ソフトウェア関連発明は特殊か

－発明該当性に関する試論－

会員　**井原　光雅**

要　約

本論では，今日の情報化社会において，ソフトウェア関連発明の保護および利用の必要性が高まっているとの立場から，ソフトウェア関連発明の特許法上の発明該当性について，法律論のみならず，その前提となる技術論を概観した上でその許容性について検討する。技術論では，ソフトウェアのプログラム言語，アーキテクチャ，開発プロセスにおいて，抽象化が重要な役割を果たしていることを示す。これを踏まえ，ソフトウェア関連発明が「自然法則を利用した」といえるか，「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」かという発明該当性について検討する。

目次

1　はじめに

かつて，未来学者アルビン・トフラーは，「第三の波」[(1)]で農耕社会，産業社会に次ぐ脱産業社会を予見していた。すなわち，狩猟採集社会から農耕社会への転換の契機が農業革命であり，農耕社会から産業社会への転換の契機が産業革命であった。そして，産業社会から脱産業社会への転換の契機が，情報革命であると目されている。21 世紀に入った今，世界は確かに脱産業社会，すなわち情報化社会に向けて進展しているように見える[(2)]。

今日，情報化社会を支えるソフトウェアは至るところで使用されている。身近なところでは，携帯電話やパーソナルコンピュータにソフトウェアは不可欠である。さらに，建設，製造，運輸，卸売・小売，金融・保険，医療をはじめとしてほぼすべての産業においてソフトウェア技術，特に業務改革における情報処理技術が活用されている。また，ソフトウェア技術は，遺伝子解析などのバイオ技術，デリバティブなどの金融工学といった他の分野にも応用されている。

ところで，現行の特許法は，産業の発展に寄与することを目的とする（特許法第 1 条）。昭和 34 年改正の現行特許法は，当時の産業社会を前提にしていたのであろう。その後，情報化社会の進展に伴い，社会の要請に応えるべく，平成 14 年改正法において，世界に先駆けて，「プログラム等」を「物」の発明と認める改正が行われた。また，特許庁においても，審査基準や運用指針の改定により，「コンピュータ・ソフトウエア関連発明」[(3)]（以下，「ソフトウェア関連発明」）が特許法上の「発明」として認められる要件を明確化している。（別表）

しかしながら，実務的には未だ，ソフトウェア関連発明について，特許法 29 条 1 項柱書の拒絶理由を受けることが多く，請求項に記載したものが特許法上の「発明」にあたるか否か（以下，「発明該当性」）が問題になる。また，請求項に記載したものが特許法上の「発明」として認められるとしても，そのための要件をどのように解すべきかが問題になる。さらに，この問題は，米国での近時の Bilski 判決にみられるように日本固有のものではない[(4)]。

本論では，今日の情報化社会において，ソフトウェア関連発明の保護および利用の必要性がますます高まっているとの立場から，ソフトウェア関連発明の特許法上の発明該当性について，法律論のみならず，その前提となる技術論を概観した上でその許容性について検討する。

2　発明該当性の基準

ソフトウェア関連発明においては，他の発明と同様に新規性および非容易性の特許要件（特許法第 29 条）に加えて発明の詳細な説明および特許請求の範囲の記載要件（同法第 36 条）が問題になるが，特に発明であることの要件（同法第 29 条 1 項柱書）も問題となる。通常，他の技術分野において，発明該当性が問題になることは少なく[5]，これがソフトウェア関連発明における特殊性であると考えられる。

この発明該当性の要件は，特許法第 29 条第 1 項柱書に規定されている「産業上利用することができる発明」にある。「産業上利用することができる」については，ソフトウェア関連発明は，産業上利用するためのものであることが通常であるから，「産業上の利用性」が問題になることはまずない[6]。また，「発明」については，同法第 2 条第 1 項において，「自然法則を利用した技術的思想の創作のうち高度のもの」と定義されており，この定義中，「高度のもの」は，主として実用新案法における考案と区別するためのものであり，発明該当性の判断において考慮する必要はないとするのが通説である[7]。そのため，特許法第 29 条第 1 項柱書の要件は，ソフトウェア関連発明に関する限り，特許法上の「発明」に該当するか否か，すなわち，「自然法則を利用した技術的思想の創作」に当たるか否かに集約される。

