ名称	一般名称	商品名	価格	ショップ名	在庫フラグ
アルコール	ビール	ABCビール720ml	100	SH1	○
アルコール	ビール	ABCビール720ml	105	SH2	×
アルコール	ビール	ABCビールドラフト	120	SH2	○
アルコール	ワイン	メドック赤720ml	1680	SH3	○

ショップ

ショップ名	アドレス
SH1	http://www.sh1.jp/
SH2	http://www.sh2.com/
SH3	http://www.sh3.co.jp/

FIG. 10

データテーブル自体はUMLにおける規定はない。しかしコンピュータ・ソフトウエア関連発明の明細書を書くにあたっては，どのようなデータがあって，それをプログラムがどのように処理しているかを記載することは処理内容を理解する上または動作内容を説明する上で重要である点はUMLでも従来型でも同様である。

さらに特許法第29条第1項柱書において要求される「発明」であることの要件を満たすためには，ソフトウエアによる情報処理がハードウエア資源を用いて具体的に実現されていることが必要となる。このためには，ソフトウエアとハードウエアが協働した具体的手段によって，使用目的に応じた情報の演算または加工を実現し，使用目的に応じた特有の情報処理装置またはその動作方法が構築されることを表さなければならない。

拒絶理由通知において，特許法第29条第1項柱書違反が指摘された場合には，従来型であれば，明細書に記載されたフローチャートやデータテーブル等に基づいて，記憶部に記憶されたデータがどのようにプログラムによって処理されるかということを補正書ないしは意見書において明らかにするのが一般的であった。

従って29条1項柱書違反対策という意味からも，UMLで明細書の実施の形態を記載する場合にも，UMLには規定されていないがデータテーブルを追加

して，当該データテーブルを用いた具体的処理を説明しておくことが好ましい。

なお，FIG. 10 の商品テーブル及びショップテーブルは，ソフトウエアによる具体的な情報処理を説明する際に用いられ，これらのテーブルの各カラムについても，同様に情報処理の説明において具体的に用いられるべき内容を含む。

（2－2）追加すべき図（シーケンス図）

FIG. 11 のようなシーケンス図は，オブジェクトとオブジェクトとの相互作用を時系列に沿って表すことができる図である。具体的には，シーケンス図の縦方向は，時間の経過を表し，横方向は，オブジェクト相互間の関係を表している。

クラス図などには，オブジェクト相互間の関わりについて具体的には記載することができないため，外部に対するデータの送受信を行う構成を「～手段」と特許請求の範囲に記載した場合等においては，シーケンス図は，明細書におけるサポートのため，加えた方が良い図面の１つであると考えられる。

但し，シーケンス図ではオブジェクト間のメッセージのやりとりは明らかになるが，各オブジェクト内における処理は具体的に示されるわけではないので，シーケンス図の一部について従来からのフローチャート（例えば以下に示す FIG. 14）を用いて補うようにすべきである。

シーケンス図では，オブジェクト間でやりとりされるメッセージを表す矢印は，上から順に時系列に並んでおり，FIG. 11 では，①クライアントからカタログサーバの商品オブジェクトへの商品検索メッセージ，②商品オブジェクトからショップオブジェクトへの検索メッセージ，③クライアントからショップサーバへの注文メッセージ，の順に出力されることが分かる。

なお，FIG. 11 では，メッセージを送信した後のリターンを点線矢印で記載しているが，煩雑さを避けるため記載しないこととしても良い。その場合は，アクティブな時間が終了した時点（すなわち，長方形の下端）でリターンされていることを表す。

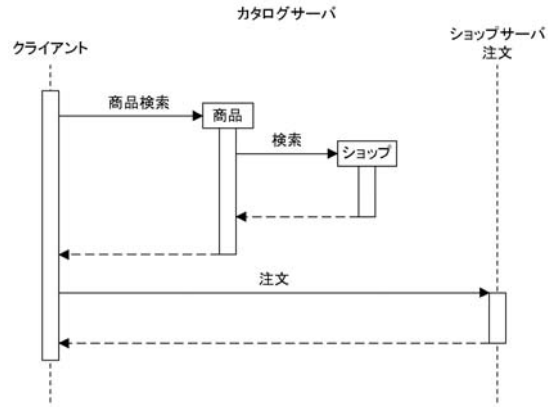


FIG. 11

（2－3）追加すべき図（アクティビティ図）

アクティビティ図には，手続き的な振る舞いが記述される。フローチャートと類似しているが，フローチャートに比べてオブジェクト間の処理の相互作用が記述できるという利点がある。但し，単一オブジェクトならばフローチャートとほぼ同様の内容となる。

