ソフトウェア開発委託契約紛争事例の研究 (2)

内　布　　光

はじめに

本稿は、「現代法学」第 10 号に掲載した前稿「ソフトウェア開発委託契約紛争事例の研究（1）」の続稿である。

前稿で明らかにしたように、ソフトウェア開発委託取引の契約は、もともと新種の非典型契約ということができるが、実際の取引では請負や準委任などの典型契約に当てはめて行われることが多く、中でも請負契約に当てはめて（いわゆる請負型で）取引されている場合が多い。

そして、ソフトウェア開発委託取引を巡る法的紛争についての主要な裁判事例を見ても、この取引は請負型で行なわれていることが多く、この争点としては、債務不履行ないしは瑕疵担保責任に関するものが中心であることも前稿で明らかにした。

そこで、本稿では、このような請負型のソフトウェア開発委託取引契約（以下「ソフトウェア開発請負契約」という）における法的紛争の主要な原因とされる債務不履行および瑕疵担保責任を中心に考察したい。

なお、ソフトウェア開発請負契約の当事者を見ると、通常、注文者は民間企業や官公庁等となっており、個人（消費者）がなることは殆どない。他方の請負人はソフトウェアハウス等の IT 専門事業者がなっている場合が殆どである。このうち本稿では、注文者が官公庁である場合を除外し、民間企業である場合いわゆる企業間取引を前提とする。

以下本稿では、注文者を「ユーザー」といい、請負人を「ベンダー」という。

1．ソフトウェア開発請負契約と債務不履行

ソフトウェア開発請負契約における法的紛争の主要原因として、請負人であるベンダーの債務不履行があるが、なぜ、ベンダーは債務不履行に陥

りやすいのであろうか。

この問題を分析するためには、ソフトウェア開発請負契約が他の典型的な請負契約（例えば、日常生活の中で身近な契約として馴染んでいる運送、住宅建築、洋服オーダーメイドなど）とは異なる特殊性について理解しておく必要がある。そして、ソフトウェア開発委託の契約は、もともと民法の請負契約にはぴったりと当てはまらないという性質を有する（いわゆる非典型契約である）が、取引実務上、これを請負契約に当てはめて（いわば請負契約のように取り扱って）いることを考慮しなければならない。つまり、ソフトウェア開発請負契約を典型的な請負契約と同様に取り扱うことにはもともと無理が生じているのであり、その象徴としてベンダーの債務不履行などの問題が生じやすいのである。

そこで、ベンダーが債務不履行に陥りやすい原因を理解するために、ソフトウェア開発請負契約の成立プロセスやベンダーの仕事の内容など、この契約が有する特殊性とユーザー・ベンダーとの関係や法的義務を明らかにしたい。

(1)　ソフトウェア開発請負契約の成立プロセス

ソフトウェア開発請負契約におけるベンダーの債務不履行の問題点を考察するためには、この契約の成立からベンダーの開発業務（作業）着手に至るまでのプロセスや開発業務（目的とする仕事の内容）の特殊性を理解しておく必要がある。

そこでまず、ソフトウェア開発請負契約の成立までプロセスを理解することにする。

日常生活における売買などの契約は、原則として申込と承諾という相対立する意思表示の合致で成立する。しかし、企業間取引として結ばれるソフトウェア開発請負契約は、日常生活における売買契約のように単純な内容ではなく、当事者間で合意を要する事項が多岐に分かれ複雑であるから、

合意事項を契約書などに書面化するのが通例となっている。また、この契約においては、ソフトウェアという抽象的な目的物を対象とし、目的とする仕事（ソフトウェア開発業務）の内容が極めて技術的行為であるので、当事者間では開発すべきソフトウェアの捉え方などについてしばしば齟齬が生じるため、両者の意思を統一するためのプロセスが必要となる。

ここで、ソフトウェア開発請負契約成立までの流れを見ると、通常、以下のプロセスがとられている。

① ユーザーから「このような業務の処理ができるソフトウェアを開発してもらいたいので提案してほしい」旨を記載にした提案依頼書[1)]が複数のベンダーに提示される。

② これを受けて、各ベンダーは開発すべきソフトウェアの内容（構成プログラム機能）や開発期間等を記載したシステム提案書および開発費用に関する見積書を作成してユーザーに提出する[2)]。

③ ユーザーは、各ベンダーから提出されたシステム提案書および見積書を比較して開発委託するベンダーを選定する。（開発作業が大規模の場合や高度技術を要する場合などには、2〜3 社を選定し、④のプロセスに入ることもある。）

④ 選定されたベンダーは、ユーザーとの間でソフトウェアの具体的内容や開発費用・スケジュールその他重要な取引条件等を話し合って（いわゆる契約交渉を行ない）、この結果をもとに契約書を作成し、当事者で契約を締結（契約書に調印）する。なお、ソフトウェアの具体的内容や開発スケジュール等は、②のシステム提案書に基づき協議・交渉されてシステム仕様書としてまとめられ、このシステム仕様書は契約内容の一部を構成する。

上記①〜④の各プロセスのうち、②および③のプロセスは絶対欠かせないので、実際の取引でも、②および③のプロセスの行為は、何らかの形で（きちんとしたシステム提案書や見積書ではなく、これに代わる資料で行