審査基準第Ⅶ部第 1 章 2.2.1 において，ソフトウェア関連発明の発明該当性の「基本的な考え方」は，次のように記載されている。

> *「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」場合，当該ソフトウエアは「自然法則を利用した技術的思想の創作」である。*
>
> *「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」とは，ソフトウエアがコンピュータに読み込まれることにより，ソフトウエアとハードウエア資源と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又は加工を実現することにより，使用目的に応じた特有の情報処理装置（機械）又はその動作方法が構築されることをいう。*
>
> *そして，上記使用目的に応じた特有の情報処理装置（機械）又はその動作方法は「自然法則を利用した技術的思想の創作」ということができるから，「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」場合には，当該ソフトウエアは「自然法則を利用した技術的思想の創作」である。*

また，審査基準第Ⅶ部第 1 章 2.2.2 において，ソフトウェア関連発明の発明該当性の「判断の具体的な手順」は，次のように記載されている。

> *請求項に記載された事項に基づいて，請求項に係る発明を把握する。なお，把握された発明が「自然法則を利用した技術的思想の創作」であるか否かの判断に際し，ソフトウエア関連発明に特有の判断，取扱いが必要でない場合には，「第Ⅱ部第 1 章　産業上利用することができる発明」により判断を行う。*
>
> *請求項に係る発明において，ソフトウエアによる情報処理が，ハードウエア資源（例：CPU 等の演算手段，メモリ等の記憶手段）を用いて具体的に実現されている場合，つまり，ソフトウエアとハードウエア資源と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又は加工を実現することにより，使用目的に応じた特有の情報処理装置（機械）又はその動作方法が構築されている場合，当該発明は「自然法則を利用した技術的思想の創作」である。*
>
> *一方，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ない場合，当該発明は「自然法則を利用した技術的思想の創作」ではない。*

このように，審査基準において，ソフトウェア関連発明の発明該当性は，「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」か否かに依る。この基準は，法律的というよりは技術的である。したがって，この基準を実際にどのように解釈し，適用するかについては，ソフトウェアの技術的な理解を必要とするものと思われる。

3　技術論としての「ソフトウェア」

（1）　コンピュータのアーキテクチャ

ソフトウェアが稼働するコンピュータを単純化して表現すると，図 1 のようになる。コンピュータの処理は，究極的にはデータの操作と格納に還元される。より具体的には，プログラムに従って処理装置が，記憶装置上のデータを操作・格納することによって，所定

の機能を実現する。

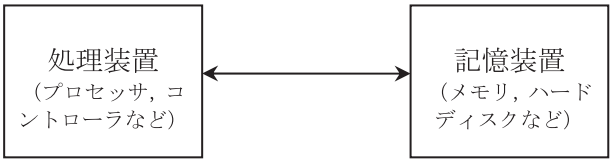

図１：コンピュータの単純化したアーキテクチャ

実用には，コンピュータは，さらに入力装置と出力装置を必要とする。これらの装置は，人間とのインターフェイスを提供する。すなわち，人間がコンピュータにデータを入力するためにはマウスやキーボード，スキャナなどの入力装置が必要になり，人間がコンピュータの処理結果を知覚するためにはディスプレイやスピーカ，プリンタなどの出力装置が必要になる。

記憶装置はさておき，処理装置はどのようなハードウェアであろうか。処理装置の典型であるプロセッサは，ソフトウェアプログラムに含まれる命令に従って，データを操作するものである。プロセッサは，複数の基本的な命令[8]を組み合わせて，ソフトウェアプログラムに含まれるすべての命令を実現している。この基本的な命令を「命令セット」("instruction set") という。どのような複雑な処理も，究極的にはすべてこの基本的な命令の組み合わせで実現されている。したがって，CやJavaなどの高水準プログラミング言語（高級プログラミング言語ともいう）で表現されたプログラムもすべてハードウェア資源たるプロセッサが実現する低水準の命令セットに還元される。

実際，今日のプロセッサの設計は，この命令セットをどのように設計するか（命令セット・アーキテクチャ）にかかっている。この命令セットが決まると，その命令セットを実現するプロセッサのハードウェアを設計することができる[9]。そして，同じ命令セットが実現されていれば，ハードウェアの設計が異なっていても，基本的には同じソフトウェアが稼働する[10]。そのため，この命令セットが決まると，その命令セットを使用するソフトウェアを開発することができる。すなわち，この命令セットがハードウェアとソフトウェアが直接インターフェイスする部分なのである[11]。

（2） プログラム言語

命令セットを実現するプロセッサは，デジタル的に，すなわち電気的な信号のオン／オフによって動作する。そのため，プロセッサによる処理は，2進数によって表現され，実行される[12]。この2進数で表現されたプロセッサに対する命令を機械語という。