FIG. 12 に，ショップサーバ部分を省略してクライアント端末とカタログサーバとについてのアクティビティ図を示す。この FIG. 12 では，誰が何を実行するかを明示するため，クライアントの区画と，カタログサーバの区画に分けている。さらにカタログサーバの区画は，商品オブジェクトの区画と，ショップオブジェクトの区画に分けて，FIG. 9 のクラス図に対応付けている。このように，図面間の抽象度のレベル合わせはある程度必要である。

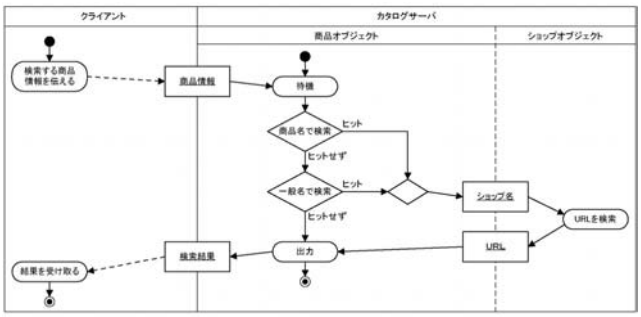


FIG. 12

FIG. 12 のようなアクティビティ図であれば，FIG. 10 に示したデータテーブルとの関連で各アクティビティにおいてどのような情報処理が行われているかを具体的に説明できるため，ハードウエアとソフトウエアの具体的な協働を示しやすくなっている。また，FIG. 12 の例では，オブジェクト間でやりとりされるメッセージに含まれるデータについても「商品情報」

「ショップ名」「URL」「検索結果」といった形で示しているため，シーケンス図を用いずとも具体的なオブジェクト間の協働を示すことができる。

クラス図と同様にアクティビティ図についても抽象度の高低がある。通常発明のポイントを説明するのにふさわしい抽象度を選択し，クラス図やアクティビティ図などにおいて同様の抽象度をもって説明すべきである。但し，FIG. 10に示したデータテーブルのような図を導入する場合には，データテーブルを用いた具体的な処理を説明できるような抽象度が望まれるため，発明のポイントにのみ着目するのでは不十分な場合もあると考えられる。

FIG. 13に示すように，若干抽象度を高めた形で，クライアント，カタログサーバ及びショップサーバについてのアクティビティ図を作成することもできる。FIG. 13のようなアクティビティ図でもカタログサーバの処理はおおよそ説明することができるが，FIG. 13だけでは類似商品検索の具体的な内容は説明できない。従って，このような場合には類似商品検索の部分を他の図でフォローする必要がある。

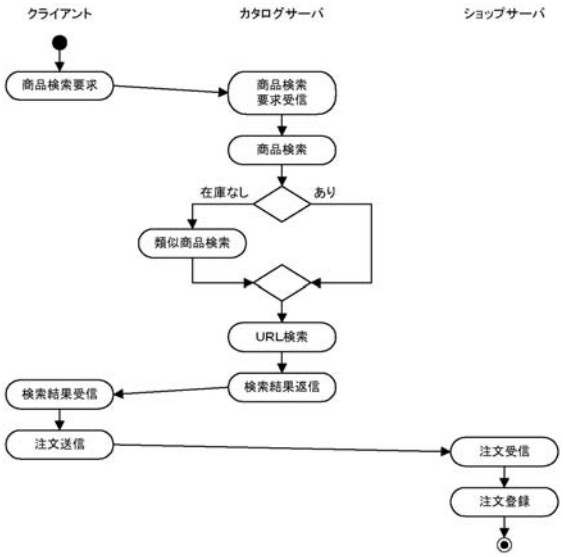


FIG. 13

(2−4) フローチャート

フローチャートは，手続き的な振る舞いが記述されるという点でアクティビティ図に似た役割を果たす。条件付きの振る舞いが「分岐」で示される点もアクティビティ図と類似する。フローチャートとアクティビティ図との違いは，フローチャートが並列の振る舞いをサポートしていないことや，フローチャートではオブジェクト間の処理の相互作用を記述しにくいことにある。