う場合を含めて）行われていると思われる。

しかし一方では、①のプロセスが全く採られない場合（例えば、直接口頭で、ユーザーから複数のベンダーに対して、②のシステム提案書や見積書の提出を要求するなど）や④のプロセスでの契約交渉が十分に行われずに、きちんとした契約書（システム仕様書を含む）が作成されないまま、ソフトウェア開発作業に着手されたりする実態も少なからず多いといわれている。特に、④のプロセスでの契約交渉が十分に行われず、契約書やシステム仕様書がきちんと作成されていないと、開発すべきソフトウェアや開発業務の内容が曖昧かつ不十分となるので、後日、当事者間で契約内容を巡るトラブルが生じやすくなる。

このように④のプロセスは、当事者（ユーザー・ベンダー）が契約の成立に向けて交渉を行なう契約締結のための準備段階といえる。判例・学説は、このように契約が未だ成立していない段階においても、相手方の信頼を裏切って交渉を不当に破棄した場合には、それによって被った相手方の損害（信頼利益）を賠償する責任を認めている[3)]。

以上のことからわかるように、ソフトウェア開発請負契約は、上記の④のプロセスにおいて、当事者間で契約内容・条件等について交渉を繰り返し、その結果を盛込んだ契約書（両当事者が交渉により合意に至った個々の事項を取りまとめた書面）に調印する方法で成立させているのである。

このような契約成立方法は、企業間取引で幅広く取り交わされている「継続的取引基本契約」の成立方法と全く同じである。つまり、一般的にソフトウェア開発請負契約は、当事者間の合意の積み重ねにより契約が成立するのであり、いつの時点で契約が成立したのかを明確に特定するのは困難である[4)]。また実際に、ソフトウェア開発請負契約の成立時点を巡って当事者間で法的トラブルが生じることは殆どないといってよいが、契約成立時点が問題となる場合には、最終的に個々の合意が盛込まれた契約書に両当事者が調印（記名捺印）した時点とみるのが妥当といえる。

なお、契約の成立と同時に、原則として両当事者の債権債務が発生することになる。しかし、このように締結された企業間取引の契約書では、契約発効日（両当事者の債権債務の発生日）を特約するのが通例となっているので、実際の企業間取引契約では契約成立時期が問題となることは殆どない。

(2) ベンダーの仕事の内容

次に、ソフトウェア開発請負契約におけるベンダーの仕事の内容について見てみる。

ソフトウェア開発業務（プログラムの作成作業）は、いわば「無」から「有」を創作する行為であるので、契約上、ベンダーの仕事の内容を具体的に特定することが極めて重要であるが、実際には、このことが難しく、曖昧となっている場合が多い。

民法上、請負契約における請負人は、仕事を完成させなければならないという債務の履行義務（民法 632 条）を負っているから、ソフトウェア開発請負契約においては、ベンダーは、ユーザーが委託を受けたソフトウェア開発業務を完遂することで、ユーザーが要求する内容のプログラムを作成しなければならない。これが民法 415 条にいう「債務の本旨に従った履行」であり、ベンダーの仕事（履行すべき債務）の内容となる。

もう少し具体的にベンダーの仕事の内容を理解するために、ユーザーが自社内で使用するための業務処理用ソフトウェア（アプリケーションソフト）を新規に開発する場合を例にとって、詳しく説明する。

ユーザーの業務処理用ソフトウェアは、一般的に、ユーザーが日常的に行っている人事管理、販売管理、生産管理、経理処理などの業務をコンピュータ処理するためのプログラムである。このプログラムを作成する業務をユーザーがベンダーに請負型で委託する契約が、ここでいうソフトウェア開発請負契約である。

この場合のベンダーの仕事の内容がどのように具体化されていくのかについて、前項（1）で述べた契約成立までの流れの①〜④のプロセスに当てはめて説明すると、概ね次のようになる。

まず、①のプロセスでユーザーがベンダーに提示する「提案依頼書」には、ユーザーの人事管理、販売管理、生産管理、経理処理などの業務処理の概要が明らかにされていなければならない。

次に、②のプロセスでベンダーがユーザーに提出する「システム提案書」には、提案依頼書に記載されたユーザーの業務処理概要をもとに、当該業務をソフトウェアでどのように処理するかという観点から、システム（コンピュータ、ネットワーク及びソフトウェア）の構成やコンピュータによる情報（データ）処理内容などの概要が記載されることになる。しかし、このシステム提案書の段階では、ソフトウェア（プログラム）の内容は具体的ではなく、開発スケジュールや開発費用も概略的なものであるから、ベンダーの仕事内容も明らかでない。

続いて、ユーザーは提案書の内容などをもとに、③のプロセスにおいて、契約相手方であるベンダーを選定することになる。そして、④のプロセスにおいて、ユーザーは当該ベンダーとの間で開発すべきソフトウェアの具体的な内容について交渉し決定することが必要である。

通常、この④の交渉結果（決定された事項・内容）を盛込んで契約書が作成されるので、本来ならば、ベンダーの仕事の内容はこの段階では具体的に特定できる筈である。しかし、この交渉においては、ソフトウェアの内容を契約締結できる程度に具体化するための事項に限定されるから、契約書（契約内容の一部を成す「システム仕様書」を含む）をもってベンダーの仕事の内容を具体的に特定しているケースは一般的に少ないとされている。