機械語は，2進数で表現されるため人間にとって非常に扱いにくい。そこで，機械語を人間の理解しやすいシンボル形式で表現したアセンブリ言語が開発された。アセンブリ言語は，アセンブラと呼ばれるプログラムによって2進数の機械語に翻訳される。しかし，このアセンブリ言語も，機械語の文法に従っており，人間の自然言語とは大きく異なる。これは，プログラマが機械（プロセッサ）のアーキテクチャを理解し，機械のように思考しなければならないことを意味する。そこで，人間の自然言語に近い高水準プログラミング言語が開発された。高水準プログラミング言語は，コンパイラというプログラムによってアセンブリ言語に翻訳される。この高水準プログラミング言語には，CやJavaなどがある[13]。機械語やアセンブリ言語は，プロセッサのアーキテクチャに固有であるが，高水準プログラミング言語は，これに依存せず，コンパイラさえ用意すれば，高水準プログラミング言語で開発されたソフトウェアは，アーキテクチャの異なるプロセッサでも利用可能である[14]。

このように，高水準プログラミング言語の利点の一つは，プログラマはもはや機械（プロセッサ）のアーキテクチャを意識することなく，独立してソフトウェアを開発することができるようになったことである[15]。この点で，高水準プログラミング言語で開発されるソフトウェアは，もはやハードウェア資源（プロセッサ）から独立しており，したがってハードウェア資源との具体的な協働関係もなくなっているのである[16]。そして，今日のソフトウェアは，そのほとんどが高水準プログラミング言語で開発されている。

（3） ソフトウェアのアーキテクチャ

一般に，ソフトウェアは，ワープロソフトなど特定の作業や業務を目的としたアプリケーションソフトウェアと，ハードウェアの管理や基本的な処理を，アプリケーションソフトウェアやユーザーに提供するオペレーティングシステムなどのシステムソフトウェア（基本ソフトウェア）に分けられる[17]。このように，ソフトウェアはその機能に応じて階層化されている。実際，大規模なソフトウェアは，それ自体いくつかの階層から構成されている。

階層化の利点は，下位層と上位層との間で必要な項目だけをやりとりし[18]，下位層の詳細を上位層に対し

て隠ぺいすること（抽象化）である。これは，ソフトウェアの設計に固有のものではなく，人間が大規模化に伴う複雑さに対処する際の常とう手段である。ハードウェアの設計においても，抽象化と階層化が行われている。例えば，プロセッサの命令セットは，下位層の物理的要求と上位層の論理的要求との間で設計される。次に，命令セットを実現する物理設計は，下位層の物性的要求と上位層の物理的要求との間で決定される。さらに，物理設計を実現する半導体チップは，下位層たる半導体製造技術の要求と上位層の物性的要求との間で決定される。別の例をあげると，自動車産業は，すそ野の広いピラミッド型の構造をしているとよくいわれる。この構造も抽象化と階層化の結果である。自動車メーカーは，部品メーカーとともに必要な部品の仕様を決めて（抽象化），部品メーカーに開発と製造を依頼する（階層化）。その場合，自動車メーカーは，その部品の開発と製造に関するすべての詳細を知る必要はない。自動車メーカーは，同じ仕様のものを他の部品メーカーに依頼することもある。この場合，同じ仕様の部品でも部品メーカーによって開発・製造の詳細は異なるであろう。今日の大規模なソフトウェア開発は，この抽象化と階層化に基づいている。