上でも述べたようにシーケンス図の一部分の詳細化や，抽象度の高いアクティビティ図の一部アクティビティの詳細化のために，FIG. 14に示すようなフローチャートを導入して対処することが必要となる場面もあると考えられる。

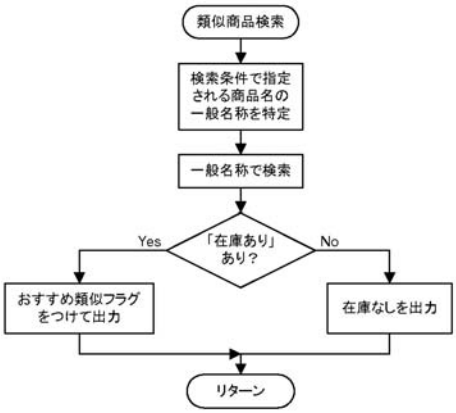


FIG. 14

(3) ソースコードの検討

ここで，上記UMLで示す本事例のシステムが，実際上，どのような形で実装されるのかについて，その一例をソースコードを用いて説明する。

なお，本例に示すソースコードでは，各クラスの概要のみを示し，詳細な処理等に対応するコードは省略している。

(3−1) ソースコード作成の前提

まず，今回の実装例の前提として，ソースコード作成者は，上記UML図の内，クラス図，テーブル及びアクティビティ図を渡している。実際上は，クラス図を参照しつつ，テーブル及びアクティビティ図に基づきソースコードを作成している。

(3−2) カタログサーバにおけるクラス構成概要

FIG. 15に示すように，本実装例では，カタログサーバを商品情報クラス（class Product），店情報クラス（class Shop）及びデータベース関連の操作を行うクラス（class DBManager以下，「DBマネージャークラス」という。）の3つで構成しており，上記クラス図に示すカタログサーバとは，若干異なっている。

```
/**
   * 商品情報クラス
   */
class Product {

}


/**
   * 店情報クラス
   */
class Shop {

}


/**
  *データベース関連の操作を行うクラス
   */
class DBManager {

   }
```

FIG. 15

(3－3) DB マネージャークラス (class DBManager) の構成概要

本実装例に示す DB マネージャークラスは，上記クラス図に示す商品クラスが行う商品検索メソッド及び類似商品検索メソッドを実行するものとなっている。具体的には，例えば，FIG. 16 のように実装される。

(3－4) 問題点

以上のように，UML 図を作成する上流工程の担当者と，実装を行う下流工程の担当者とが異なる場合には，本実装例に示すように，クラス図に示すものとは異なるクラス構成となる可能性がある。

例えば，今回の実装例については，商品クラス，ショップクラスが行うメソッドの内，個々の「商品」について行う「商品更新」は商品情報クラスのメソッドとして実装しているが，「商品群」について行う「商品検索」「類似商品検索」や，「店舗群」について行う「検索」などのデータベース操作関連のメソッドにつ

```
/**
    *データベース関連の操作を行うクラス
    */
   class DBManager {
      }
     // データベースへの問い合わせ結果から、商品オブジェクトを生成
      private Product createProduct(ResultSet rs) throws SQLException{
      }

     // データベースへの問い合わせ結果から、店オブジェクトを生成
      private Shop createShop(ResultSet rs) throws SQLException {
      }

      // 店 ID から店オブジェクトを生成
      private Shop searchShop(int shop_id) throws SQLException {
      }

      // 商品名から商品を検索
      public Product[] search(String name) throws SQLException {

         // 検索結果の全ての商品について、店情報を検索
         while(rs.next()){

            }
            l.add(p);
         }

         // 店情報の配列を返す
         Product[] products = (Product[])l.toArray(new Product[0]);
         return products;
      }

      // 検索キーワードの商品の一般名を調べ、その一般名を持つ代替商品を表示
      public Product[] generalSearch(String name) throws SQLException {
         // 指定された商品の一般名を検索

         List l = new ArrayList();
         Map shops = new HashMap();

         // 検索結果の全ての一般名について、商品を実際に検索
         while(rs.next()){

            // 検索された全ての商品について、店情報を検索
            while(rs2.next()){

               }
               l.add(p);
            }
         }

         Product[] products = (Product[])l.toArray(new Product[0]);
         return products;
      }

      // 商品名から商品を検索
      public Product getProduct(int product_id) throws SQLException {

         }
         throw new SQLException("Can't find the product");
      }
   }
```