更に、ベンダーの仕事の内容を特定することを困難にしているのは、ソフトウェア開発という仕事が極めて技術的内容であることも一因と考えら

れる。ユーザーは一般的にソフトウェア技術に長けていないので、高度なソフトウェア技術を駆使することを求めておらず（ユーザーは技術内容には余り関心はない）、むしろ、目的とする業務処理を簡単にできるような（ユーザーが使いやすい）ソフトウェアを求め、しかもそれが短期間で安価に開発できればよいのである。従って、④のプロセスにおいて作成されるシステム仕様書は、ソフトウェアの基本的機能は明確にされているが、詳細機能や性能までは明確にされていないので、開発行為着手後の開発プロセス（基本設計、機能設計、詳細設計という一連の設計プロセス）を通して具体明確化されていくのが一般的となっている。

このようにベンダーの仕事の内容の特定は、開発すべきソフトウェアの内容を技術的観点から詳細具体化することに依拠するので、ソフトウェア内容が具体明確化されていない段階である契約締結時において、ベンダーの仕事の内容は未だ特定されるに至っていないことになる。このことは、民法 632 条により請負人であるベンダーの履行すべき債務（完成すべき仕事）の内容が抽象的となっていることを意味する。

(3) 開発完了と仕事の完成

ソフトウェア開発請負契約において、ベンダーの「開発完了」をもって「仕事の完成」と見ることができるかが問題となる。これはベンダーが契約通りに開発業務を遂行し、事実上、当該業務（全作業）が完了したことをもって、民法 415 条の「債務の本旨に従った履行（仕事の完成）」といえるかという問題である。

ソフトウェア開発請負契約を巡る裁判事例を見ても、ベンダーが開発完了したとしてユーザーに引き渡したソフトウェアについて、ユーザーから当該ソフトウェアは未だ完成していないと主張して法的紛争に発展するケースがある[5)]。つまり、開発すべきソフトウェアの内容に対するユーザーとベンダーそれぞれの捉え方・認識が乖離していることが大きな原因とい

える。

この場合、前項（2）で論述したとおり、契約締結時において一般的に、開発すべきソフトウェアの技術的内容が明確でなく、ベンダーの仕事の内容も抽象的となっていることが背景にある。特に、近年主流となっている Web 対応・分散処理型ソフトウェアの開発においては、プロトタイピングモデルという開発手法[6]が採用されることが多いが、この開発手法は、契約段階では不明確・抽象的であることを前提に、開発プロセスの中でユーザーを参画させることにより明確化・具体化をする方法であるから、この開発手法が近年のソフトウェア開発の主流となっているのである。

また、実際のソフトウェア開発請負契約の取引実態を見ると、ユーザー・ベンダー間には以下のような事情が介在していることが多いので留意しなければならない。

① 契約締結時点で、ユーザーが要求するソフトウェアの内容は不明確・抽象的であることが多いが、これに加えて開発作業着手後においても、ユーザーからソフトウェアに対する内容の変更要求が行われることが多いこと。

② ユーザーからの不明確・抽象的なソフトウェア内容の要求に対して、ベンダーは自ら保有するソフトウェア開発技術力に照らして技術的に開発可能な範囲内でしか、これに応えることができないという制約があること。つまり、ユーザーが要求しているソフトウェアの内容を技術的観点から見ると開発困難な内容のものもあるが、ユーザーは、このような開発困難な内容のものまで要求する場合があること。

③ 以上に伴い、ユーザーの要求しているソフトウェア内容とベンダーが開発可能であるソフトウェア内容との間には乖離が生じやすいことになる。

一般的には、両当事者は開発完了と仕事の完成とが一致することを想定して契約する筈であるが、しばしば両者が一致せず法的紛争へと発展して

いる。このような問題が生じるのは、開発すべきソフトウェア内容に対するユーザーとベンダーそれぞれの認識に乖離があることに起因している場合が多いと思われる。そして、このソフトウェアの内容を具体明確化することは、ベンダーの仕事（債務）の内容を具体特定化することに繋がるのである。

なお、ベンダーは「開発完了をもって開発すべきソフトウェアは完成した」と主張する傾向にあるが、上記①～③の事情が介在している場合が多いことを考慮すると、必ずしも、開発完了（事実行為の終了）をもってベンダーの仕事の完成（債務の完全履行）とみることはできない場合が多いのではなかろうか。

(4) 仕様変更と債務不履行

ソフトウェア開発請負契約における請負人であるベンダーは、注文者であるユーザーの要求する内容のソフトウェアを開発完了（仕事を完成）させ、当該ソフトウェアを契約所定の納期までにユーザーに引き渡さなければならないことになる。しかし実際には、ベンダーが開発行為を着手した後、何らかの事情が介在する場合が多く、この結果、ベンダーは履行遅滞（債務不履行）に陥ることになる。

この履行遅滞を引き起こす事由には様々なものがあるが、代表的な事由の一つとして仕様変更がある。つまり、契約締結時にソフトウェア内容はシステム仕様書により特定されることになるが、このシステム仕様書に盛込まれた要求仕様（ユーザーが当初要求したソフトウェアに対する内容）が開発作業の途中で変更されることである。もちろん、このような契約締結時のシステム仕様書に対するユーザーからの仕様変更の要求が発生すると、変更の程度によっては始めから開発のやり直しとなる場合もあり得るほどベンダーの開発業務に重大な影響を及ぼす。この仕様変更の程度が小さい場合といえどもベンダーにとっては新たな作業が追加されることが多