図２に，一例として，データベースを用いたシステムの階層構造を示す。この図は，階層構造を示すためにかなり単純化している。

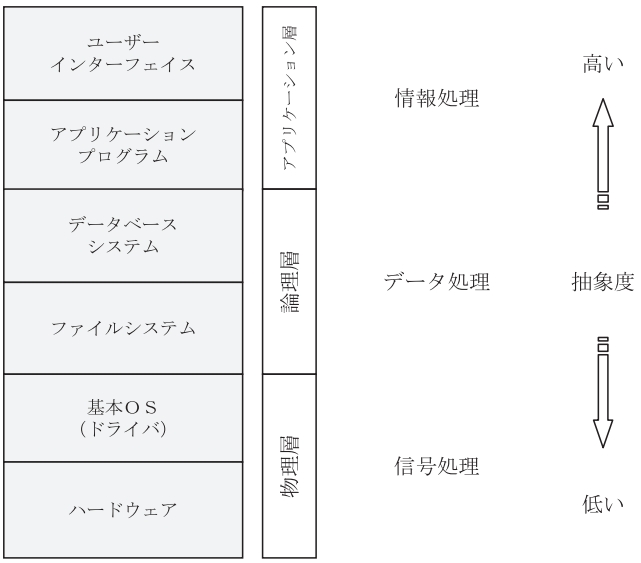

図２：データベースを用いたシステムの階層構造

図２において，ハードウェアは，プロセッサなどの処理装置，メモリやハードディスクなどの記憶装置，マウスやキーボードなどの入力装置，ディスプレイやプリンタなどの出力装置から構成される。ハードウェアの各装置は，物理的な信号を介して動作する。これら物理的な信号は，演算処理のように２進数のビットを表すデジタル的なものもあれば，音声や画像の処理のように実世界との入出力を表すアナログ的なものもある。このような物理的な信号の処理は「信号処理」と呼ばれ[19]，後述する人間にとって意味のある情報を処理する「情報処理」と対比されることがある。

基本オペレーティングシステム（OS），特にハードウェアドライバは，下位層のハードウェアを直接管理したり，制御したりする[20]。基本 OS のハードウェアドライバは，ハードウェアの各装置に対しては物理的な信号を扱うが，上位層に対しては物理的な信号から必要なデータのみを抽出する。例えば，ハードディスク上の複数の場所（セクター）に格納されたビットを一まとまりのデータとして抽出したり，上位層からのデータを空いている複数のセクターに格納したりする。このとき，基本 OS は，どのデータのどの部分がどのセクターに格納されているかを管理している。

ファイルシステムは，基本 OS からのデータをファイルとして扱うためのものである。また，ファイルを束ねて管理するディレクトリ（フォルダ）の機能も提供する。階層構造のこのレベルでは，ファイルは，記憶装置上のビットから抽象化されている。すなわち，ファイルシステムは，上位層に対して，ファイルが実際に記憶装置上のビットとしてどこにどのように格納されているかを意識させることなく，ファイルを取り扱えるようにしている。

データベースシステムは，ファイル単位の管理から，情報[21]の性質に基づく管理を実現する。これは，情報を要素に分解し，その関係性とともに管理することである。情報は，発生，変更，消滅するので，複数人がある情報を管理したり，ある情報を複数のファイルで管理したりすると，情報の整合性が問題になる。データベースシステムでは，情報を要素とその関係性で管理することにより，情報要素が変動してもデータベース全体での整合性が保たれるようにしている。これにより，情報を多くの拠点間で矛盾なく共有できるだけでなく，情報の集計や統計などの高度な機能を提供することができる。そのため，データベース化された情報は，上位層にとって，ファイルの構造に縛られることなく，加工しやすく，扱いやすいものとなる。

プログラムは，一般に，コンピュータに対する命令（処理）を記述したものである[22]。図２のアプリケー

ション・プログラムは，データベース上の情報を加工したり，利用したりするものである。例えば，検索プログラムは，データベース上の情報から所望の情報を検索する。販売管理プログラムは，データベース上で販売データを管理する。物流管理プログラムは，データベース上で物の流れを管理する。集計プログラムは，データベース上の情報を所望の基準に応じて集計する。このように，アプリケーション・プログラムは，上位層に対して，必要な機能を提供する。

ユーザー・インターフェイスは，下位層のアプリケーション・プログラムと上位層のユーザーとの間で情報をやりとりするためのものである。換言すると，ユーザー・インターフェイスは，下位層のアプリケーション・プログラムに対するユーザーの入力および出力のための手段を提供する。このインターフェイスを介して，人間であるユーザーはシステムが処理した内容や結果を知覚・認識することができる。これに対して，アプリケーション・プログラム・インターフェイス（API）は，プログラムに対する他のアプリケーション・プログラムの入力および出力のための手段を提供する。そのため，API を介して，アプリケーション・プログラムを別のプログラム（例えば，図２のデータベースシステム）の上に階層的に構築することができる。

図２の階層構造において，基本 OS のレベルまではハードウェア資源との協働関係はかなり具体的である。しかし，ファイルシステムから上のレベルでは，概念的には，物理的な制約はなく，ハードウェア資源との協働関係は抽象化されている。ただし，図２のシステムが実際のハードウェア資源上に実装されると，ファイルシステムから上のレベルでも，理論的には，ハードウェア資源との具体的な協働関係が生じることになる。すなわち，特定のソフトウェアが特定のハードウェア資源上にインストールされると，システム全体として具体化された厳密かつ一義的な関係が生じる結果，ファイルシステムから上のレベルでも，ハードウェア資源との具体的な協働関係が生じる。したがって，このようなシステムについて，ハードウェア資源との具体的な協働関係を要求することは，発明の範囲を一つの実装形態にまで縮減することに等しい[23]。