FIG. 16

いては，DBマネージャークラスのメソッドとして実装している。

今回，実装を担当していただいた方の話によると，データベース操作関連のメソッドなどについては，それらをまとめてDBマネージャークラスとして実装するのが一般的とされる場合もあるとのことであった。（但し，そのまとめ方については，様々な実装の仕方が可能とのことである。）

従って，実装例を考えずに，単に上流工程で作成されたクラス図等を鵜呑みにしてクラス構成を考えて明細書を作成した場合，実際の実装と詳細な説明の構成とが異なってしまうケースも出てくるようである。

4．考察

(1) 発明の詳細な説明の一般的な問題

(1－1) 実施可能要件

発明の詳細な説明の記載は，特許・実用新案審査基準第Ⅶ部第1章コンピュータ･ソフトウエア関連発明において「発明の詳細な説明は，ソフトウエア関連発明の分野における通常の技術的手段を用い，通常の創作能力を発揮できる者が，特許請求の範囲以外の明細書及び図面に記載した事項と出願時の技術常識とに基づき，請求項に係る発明を実施することができる程度に明確かつ十分に記載しなければならない」(1.2.1 実施可能要件）とされており，また「課題を解決するための手段では，手順又は機能がどのように具体化されたかをフローチャートなどを用いて説明する」(1.2.2（1）発明が解決しようとする課題及びその解決手段）必要があることから，詳細な説明においてもフローチャートなどを用いて説明することが通常であると思われる。しかし，明細書中の実施例，実施形態をUMLを用いて説明すること自体は基本的に可能であり，オブジェクト指向言語で開発されたシステムに基づく出願の事案を見るとフローチャートを用いずに実施例を表現して特許されたものもあり，特許庁においても詳細な説明でUMLを使用することは記載不備（特許法第36条第4項第1号違反）にはならないと考えているものと思われる。

例えば，特許第3687587号の明細書の記載は，ほとんどがプログラムの動作についてであり，メソッド，オブジェクト，クラス，インスタンスといった用語が用いられ，また，説明図面についてもフローチャートは一切なく，システム構成図とシーケンス図がほとんどである。しかし，単にオブジェクト指向言語は公知であるとすることなく，必要な範囲でメソッドやオブジェクトについて説明しており，一度は拒絶理由に挙げられたが最終的には記載不備（36条第4項）も解消している。このように，UMLを用いて詳細な説明を記載する場合は，従来の記載との橋渡しをする一定の説明が必要であるものの，必ずしもフローチャートなど従来の記載方法を踏襲しなくても記載不備を生じさせないようにすることができると考えられる。

したがって，UMLのみを用いても発明の内容が明確に表現されて当業者が理解でき，実施可能要件を満たす場合は，審査上問題はないものと考えられる。

(1－2) 請求項のサポートおよび中間処理への対応

言うまでもなく，請求項に記載される発明は，発明の詳細な説明に記載されていなければならず，また中間処理において請求項を補正する場合は明細書に記載した事項の範囲内でしなければならないから，請求項に記載する，あるいは今後記載する可能性のある事項は発明の詳細な説明に記載しておくよう十分注意しなければならない。特にコンピュータ・ソフトウエア関連発明では，UMLを用いるか否かにかかわらず，「ソフトウエアによる情報処理が，ハードウエア資源を用いて具体的に実現されている」ことが理解できるように請求項を記載しない場合，請求項に記載された発明は法上の発明として成立していないとされる可能性がある。