いので、この結果、当初のスケジュール通りに開発を進めることができず債務不履行に陥りやすくなる。

ソフトウェア開発請負契約においては、このような仕様変更は頻繁に生じやすい。この背景として、当初、ユーザーは開発すべきソフトウェア内容に対して抽象的なイメージ（概念）で捉えているが、開発作業が進みソフトウェア内容が徐々に具体明確化してくると、「このような処理もしたい」などと新たな要求が出るのが通常であるからである。特に、近年は絵やアイコンなど GUI（グラフィカル・ユーザー・インタフェース）を利用した画面処理のソフトウェアが主流となっている。このようなソフトウェアがプロトタイピングモデルにより開発される場合、開発中の処理画面についてのユーザーによる確認作業は不可欠であり、この確認作業の結果、ユーザーから仕様変更に係るような要求が出されやすいことになる。

このような要求がユーザーから出されると、ベンダーはそれが仕様変更に係るものかどうかを素早く見極めて、速やかにその旨をユーザーに申出なければならない。なぜなら、ユーザーは、その要求をソフトウェアの簡単な修正（プログラムの簡単な改変）で済むのではないかという軽い気持ちで行なう場合が多いからである。つまり、当初の要求仕様の変更には当たらないと一般的に考えているのである。

しかし、このユーザーからの仕様変更の要求の中には、ソフトウェア内容や開発工数を大幅に変更しなければならないような機能追加等が比較的多く含まれる場合もあるが、この場合は、当然ながら開発スケジュールの伸長や開発費用（契約金額）の増額に直結する。そこで、この仕様変更を行うかどうかについて速やかにベンダーはユーザーと協議して決定しなければならない。

この仕様変更を直接争った裁判事例は見当たらないが、機能追加等により当初想定したプログラムのステップ数が大幅に超えたため、これが当初の委託代金の範囲内かどうかが争われた判例がある[7)]。この裁判では、商

法512条に基づく報酬請求権（委託代金の増額変更）が争われたものであるが、裁判所はこれを認めなかった。なお、商法512条は「商人カ其営業ノ範囲内ニ於テ他人ノ為メニ或行為ヲ為シタルトキハ相当ノ報酬ヲ請求スルコトヲ得」と規定し、この規定は、民法の委任、寄託、事務管理等が無償とされていることの例外規定とされているが、企業取引は委任、寄託等の契約も有償で行なわれるので、当然のことを規定しているに過ぎない。要するに、報酬（契約金額）増額変更をしょうとする場合は、改めて合意をし直さなければ、その法的効力は生じないということである。

このことから、機能追加等の仕様変更により開発工数が大幅に増えて契約金額も増額変更しなければならないような場合には、上記の仕様変更を行なうかについての協議・決定をする際、併せて、当初の契約金額や開発スケジュールなどの契約内容についての変更の合意を改めて行っておく必要がある。つまり、この変更合意がなされない限り、仕様変更による機能追加等の作業も当初の契約の範囲内とされる可能性があり、ベンダーは、当初の納期までにソフトウェアを完成できないときは債務不履行の責任を負わされることになる。

(5) 検収の意義

検収とは、企業取引契約実務で一般的に使用されている用語であり、ソフトウェア開発請負契約においては、ベンダーが開発完了したソフトウェアをユーザーに納入した際、ユーザーがこのソフトウェアが要求通りのものであるかを検査することである。また契約実務では、ユーザーがこの検査を行い合格（要求通りのもの）と認めることを検収完了という。そして、請負契約の対価である報酬支払時期について民法633条は「報酬は、仕事の目的物の引渡しと同時に、支払わなければならない。」と規定しているが、契約実務では、この検収完了をもってソフトウェア（仕事の目的物）が完全に引き渡されたとみなすのが、ほぼ取引慣行となっている。つまり、

この検収完了をもって、ベンダーが債務を完全履行したことをユーザーが認めたことになる。そこで、契約実務では、契約対価支払計算期間の始期(契約条項例としては「甲は、検収完了の時から○日以内に本契約の対価を乙に支払うものとする。」)、目的物に関する権利等の移転時期（契約条項例としては「検収完了の時をもって、目的物の所有権は乙から甲に移転する。」）などのように、この検収完了を各種の法律効果を生じさせる基準時点として重視している。

この取引慣行を受けて、ソフトウェア開発請負契約においても、契約書には検収に関する条項が必ずといっていいほど盛り込まれることになる。この検収条項では、検収をユーザーの義務として、検収方法、検収期間、検収完了基準、検査不合格の取扱い等について具体的に規定するのが通例である。

ところで、民法は、売買契約において買主に対して目的物の検査義務を規定していないが、商法 562 条では、商人間売買における買主の検査義務として「商人間ノ売買ニ於テ買主カ其目的物ヲ受取リタルトキハ遅滞ナク之ヲ検査シ……」と規定している。商法は、商人間売買では特に取引の安全性と迅速性が要求されていることから、このような規定を設けているのである。なお、請負契約において仕事の目的物がある場合は、前述の通り、民法 633 条は注文者（ユーザー）の報酬支払時期について規定するだけで、仕事の目的物の検査義務について何ら規定していない。

企業間取引としてのソフトウェア開発請負契約でも、注文者であるユーザーに対して目的物であるソフトウェアについて検収義務を負わせているのは、商法 562 条が規定したのと同じ趣旨（取引の安全性と迅速性を目的）から慣行化したものとみることができる。