（4） ソフトウェアの開発プロセス

ソフトウェアが大規模化・複雑化するにつれて，ソフトウェア開発プロセスにはさまざまな方法論が提唱されてきた[24]。しかし，いずれの方法論においても，ソフトウェアの開発プロセスは，機能，設計，実装の３つの段階に大別される[25]。

抽象度の最も高い「機能」の段階では，ソフトウェアの機能を仕様として定義する。例えば，ワープロのコピー・アンド・ペースト機能であれば，仕様は，「ユーザーが選択したテキストをコピーして，別の場所にペーストできるようにするプログラム」となる。「設計」の段階では，「機能」を実現するための計画を策定する。具体的には，「機能」を実現するための処理の流れ（フローチャート）を確定し，必要なルーチン，モジュール，アルゴリズムを特定する。「実装」の段階では，「設計」に従って，命令を組み合わせて「機能」を実現するプログラムを書く[26]。実際には，この後にさらにテストという段階が続く。

いずれの段階においても「技術的思想の創作」はなされ得る。しかし，発明の特定の仕方は各段階で異なるであろう。「機能」の段階では，発明も抽象度が高く，したがってハードウェア資源との協働関係も希薄である。すなわち，この段階では，発明たる「機能」をどのように実現するかは，発明の本質的な問題ではない[27]。なぜなら，この段階ではその機能はソフトウェアやハードウェアといった区分を超えて様々な態様で実現され得る可能性を有しているからである。そのため，このような発明に対して，ハードウェア資源との具体的な協働関係を要求することは，発明の範囲を一つの実施形態に縮減することに等しい。

「設計」の段階でも発明はなされ得る。例えば，同じコピー・アンド・ペースト機能でも，設計によってはより高速に処理できたり，より直感的に理解できたりするかもしれない。この段階では，発明である「設計」はより具体的である。なぜなら，この段階では，フローチャートが確定され，ルーチン，モジュール，アルゴリズムが特定されているからである。

「実装」の段階でも発明はなされ得る。しかし，ほとんどの場合，新規性および非容易性の要件を満たさないであろう。すなわち，単なる命令の組み合わせは，特許発明としては認められにくい[28]。しかし，特定の命令の組み合わせたるプログラムは著作権の対象になる（著作権法第 10 条第１項第９号，なお同条第３項）。

４　ソフトウェア関連発明の特殊性

以上，コンピュータのアーキテクチャを踏まえ，ソ

フトウェアのプログラム言語，アーキテクチャ，開発プロセスを概観してきた。ソフトウェアの場合，いずれの側面においても，抽象化が重要な役割を果たしていることが分かる(29)。プログラム言語においては，ハードウェア資源から独立した高水準プログラミング言語が主流となり，アーキテクチャにおいては，物理層から独立した上位層のプログラムが様々な機能を提供し，開発プロセスにおいては，「機能」レベルでの技術的思想の創作が行われている。

この抽象化は，ソフトウェア技術の進展の結果である。事実，これは，技術者の長年にわたる技術的課題の克服の結果である。しかしながら，この抽象化のために，発明として，「自然法則を利用した」といえるか，「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」かという発明該当性が問題になる。

確かに，ソフトウェア関連発明を特許請求の範囲に記載すると，必ずしも「自然法則を利用した」ものとはいえない場合が含まれ得る。例えば，請求項にコンピュータ等のハードウェア資源の記載がなく，情報を処理するステップのみを記載した場合である。そのような場合には，請求項の記載によって特定されるものは，人間の精神活動によって実現される態様のものを含み得る。そのような態様のものは，全体として自然法則を利用しているとはいえない場合があり，特許法の規定する「発明」としての保護範囲を超える態様を含み得る。そうすると，特許法に不慣れな第三者は，不測の不利益を被るおそれがある。そのため，請求項の記載において，「自然法則を利用した」発明であることが明らかとなるようにする必要がある(30)。