したがって，請求項の記載のサポートのため，あるいは中間処理において補正を担保するため，出願当初から明細書中にソフトウエアおよびハードウエア資源について具体的に十分な記載がなされている必要がある。特に発明の成立性要件については，出願当初より明細書にサポートされていなければ，特許請求の範囲に成立性要件を満たす記載があっても，あるいは出願後，成立性を満たす補正を行おうとしても結局，特許法第29条第1項柱書の規定を満たす請求項とすることはできないため，発明の成立性の要件を満たす特許請求の範囲の記載とすることができるような明細書について検討する。

発明成立性については，従来の記載方法によると，フローチャートとともに機能ブロック図やデータテーブルなどを用いることにより，構成要件であるモジュールの間の関係やデータの具体的処理内容を明確に表現することができ，これによりハードウエア資源を用

いた具体的な記載が可能であり，適法な特許請求の範囲の記載を担保することができる。しかし，UMLを用いた場合，機能ブロック図に代わる表現方法としていずれが適切かを検討する必要がある。1つの具体的方法としては，クラス図と他の図面などを組み合わせることも有効と考えられるが，この点については後述の「クラス図について」における検討に譲る。

オブジェクト指向言語による開発は従来技術であり結果的にソフトウエアとハードウエアが協働して処理がなされるのは明らかであるから，一般にオブジェクト指向言語により開発されたソフトウエアも，「ソフトウエアによる情報処理が，ハードウエア資源を用いて」行われること自体は自明であり保護の対象足りえると考えられ,実際そのように主張する出願人もいる。しかし，かといって単にオブジェクト指向プログラムの要素を並べ立てただけでは，具体的な動作が不明瞭になることもあると考えられる。すなわち，「オブジェクト指向ソフトウエアにおいては，オブジェクトのクラスとインスタンスとメソッドを特定することが肝心であり，オブジェクトのクラスとインスタンスとメソッド（必要に応じて継承）を特定すれば，オブジェクトを生成すること自体はたとえばC言語によって当業者にとって任意にプログラムが可能」であると主張することも可能であるが，実務上これらの要素は従来式の表現における機能モジュールとは異なるのであるから，オブジェクトの具体的な機能，作用をハードウエア資源との関係を明確にして特定しなければ拒絶される可能性があると考えられる。

例えば，特願2000-72826は，特許法第29条第1項柱書違反として拒絶されているが，明細書の記載も元々オブジェクトの動作に対する記載が少ないため，「ソフトウエアによる情報処理が，ハードウエア資源を用いて具体的に実現されている」特許請求の範囲の記載とするような補正は困難であったと思われる。すなわち，本出願の拒絶審決（不服2003-15585）では「本願請求項1の“属性情報により分類したクラスと商品の流通経路からなる関連づけと商品の在庫数からなるインスタンスと商品の販売または発送のメソッドとを有する店舗オブジェクトと，要求してから商品が移送されるまでの時間あるいは商品配分の順序・数量をメソッドの制御パラメータとする待行列管理オブジェクトとを生成し，前記オブジェクト群により商品の在庫と販売事象をシミュレーションするシミュレーション手段”は，制御パラメータが通常物理的数値を表すことからハードウエアを用いているといえるが，その具体的内容がハードウエアとの関連で記載されておらず」，ハードウエアの動作を単に記載するだけでなく，発明の特徴部分をハードウエアとの関連で記載する必要があるとしている。しかし，本事例の場合そもそも明細書の記載が，このような特許請求の範囲とする補正を担保する，ハードウエア資源を具体的に記載した明細書とはなっていないため，補正のしようがなかったものと思われる。

これは，従来の特許請求の範囲の記載では例えば，ソフトウエアモジュールを手段にして構成要件とすることができたが，オブジェクト指向におけるオブジェクトやインスタンスは従来のソフトウエアモジュールと異なり，動的に生成され各要素の関係も常に変化するため，ハードウエアと関連させて記載することが容易ではないという点も一因であると考えられる。また，そもそも発明者自身もオブジェクト指向で開発している場合にはハードウエアを意識していない場合が多いということも一因であろう。