そして、ベンダーが開発完了したソフトウェアを納入することをユーザーが拒んだり、契約所定の期間内に検収を行わなかったりすると、ユーザーは民法 413 条の受領遅滞の責任を負うべきで解するのが妥当であろう。

(6)　ユーザーの開発作業への参画と義務

一般的に、契約当事者は対等な立場であることを前提に、契約自由の原則に基づいて、自由に相手方を選択し、契約内容を自由に決定している。しかし、事業者と消費者間の契約（消費者契約）においては、一般的に消費者は事業者に比べ情報・知識が不足するなど弱い立場にあるので、消費者保護のため特別法により契約自由の原則に対する制限を行なっている。中でも、消費者の情報・知識不足を利用して著しく不当な契約が結ばれやすいことに着目して、公序良俗違反、錯誤・詐欺・脅迫、契約締結上の過失などの法理の活用が図られている。そして、近年は事業者の説明義務・情報提供義務について議論されている[8)]。

一方、企業間取引における契約の当事者の関係を見ると、親事業者と下請事業者の関係のように、一般的には発注者側が強い立場にある。特に、当事者の相互信頼関係を前提としている請負、委任などの役務（労務提供）型の契約に、この傾向は強く表れる。

そして、企業間契約は有償・双務契約となっているので、契約当事者それぞれは、自らの債務を履行しなければならないことは当然であるが、この履行義務に加えて、相手方の債務履行に対する協力義務や秘密保持義務などの付加的義務を負わされているのが通常である。なお、これらの付加的義務を相互に負担し合うことは信義則から妥当といえよう[9)]。

企業間取引契約の中でもソフトウェア開発請負契約は、伝統的に行われている一般的な役務型の契約に比べ、一段と強固な当事者（ユーザー・ベンダー）間の相互信頼関係が求められている。これは、この契約の内容とその性質から当然に求められるものであるから、これまで事業者（ベンダー）の義務として論じられてきた義務のうち、注文者であるユーザーに対しても次のような義務を負担させるべきであろう。

まずは、情報提供義務である。これは、この契約がユーザーの重要な経営資源であるソフトウェアの開発（情報システムの構築）を目的としてい

るので、ユーザーが当該ソフトウェア開発に必要とされる十分なユーザー情報（各種資料・データ等）を提供しなければ、ベンダーは開発できないからである。このユーザーが提供する情報の中には、ユーザーの高度な機密情報（不正競争防止法で保護を受ける営業秘密[10]など）も含まる場合も当然あり得るので、このことからもこの契約がユーザーとベンダー間に強固な相互信頼関係がないと成立しないことがわかる。

次に、ベンダーの開発作業に対するユーザーの協力義務がある。この協力義務については、コンピュータ売買契約の裁判事例でも買主の売主に協力すべき信義則上の義務として認められていることから、このような契約よりも相互信頼関係が強く求められるソフトウェア開発請負契約においては当然のユーザーの義務といえる[11]。すなわち、ユーザーはソフトウェア開発が円滑に進み、期待通りのソフトウェアが完成することが自らの経営・事業に貢献することになるので、ベンダーが行う開発作業の内容や進捗状況などに強い関心を持ち、必要に応じベンダーが行う開発作業に協力しなければならないことになる。

このユーザーの協力義務の内容については、従来と今日とでは大きく変わってきている。すなわち、ソフトウェア開発請負契約の仕事の目的物（開発すべきソフトウェア）は、従来は汎用コンピュータ用ソフトウェアが中心であったが、今日ではクライアント・サーバー・システム（CSS）用ソフトウェアに中心が移ってきている。そうすると、従来の汎用コンピュータ用ソフトウェア開発におけるユーザーの協力の内容や度合いは、ベンダーから要請された最小限の行為を行うこと、いわば消極的・受身的な内容の協力でよかったが、今日の CSS 用ソフトウェア（例えば、Web 対応・分散処理型ソフトウェア）をプロトタイピングモデルにより開発する場合においては、ユーザー自身が開発作業そのものに対し積極的に参画し、ベンダーの開発作業を進めやすいように協力することが求められるようになってきている。つまり、このような今日の Web 対応・分散処理型ソフ

トウェア開発においては、ユーザーがベンダーの開発作業（債務履行）に対して積極的に参画し協力しなければ開発作業そのものが円滑に進められないという特性をもっているのである。

ところで、開発作業は、ソフトウェア開発請負契約におけるベンダーが履行すべき債務（役務提供）そのものである。この開発作業にユーザーが参画するということは、どのような義務を具体的にユーザーに負わせることになるのであろうか。

例えば、プロトタイピングモデル[12)]による開発工程（開発作業の流れ）では、それぞれの工程ごとにユーザーによる確認・チェックが組み込まれており、ユーザーが契約所定の確認・チェックを契約所定の期間内に終了させないと次の工程に進めないことになるなど、このユーザーの確認・チェックは開発工程上重要なものである。そうすると、ユーザーがこれら所定の確認・チェックを確実に実施することを義務づけておかなければならない。

このような確認・チェックは、ベンダーの開発作業（債務履行）に対する協力という域を脱して、ユーザーが開発業務の一部をベンダーと共同あるいは分担して作業することにほかならない。このように開発業務の一部をユーザーに共同・分担作業させるためには、勿論、契約書にユーザーの義務として（ユーザーは共同・分担作業を実施しなければならない旨）規定しておかなければならないことになる。