しかし，発明は，そもそも技術的思想の創作であり，本来，概念的なものである。したがって，ソフトウェア関連発明に対して，ハードウェア資源との協働関係を過度に要求することは(31)，発明の範囲を実施形態にまで縮減することに等しく，発明の保護に欠けることになろう。他方，ハードウェア資源との協働関係をまったく要求しなければ，全体として自然法則を利用しているとはいえない場合にも発明として認められるとの誤解が生じ得るし，現行法のもとでは妥当性を欠くことになろう。

そこで，「自然法則を利用した」といえるか否かは，そのソフトウェア関連発明の抽象化のレベルに応じて実質的に判断すべきであると考える。具体的には，抽象化のレベルの高い発明は，本来，ハードウェア資源との協働関係に特徴があるのではなく，ソフトウェアによる情報処理に特徴があると考えられる(32)。そうであれば，「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」とは，この場合，「ハードウエア資源を用いて具体的に」ではなく，「ソフトウエアによる情報処理が，・・・具体的に」実現されていると解すべきである(33)。対照的に，抽象化のレベルの低い発明は，ソフトウェアによる情報処理にではなく，ハードウェア資源との協働関係に特徴があることが多い。そのような場合には，「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」とは，「ソフトウエアによる情報処理が，・・・具体的に」ではなく，「ハードウエア資源を用いて具体的に」実現されていると解すべきである(34)。なぜなら，発明として「具体的に」特定されるべきは，発明の特徴部分であるべきだからである。

5　おわりに

本論では，ソフトウェア関連発明について，発明該当性の基準を所与として，技術論を交えて検討した。しかし，そもそも条文上の「自然法則を利用した」との文言から，「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」との基準が導き出される根拠は必ずしも明らかではない(35)。このように，特許法上，「ソフトウェア関連発明」は特殊であり，特殊な取り扱いが必要であるとする根拠は何であろうか。これは，多分に政策的な問題のように思われる。すなわち，ソフトウェア関連発明は，特許法の目的に合致し，積極的に保護する必要があるかどうかということであろう(36)。

この点，平成13年12月産業構造審議会知的財産政策部会「ネットワーク化に対応した特許法・商標法等の在り方について」第2章第1節1．（1）では，次のように記載されている。

> *ソフトウェア関連発明の特許保護を考える際，「自然法則を利用した技術的思想の創作」という現行特許法上の発明の定義，特に「自然法則の利用」という要件が，ソフトウェア関連発明の特許適格性（発明の成立性）を認める上での制約となっているとの指摘がある。*
>
> *しかし，実際には，審査基準の累次の改訂により，発明の定義を弾力的に解釈し，ソフトウェア*

関連発明の特許適格性を広く認める運用が行われており，ビジネス方法発明を含むソフトウェア関連発明の特許適格性の判断については，日米の運用に大きな差はない。したがって，今後の発明の定義規定及びその解釈･運用のあり方については，なお検討を継続していく必要のあるものの，現行特許法上の発明の定義が，ソフトウェア関連発明の特許法による保護を実質的に妨げる制約要因となっているとは認められない。また，現在の我が国の運用は，産業界等からも肯定的に受け止められており，今後もソフトウェア関連発明の特許法による保護を積極的に進めて行くべきである。

確かに，実務的には，発明該当性の理由のみで拒絶査定となることは少ない。また，特許後に，発明該当性の理由で無効となることも少ないであろう。しかし，発明該当性の拒絶理由を解消するために，発明の範囲を限定せざるを得ないことが少なくないのも現実である。また，不慣れな出願人は，発明に該当しないのであれば特許法による保護の埒外であると考えて，権利化自体を断念することも多いであろう。

本論は，あくまで「ソフトウェア関連発明」が，情報化社会において産業の発展に寄与しているとの認識の下，発明としての保護および利用が望まれているとの立場に立ち，実務的な観点からソフトウェア関連発明の発明該当性について検討した。しかし，上記の報告書から 10 年が経過しようとする今，ソフトウェア関連発明の特殊性，即ち，発明該当性について，再度，より本質的な議論が望まれる時機ではないだろうか。

別表：審査基準および運用指針の変遷(37)