したがって，ハードウエア資源を用いて具体的に実現されている特許請求の範囲を担保するため，UMLを用いて発明を説明する際には，ハードウエアとソフトウエアとの協働関係を十分説明し，可能な限り具体的に処理内容を記載する必要がある。例えば，アクティビティ図をテーブルに関連付けすることによりハードウエアとソフトウエアとの協働関係を具体的に説明することができると考えられる。

（1－3）その他

特許第3687587号について上記では，フローチャートが明細書になくても記載不備とはならない例として説明したが，一方，この特許の審査過程を検討すると出願当初のオブジェクト指向技術に限定されていなかった特許請求の範囲が，最終的に「オブジェクト指向技術に基づいて処理を行うデータ処理システム」と補正されている。これは，審査官は条文こそ適用しなかったが，明細書にオブジェクト指向技術以外の記載がないことからオブジェクト指向技術以外に本願発明を適用するのはサポート要件違反である旨示唆しており，出願人はこの示唆に沿って限定したものである。確かに，本特許は本来オブジェクト指向技術以外のソフトウエア技術に適用することができない可能性が高いと考えられるが，オブジェクト指向技術以外に適用

することができる技術であったとすれば，その他の技術に適用する場合も考慮した記載を含めておくことにより，限定をしなくてもある程度の反論が可能だったものと考えられる。

このように，実施例自体がオブジェクト指向技術により達成されるものであっても，特にその発明の内容がその他の技術にも適用可能である場合は，その他の技術に適用できる旨記載するとともに，そのように適用する際の留意点やオブジェクト指向技術との相違点などを記載しておく必要があると考えられる。

(2) クラス図について

(2－1) クラス図におけるデータの処理内容の表現の限界

クラス図は，システムの構造を説明するために，クラスの仕様とクラス間の関係を表現するための図である。各クラスには，クラスの「名前」と「属性」と「操作」とが含まれ，クラス図では，クラス間の関係は実線で結ばれて示される。クラス図において，クラスのオブジェクトで何ができるかを示すのが，「操作」である。

「属性」をデータと捉え，「操作」を処理内容と捉えると，クラス図は，一見システムにおけるデータ処理の内容を示すものであるとも見えるが，クラス図だけでは，出願における図面として十分なものではないことに，注意すべきである。

上述した事例では，FIG. 9 において，カタログサーバにおける商品クラスは，「属性」として，グループ名称，一般名称，商品名，価格，ショップ名を示し，「操作」として，商品検索，類似商品検索，商品更新を示している。

「属性」を処理対象のデータと考えれば，各「属性」は，処理対象のデータの構造を説明しているということができるかもしれない。「操作」をデータ処理の内容と考えれば，各「操作」は，処理内容を説明しているということができるかもしれない。

しかし，クレーム中に，カタログサーバ，ショップサーバ及びクライアント端末間で各データの送受信についての構成要件として，例えば「送信手段」，「受信手段」等が，構成要件の一部として必要な場合，上述した事例のクラス図には，そのような「送信手段」あるいは「受信手段」に対応する「操作」は，存在していない。

それは，何故であろうか。クラス図は，クラスの仕様とクラス間の関係を表現するだけの図であるため，CPU が行う各機能の処理を示すものではないからである。

通常，我々がある発明をクレーミングするとき，ある処理部における処理結果を，他の処理部（ソフトウエアあるいはハードウエア）に送信したり，あるいは他の処理部において受信したりするという，いわゆるデータのやりとりを意識している。そのような意識は，その発明に係るデータの処理が観念的になり過ぎないようにしたり，審査基準に言うハードとソフトの協働を記述するために，重要である。すなわち，クラス図の背後にある，OS，ミドルウエア等のプラットフォームを意識して明細書とクレームの作成を行う必要がある。

また，米国出願においては，クレームの構成要件は，図面に示す必要があるため，「送信手段」等がクレームに含まれる場合は，その「送信手段」等に対応するものを図面に開示しなければならないという要請があることにも注意すべきである。