なお、このようにユーザーも開発業務の一部を共同・分担するようなソフトウェア開発においては、ユーザー・ベンダー間の意見や認識に齟齬をきたしてはならず、常に、両者の意思統一が図られていることが絶対的に必要とされる。このような開発作業形態をとるソフトウェア開発請負契約は、ソフトウェア共同開発契約に近いものとなる。そこで契約実務では、ユーザーに対してもソフトウェア開発プロジェクト推進体制の整備・確立を義務づけている。具体的には、ユーザーはソフトウェア開発プロジェク

トの責任者や主任担当者を選任し、日常的な報告・連絡等は双方の主任担当者間で行わせること、また、開発作業に関し双方で協議解決すべき事項は連絡協議会を開催し、これに責任者や主任担当者などの関係者を出席させること、などを契約書に規定しておく必要がある[13)]。

そして、このようにユーザーの開発作業に対する参画義務がソフトウェア開発請負契約に規定された場合、当該開発作業はベンダーの単独行為ではなく、ユーザーとの共同行為の性質を有することになる。そうすると当該ソフトウェア開発請負契約は、ソフトウェア共同開発契約の性質を併せ持つことになる。そこで、ベンダーが契約通りソフトウェアが完成できなかった場合は、すべての責任をベンダーは負わなければならないのであろうか。

この問題についての裁判事例を見ると、ソフトウェア共同開発の締結を目指した協定に基づいて、当事者の一方が担当分の開発を終えたのに他方が担当分を開発できなかった場合には、契約締結上の過失に基づき、一方当事者は開発費用を損害として他方当事者に対して賠償請求を認めた判例[14)]がある。この裁判では、契約締結前の段階における問題であるので債務不履行として争われていないが、共同開発契約の性質を有する契約の債務不履行の問題について考える際の一つの参考となろう。

つまり、ソフトウェア開発請負契約においては、ベンダーが契約所定の期日までにソフトウェア開発を終了させ、目的物であるソフトウェア（開発成果）をユーザーに引き渡せなかったら、形式上、ベンダーは債務不履行の責任を負わなければならないことになる。しかし、ユーザーの開発作業への参画がベンダーの債務履行に大きな影響を及ぼすような、いわば共同開発の性質を併せ持つようなソフトウェア開発請負契約において、結果としてベンダーが債務不履に陥った場合を考えると、それを引き起こした原因事由がベンダー側にある場合（民法 415 条でいう「債務者の責めに帰すべき事由によって履行をすることができなくなったとき」）だけに限ら

ない筈である。ユーザーに帰責事由がある場合や双方に帰責事由がある場合も十分考えられる。

従って、ソフトウェア開発請負契約における債務不履行の問題を処理する場合、当該契約におけるユーザーの開発作業への参画義務がどのような内容であるか、それが開発作業にどのように影響しているかなどを考慮して、債務不履行に至った原因事由がどこにあるかを究明することが必要である。

2. ソフトウェア開発と瑕疵担保責任

ソフトウェア開発請負契約における法的紛争の主要原因として、ソフトウェア（仕事の目的物）に対する瑕疵担保責任がある。請負契約の請負人には仕事の目的物につき担保責任が負わされているから、ベンダーはユーザーに引渡した（ユーザーが検収完了した）ソフトウェアにつき、後日、瑕疵が発見されたときは、先ずはこれを相当期間内に修補しなければならない（民法 634 条 1 項）。そして、ユーザーがこの瑕疵によって契約の目的を達成できないときは契約解除されても仕方がない（民法 635 条）。債務不履行が過失責任（債務者の帰責事由を要件）であるのに対し、瑕疵担保責任は無過失責任とされているが、請負契約では瑕疵のない仕事を完成することが契約上の義務であるから、この瑕疵担保責任は、債務不履行（不完全履行）の特則と解されている。また、請負契約の瑕疵担保責任に関する民法 634 条以下の規定は、民法 559 条により有償契約一般に準用されている売買の担保責任に関する規定の特則でもあるとされている[15)]。

ところで、通常、ソフトウェアの使用によって生じた情報システムの不具合は、当該ソフトウェアの瑕疵として担保責任の問題となる場合が多い。ここで、不具合とはソフトウェアの使用により情報システムが正常に稼動していない事象のことであり、誤作動したり、間違った処理結果が出力さ

れたりすることがその典型である。この不具合を引き起こす原因は、当該ソフトウェアだけに起因しているとは限らず、当該ソフトウェアが使用される情報システムの環境や構成状況（コンピュータ、各種ソフトウェア、ネットワーク等）との適合性・親和性に欠ける場合などにも不具合は生じる。例えば、データベース・サーバー（コンピュータ）のハードディスクの記録容量不足やネットワーク回線の通信速度不足など様々な原因により不具合が生じることもある。従って、不具合（という事象）が生じたからといって、ソフトウェアに瑕疵があると断定することはできない。しかし一般に、不具合の原因がソフトウェアの瑕疵にあることが圧倒的に多いことから、契約実務では、先ずはソフトウェアの瑕疵担保責任と問題として処理されることが多いのである。

しかし、「ソフトウェアの瑕疵とは何か」が明確にされていない。中でも、(1) で後述するようにプログラム・バグは、ベンダーが担保責任を負うべきソフトウェアの瑕疵といえるかについて、以下の通り、この結論を出すことを難しくしている背景・諸事情がある。