1975（S50）	コンピュータ・プログラムに関する発明についての審査基準（その 1） －手法の因果関係が自然法則を利用しているか否か。 －コンピュータ・プログラムに関する発明が「方法」の発明として特許されうることを明示。 －コンピュータ・プログラムそのものは，“きわめて抽象的なもの”として保護対象外。 －媒体クレームは，単に手順が記録されたものにとどまり，保護対象外。
1982（S57）	マイクロコンピュータ応用技術に関する発明についての運用指針 －マイクロコンピュータ応用技術に関する「物」（装置）発明は特許されうることを明示。
1988（S63）	コンピュータ・ソフトウエア関連発明の審査上の取扱い（案） －従来の「審査基準」及び「運用指針」を補足。
1993（H5）	特定技術分野の審査基準第 1 章「コンピュータ・ソフトウエア関連発明」 －自然法則利用性の要件を明確化。 －コンピュータ・プログラムそのものは，“技術的思想でない”として保護対象外。 －媒体クレームは，“技術的思想でない”として保護対象外（単なる情報の提示）。
1997（H9）	特定技術分野の審査基準第 1 章「コンピュータ・ソフトウエア関連発明」 －プログラムを記録した記録媒体の保護。 －プログラムクレームは記載要件違反（従来の“技術的思想でない”からの運用変更）。
2000（H12）	特定技術分野の審査基準第 1 章「コンピュータ・ソフトウエア関連発明」 －媒体に記録されていないコンピュータ・プログラムを「物の発明」として取扱うことを明らかにした。 －ハードウェアとソフトウェアを一体として用い，あるアイデアを具体的に実現しようとする場合には，そのソフトウェアの創作は，特許法上の「発明」に該当することを明らかにした。 －ビジネス関連発明の進歩性の判断に関する事例を充実させ，個別のビジネス分野とコンピュータ技術分野の双方の知識を備えた者が，容易に思いつくものは進歩性を有しないことを明らかにした。

注

(1) Alvin Toffler, “The Third Wave,” 1980

(2) ピーター・ドラッカーはこの変化の基底を「知識社会の到来」という言葉で表現している。（Peter F. Drucker, “The Age of Discontinuity,” 1968）

(3) 審査基準第Ⅶ部第 1 章によれば，「コンピュータ・ソフトウエア関連発明」は，「その発明の実施にソフトウエアを必要とする発明」であるとしている。

(4) Bilski 判決はビジネス方法に関するものであるが，ソフトウェア関連発明と無関係ではない。この点に関して，現行の審査基準第Ⅱ部第 1 章 1.1(4)には次のように記載されている。

ビジネスを行う方法やゲームを行う方法に関連する発明は，物品，器具，装置，システムなどを利用している部分があっても，全体として自然法則を利用しない場合があるので，慎重に検討する必要がある。（事例 5 ～ 7 参照）

なお，ビジネスを行う方法やゲームを行う方法という観点ではなく，ビジネス用コンピュータ・ソフトウエアやゲーム用コンピュータ・ソフトウエアという観点から発明すれば，「発明」に該当する可能性がある。（コンピュータ・ソフトウエア関連発明における判断については，「第Ⅶ部　第 1 章　コンピュータ・ソフトウエア関連発明」2.2 参照）

(5) なお，現行の審査基準第Ⅱ部第 1 章によれば，特許法第 29 条第 1 項柱書の要件は，「発明該当性」と「産業上の利用性」に分けられる。「発明該当性」が問題となるのは，主に「コンピュータ・ソフトウエア関連発明」であり，「産業上の利用性」が問題となるのは，主に「人間を手術，治療又は診断す

る方法」である。

(6)「産業上の利用性」は可能性があれば足りるとするのが通説である。（吉藤幸朔著「特許法概説」第13版72頁，中山信弘著「工業所有権法〈上〉特許法」第2版増補版114頁参照）

(7)審査基準第Ⅱ部第1章1参照。

(8)基本的な命令には，例えば，算術演算，論理演算，条件分岐などがある。

(9)ハードウェアは，例えば，加算器，ビットシフタ，マルチプレクサ，コンパレータなどで構成される。

(10)AMDのIntel x86互換のマイクロプロセッサは，Intelのものとはハードウェアの設計が異なるものの，命令セットが同じであるため互換性がある。

(11)David A. Patterson and John L. Hennessy, “Computer Organization and Design, Third Edition: The Hardware/Software Interface,” 2004

(12)処理されるデータも，2進数によって表現され，操作・格納される。

(13)その他にも，科学技術計算用のFortran，事務処理計算用のCOBOL，記号処理用のLISPなどさまざまなものがある。

(14)実際，新たなプロセッサを開発する場合，コンパイラを用意して，これまでに開発されてきたソフトウェア資産を活用できるようにするのが通常である。

(15)ただし，ソフトウェアの最適化のためには，プロセッサの処理速度やメモリ容量などのハードウェア要件を考慮しなければならない。

(16)理論的には，ソフトウェアがコンパイルされ，ターゲットのコンピュータにインストールされると，そのコンピュータのハードウェア資源との具体的な協働関係が生じる。