(2－2) クラス図を補完する図面の検討

日本においては，方法クレームでもステップ毎の実行主体の記載が要求される場合があることも考慮すると，クレームに対応する機能的なブロック図を別途描いておくことが有用である場合もあると考えられる。機能的なブロック図は，オブジェクト指向技術という限定を外して，オブジェクト指向による実現方法も含めてクレーム化する場合も考慮すれば，重要である。従って，明細書作成時には，クレームとの関係で，クラス図に加えて，クレームの構成要素に対応する内容を示す図面がさらに必要かを検討する必要がある。

一方，そのような機能ブロック図ではなく，クラス図以外のアクティビティ図等と併せて，上述したような「送信手段」等の構成を説明する，又はクラス図などにおいて上記プラットフォームの部分に相当するブロックを書き足して説明するようにしても良いのではないか，との意見もあった。従って，クラス図を補完する図面等が必要か否かについても検討する必要がある。

なお，「送信手段」等の機能は，プラットフォームとなる OS が実行する場合もあるので，プログラムクレームなどにおいてクレーム中に送信手段を安易に入れることには，注意が必要である。

(2－3) 設計時と実装時のクラス図の相違

上述したように，委員会の検討の中で，クラス図に基づいて，プログラムの作成を実際にしてもらったが，結果として作成されたプログラムは，クラスの構成が変更されて作成されていた。その変更は，実際のソフトウエアの作成を考慮して，当業者であれば通常行う変更であった。すなわち，設計時のクラス図に，当業者であれば，通常考える内容の変更が加えられていた。

この事実から言えることは，設計時のクラス図は，ソフトウエアの実装時のクラス図とは異なっている可能性がある，ということである。従って，クラスの構成が多少実装時には変化する場合もあることを考慮に入れてクレームなどを作成すべきである。

(3) オブジェクト指向プログラム用のクレーム作成について

(3－1) 問題点

従来型プログラムの場合には，当該プログラムをインストールすることにより，実行モジュールA，B，Cがハードディスクに存在する。したがって，プログラムクレームが，「コンピュータをA手段，B手段，C手段として機能させるためのプログラム」である場合，上記実行モジュールA，B，Cを静的にメモリへ展開することによって，A手段，B手段，C手段として機能させることができ，クレームの技術的範囲に含まれるものと考えるのが自然である。

これに対して，オブジェクト指向プログラムの場合には，インストール時には，クラス情報だけがメモリに展開されており，所定のタイミングで，当該クラスから動的にオブジェクトが生成されてメモリ上に展開されるという特徴を有する。このように，プログラムをインストールした状態におけるコンピュータは，あるタイミングではAモジュールを生成するXクラスを，また他のタイミングではBモジュールを生成するYクラスを，さらにあるタイミングではCモジュールを生成するZクラスを，それぞれ単独で備えたコンピュータに過ぎないと考えることもできる。

すなわち，従来型プログラムでは実行時にメモリへ展開されるモジュールはインストール時のモジュールそのものであるのに対して，オブジェクト指向プログラムにおいては実行時にメモリへ展開されるオブジェクトとインストール時のクラスとは，同じものではない。

したがって，このような特質を有するオブジェクト指向型の被擬侵害プログラム用のクレームを作成する必要があるかが問題となる[(1)]。

(3－2) 間接的な記載形式との関係

現行審査基準において，プログラムクレームとして，1) コンピュータに手順A，手順B，手順C，…を実行させるためのプログラム，2) コンピュータを手段A，手段B，手段C，…として機能させるためのプログラム，3) コンピュータに機能A，機能B，機能C，…を実現させるためのプログラムといった間接的な記載が認められている。動的に生成されるというオブジェクト指向プログラムも，上記のような「機能させるためのプログラム」や「実現させるためのプログラム」に文言上，該当する。このように，間接的な記載が認められていることから，上記の問題点はほとんど影響がなく，オブジェクト指向プログラムに合致させたクレームを作成する必要性には乏しいといえる。

なお，「～するためのプログラム」というクレームでは，被擬侵害プログラム提供者が，オブジェクト指向プログラムのような実行オブジェクトを動的に生成するプログラムは含まれないと反論する可能性もある。そこで，かかる反論に対抗できるようにするために，「“～するためのプログラム”とは，実行モジュールが動的に生成されるプログラムを含む」というような一文を挿入しておく方が好ましいといえよう。