① ソフトウェア（正確にはプログラムであり、以下同じ）は、コンピュータで使うことにより所定の機能等を発揮するものであり、しかもアプリケーションソフトの場合、特定のコンピュータ（ハードウェア）及びオペレーティング・システム（OS）上でしか正常稼動しないという特性を持っていることである[16)]。つまり、当該ソフトウェアをユーザーにおける情報システムの使用環境（当該ソフトウェアが使用されているコンピュータや OS など）に適合させなければならないのである。

② 今日ではソフトウェア開発の生産性を向上させるため、開発ツールや高級プログラム言語などを駆使して開発することはもちろん、データベース用ソフト（DBMS）や通信用ソフトなどのパッケージソフトを部品として利用したり、場合によってはユーザーの既存プログラム

の一部を流用したりしている。このようにして開発されたソフトウェアを構成する個々のプログラム（ルーチンやモジュール）の内容は、必然的にブラックボックス化している。すなわち、開発を担当する技術者さえも実行ルーチンのコード・レベルでの詳細内容を完全に掌握できなくなっているのである。

③　前述したとおり、ソフトウェア開発請負契約におけるソフトウェアは、ユーザーの要求仕様と開発作業へのユーザーの参画（これらの要求仕様や参画の中にはユーザーの指示も含まれる）に基づき開発されているので、何らかの不具合（ソフトウェアの瑕疵を含む）は、これらユーザーの指図が原因となっている可能性もある。なお、このようにソフトウェアの瑕疵がユーザーの指図によって生じたときは、ベンダーには瑕疵修補義務がなく、かつ契約解除されることもない（民法636条）。

④　今日のCSS用ソフトウェアは、Web対応（インターネットなどのネットワークを介して処理）で、かつ多くのコンピュータ（各種サーバーや多数のクライアント）で構成された情報システムで使用されるので、当該ソフトウェアの使用中に何らかの不具合が生じたとしても、その不具合の原因が必ずしも当該ソフトウェアにあるとは限らないのである。

以上の背景・諸事情を踏まえ、ソフトウェア開発請負契約における瑕疵担保責任に関わる個々の問題を検討する。

(1)　プログラム・バグとソフトウェアの瑕疵

まず、ソフトウェアの瑕疵とは何であろうか。この問題を検討する前に、どうしても『プログラム・バグ（通常「バグ」と呼ばれている）[17]とは何か』について明らかにしておく必要がある。

バグ（bug）とは、「虫」とも呼ばれるコンピュータ・プログラムの誤

りであり、プログラムを人間（プログラマーと呼ばれるプログラムを作成する IT 技術者）が作成する以上、バグのまったくないプログラムを作成するのは極めて困難であるとされている[18)]。このため、ソフトウェア開発工程の中に、作成したプログラムが正常に動作するかについてのテスト（テスト用データを使用してプログラムを実行し、想定した結果が得られるかチェックする方法など）を必ず組み込み、このテストで発見されたバグを取り除く作業（「デバッグ（debugging)」と呼ばれる）によりプログラムを修正している。このデバッグ作業は、バグの発見や修正を支援する「デバッガ」と呼ばれるソフトウェアを使用して行われるのが普通である。

このようにテストやデバッグ作業はソフトウェア開発工程の一連の作業の中で行なわれているが、この作業を行なったとしても当該ソフトウェア（プログラム）からバグを完全に取り除くことができないこともあり得ると考えられる。

この理由として、以下の開発作業の実態をあげることができる。

① 今日のソフトウェア開発請負契約における開発期間は、従来の汎用コンピュータ用大規模システムの開発期間に比べてかなり短期間に設定されるのが普通であるので、このテストやデバッグ作業に十分な期間を取ることが難しくなっていること。

② 一般に、Web 対応・分散処理型ソフトウェアは、ベンダーが独自に開発するソフトウェア（プログラム）のほかパッケージソフトなど様々なソフトウェアをもって構成されているので、仮に、開発されたソフトウェアによって何らかの不具合が生じたとしても、その原因がベンダー独自開発のソフトウェアのバグ（プログラムの誤り）であるとは限らず、他のソフトウェアに原因がある場合には、当該ソフトウェアの開発者でなければ適切な修正ができないこと。

このようなバグは、はたしてソフトウェアの瑕疵といえるのだろうか。

ソフトウェアのバグについては様々な考え方がある。従来は、バグが存

在するソフトウェアは完成しておらず不完全履行とする考え方[19]とバグをソフトウェアの瑕疵とする考え方[20]とに大きく区分することができた。

しかし、今日のWeb対応・分散処理型ソフトウェアのバグについては、上記①および②の実態から、従来とは違った考え方が導き出すことができる[21]。すなわち、バグを完全に取り除いたソフトウェア（プログラム）を開発することは極めて困難であるという実態から、万一、ソフトウェアの機能に影響しないような軽微なバグが取り残されていたとしても、所定の（ユーザーの要求仕様を満たす）機能を発揮できるソフトウェアについては、完成したものとして取扱うこと、すなわち完全履行があったと考えるのが妥当であろう。もちろん、テストやデバッグ作業に十分な時間をかけて徹底的に行なえば、一定限度までバグを減らすことはできるかもしれないが、実際は、ユーザーとしては所定の機能を発揮できるソフトウェアであれば、それを早く使用したいと要求するのが通常であろう。そして、バグの中にはプログラムの誤りとして所定の機能を発揮できないという重大なものから、機能には影響しないが軽微な誤動作をするものまで色々あるので、ユーザーが当該ソフトウェアを使用していく中で不具合が発見されたときに、その原因がバグだと判明した場合に、ソフトウェアの隠れた瑕疵とみなしてデバッグ（修補）する方が合理的であるからである。