(17)ウィキペディア，「ソフトウェア」のページを参照

(18)下位層と上位層との間で必要な項目をやりとりする手段をインターフェイスという。

(19)このような物理的な信号の制御・処理について，審査基準第Ⅶ部第1章2.2.2（注）では，ソフトウェア関連発明に特有の判断，取り扱いが必要でなく，「第Ⅱ部第1章　産業上利用することができる発明」により判断を行う例として，「請求項に係る発明が，(a)機器等（例：炊飯器，洗濯機，エンジン，ハードディスク装置）に対する制御又は制御に伴う処理を具体的に行うもの，又は(b)対象の物理的性質又は技術的性質（例：エンジン回転数，圧延温度）に基づく情報処理を具体的に行うものに当たる場合は，「自然法則を利用した技術的思想の創作」である。」としている。

(20)Windowsの前身であるMS-DOSは，ディスク・オペレーティング・システム（DOS）であり，主にハード・ディスクやフロッピー・ディスクなどのディスク管理を行うためのものであった。

(21)データは，記憶装置に蓄積されている状態であるのに対して，情報は，ある目的に利用するために解釈された状態であるとする定義の仕方がある。

(22)ウィキペディア，「プログラム（コンピュータ）」のページを参照

(23)また，現実問題として，このような協働関係を発明の詳細な説明や特許請求の範囲に記載することはほとんど不可能である。

(24)このような方法論には，ウォーターフォール型，スパイラル型，反復型，アジャイル型などがある。

(25)この分類はハードウェアの開発プロセスにも当てはまる。

(26)Kirk Teska, “SOFTWARE PATENTS 101,” IEEE Spectrum, March 2008, p20.

(27)実施可能要件の問題にはなり得る。

(28)なお，審査基準第Ⅱ部第1章1.1(5)(b)によれば，コンピュータプログラムリスト（コンピュータプログラムの，紙への印刷，画面への表示などによる提示（リスト）そのもの）は，情報の単なる提示（提示される情報の内容にのみ特徴を有するものであって，情報の提示を主たる目的とするもの）に当たり，技術的思想でないものとされている。

(29)近年，ハードウェアの設計および開発においても，コンピュータ（ソフトウェア技術）が広く利用されており，ハードウェアの抽象化やモデル化が重要な役割を果たしている。

(30)このような必要性以外にも，ソフトウェア関連発明は，その性質上，請求項の記載が抽象的となり，発明が特定し難く，その外延も不明確であるとする見解もあるが，これは，本質的には記載要件の問題であろう。

(31)厳密に言えば，引用例に記載された発明についても，ハードウェア資源との協働関係，すなわち発明該当性を判断しなければならないことになる（特許法第29条第1項各号）。

(32)例えば，仮想化技術や分散コンピューティング技術は，ハードウェア資源との協働関係を特定しにくい。というのは，仮想化技術は，ハードウェア資源との協働関係を仮想化する点に特徴があり，分散コンピューティング技術は，処理の主体を複数のハードウェア資源のいずれにも特定しない点に特徴があるからである。

(33)これに該当する事例としては，審査基準第Ⅶ部第1章3.2.1事例2-3，2-5が挙げられる。

(34)これに該当する事例としては，審査基準第Ⅶ部第1章3.2.1事例2-1が挙げられる。

(35)特に，「具体的に」との具体性の要件の内容および根拠は必ずしも明らかではない。

(36)例えば，Research on Innovationの所長James Bessenと2007年に「メカニズム・デザイン理論」の基礎を築いたとしてノーベル賞を受賞した経済学者Eric S. Maskinは，累積的かつ相補的な技術開発が行われる分野，特にソフトウェア・パテントについて，特許制度は全体的なイノベーションを，延いては社会的利益を減じる可能性がある，としている。（Sequential Innovation, Patents, and Imitation, by James Bessen and Eric Maskin, Discussion paper, MIT（2000）, published in The RAND Journal of Economics, Volume 40, Issue 4, pages 611-635, Winter 2009）

(37)小栗昌平「コンピュータ・ソフトウェア関連発明と特許庁の審査基準」，平成11年3月18日・19日知的財産権講座「コンピュータ・ソフトウェア関連発明と特許出願」テキスト，発明協会

（原稿受領2011. 12. 21）