(3－3) 侵害追求手法との関係

また，積極否認の規定（特104条の2）からも，オブジェクト指向プログラムに合致させたクレームを作成する必要性には乏しいといえる。

被擬侵害プログラムが技術的範囲に属することの立証の手法の一つとして，特許権者は，プログラムを動作させた場合の処理を特定し，特許発明と同じデータが作成されることをもって，当該処理を実行する構成がコンピュータ内に存在するであろうという手法が知られている。例えば，上記のカタログサーバについては，検索した商品が存在しない場合，類似商品検索を行うという処理がある。かかる処理が請求項に記載されている場合，被擬侵害プログラムを実行した時には，同じ処理が実行され，類似商品がユーザに報知されるのであれば，かかる処理を行う構成が存在するとの主張が可能である。

これに対して，被擬侵害プログラム提供者がこのような構成が存在しないことを主張するのであれば，自己のプログラムにおける構成を積極的に開示しなけれ

ばならない（特104条の2）。

オブジェクト指向プログラムであっても，侵害被擬プログラムの，上記クラスには，上記類似商品検索処理を行うメソッドが定義されているはずである。例えば，「検索キーワードの商品の一般名を調べ，その一般名を持つ代替商品を取得」，「検索結果の全ての一般名について，商品を検索」というメソッドである。したがって，現実には反論が困難である。よって，実務的には，あえてオブジェクト指向プログラムで実現された場合のクレームを作成しておく必要性は乏しいといえる。

5．おわりに

オブジェクト指向に係る発明であっても，コンピュータ・ソフトウエア関連発明として取り扱われる場合があるため，注意すべき事項は通常のソフトウエア関連発明と大きくは変わらない。しかし，オブジェクト指向が開発者にハードウエアをあまり意識させないようにするための概念であるにもかかわらず，特許法第2条第1項における発明の定義を受けて，コンピュータ・ソフトウエア審査基準ではハードウエアとソフトウエアとの協働を求め続けている。すなわち，実施可能性は満たしているのに，発明として認めてもらえず，有用な発明が有効に保護されない場合が生じうる。このように開発の現状と特許法の運用とが乖離した形となっており，その乖離を埋めるための作業が有効な権利取得において重要性を増している。一方で，製品開発におけるソフトウエアの比重が大きくなっている現代においては，有用なアイデアを保護するという観点から，何らかの法的な手当が必要になってきているとも考えられる。

最後に，本稿作成上様々な点においてご指導いただいた一橋大学助教授兼宗進先生，そして実際にプログラムを作成して頂いた東京大学大学院生高橋慧様に，感謝申し上げます。

注

（1）上記カタログサーバのような発明では，従来型プログラムでもオブジェクト指向プログラムでもどちらでも実現可能である。このような場合に，被擬侵害プログラムがオブジェクト指向で構成されている場合の問題点についての考察である。

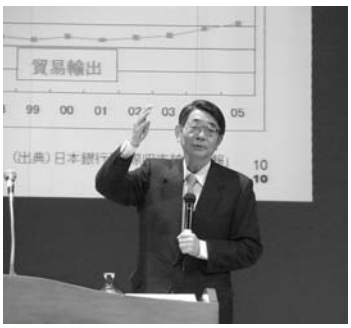
（原稿受領 2007.9.2）

長野県「知的財産支援フォーラム」が開催されました！

日本弁理士会の「知的財産支援フォーラム」が11月30日に長野県諏訪市文化センターにて開催され，300人を超える方々にお越し頂き，大盛況のうちに幕を閉じました。

本フォーラムでは，元知的財産推進事務局長の荒井寿光氏による基調講演や，地元企業の方をパネリストに迎えたパネル討論を行い，産業振興や地域活性化に向けた知的財産制度に理解を深めました。

またフォーラム内で長野県と日本弁理士会による知的財産活用による地域活性化や県内の産業振興を目的とした知的財産支援協定の覚書の調印を長野県商工労働部長と日本弁理士会東海支部長との間で取り交わしました。これまでに日本弁理士会は全国16の道県市と同様の協定を締結しており，本協定は17番目の協定となります。



荒井氏による基調講演

覚書調印式