そこで、ソフトウェアの瑕疵や欠陥についての裁判事例を見ると、以下の通り、上述した考え方に沿った傾向にあるといえる。

まず、ソフトウェアを稼動させて所定の機能を発揮できない場合、すなわち契約の目的を達成できない場合にだけ、債務不履行（不完全履行）や瑕疵担保責任を認めている判例がいくつか見受けられることである[22]。

しかし一方では、バグを原因とする不具合以外にはプログラムの欠陥を原因とする不具合は発現しないことを認めた上で、不具合発生後遅滞なく補修を終え、ユーザーと協議の上相当と認める代替措置を講じた場合は、ソフトウェアの欠陥と評価することはできないとして、損害賠償責任を否

定している判例もある[23)]。

つまり、裁判所は、ソフトウェア（プログラム）を使用することによって情報システムに不具合が見つかった場合、その不具合の原因はプログラムの欠陥すなわちバグとみなすが、一方では、バグがあった場合においても、すぐに修補や代替措置がとられたならば、ソフトウェアに瑕疵や欠陥に該当しないと判断していることが伺えるのである。

(2)　2000 年問題とソフトウェアの瑕疵担保責任

1998 年前後をピークに、世界中で 2000 年問題が大きな社会問題となった。この 2000 年問題とは、当時の多くの汎用コンピュータが内臓時計により年号を下（しも）2 桁で管理していたため、西暦 2000 年を 1900 年と誤認してしまい正常な処理を続行できなくなることによって、社会的大混乱が起きるのではないかというものであった。つまり、ソフトウェア（プログラム）で処理する日付データを下 2 桁で設定していると、2000 年の下 2 桁の日付データ「00」を 1900 年と誤認して処理してしまうことになる。また、当時のソフトウェアの中には「00」のコードを特殊な意味を持つコードと定義しているものもあったので、2000 年になった瞬間、どのような動作をするか予想できないというものであった。

この 2000 年問題は、各企業や官庁等の準備対策が功を奏し、西暦 2000 年を迎えても、小さなトラブルは頻発したが、社会に大きな影響を与えるような大規模の事故等は生じなかった。この準備対策の一つとして、2000 年問題を引き起こすおそれのあるソフトウェアについて、日付データを下 2 桁から 4 桁に設定しなおすなどの修正作業が行なわれたが、この修正作業には多大な労力・時間・費用を要するものであった。

そこで、当時はこの多大な負担をユーザーが負うべきか、それとも当該ソフトウェアの開発者であるベンダーが負うべきか、という問題について様々な議論が交わされた。ベンダーが負うべきという主張の主たる根拠は、

2000年問題に対応していないこと（日付データを下2桁に設定したこと）がプログラムの設計・製造ミス、すなわちソフトウェアの欠陥・瑕疵に該当するから、ベンダーは債務不履行ないしは瑕疵担保の責任を負うべきだということであった。

しかし、この主張には幾つかの無理がある。

第一に、ユーザーは当該ソフトウェアを使って現に正常処理しているのであり、2000年問題未対応の部分が含まれ、かつ、それが原因で2000年を迎えたときに誤作動するかもしれないというだけで、それをソフトウェアの欠陥や瑕疵といえるかということである。つまり、当該ソフトウェアは不具合なく所定の機能を発揮しており、契約目的を達成しているのだから債務不履行や瑕疵担保責任の主張はできないと考えられる。

第二に、当時の多くの汎用コンピュータは、上述の通り、内臓時計により年号を下2桁で管理していたので、当該コンピュータで使用するソフトウェアを開発する際は、年号の日付データも下2桁で設定するのが普通であったことである。一般に、ソフトウェアは開発当時の技術水準に合わせて開発されるので、開発当時に年号の日付データを下2桁で設定するのが普通であったとするならば、ベンダーの設計・製造ミスとはいえず、ベンダーの債務不履行責任を追求することはできないからである。

第三に、民法上の瑕疵担保責任の存続期間は「仕事の目的物を引き渡した時から1年以内」（民法637条1項）であるが、この期間を契約で6ヶ月程度に短縮していることも多いので、事実上、この存続期間を経過している場合が多かったことが想定され、契約上、ベンダーの瑕疵担保責任を追及することはできないからである。

以上のことにより、この2000年問題を巡ってユーザー・ベンダー間で法的紛争までに発展したケースは、殆ど聞かれなかった。しかし、裁判事例を調べてみると、以下の通り1件の判例[24)]を見つけることができたので、この判例の概要を紹介する。

原告（ユーザー）の会計システム（第三者開発の一般会計パッケージとベンダーが実施権を有するサブ・システムから構成される）の構築及び使用許諾の契約を締結し、同システムを 2000 年問題に対応させるとの約定をしたにもかかわらず、被告（ベンダー）が対応させる補修をしなかったとして、請負契約又は保守契約の債務不履行もしくは瑕疵担保責任に基づき修補費用及びこれに対する遅延損害金をベンダーに求めた。

これに対し裁判所は、2000 年問題に対応させる合意及び保守契約が締結されていたことを認めず、また、ベンダーが実施権を譲渡したサブ・システムが 2000 年問題に対応していなかったことが、「通常有すべき性状を欠くものとは認めない」などと判断したうえで、ユーザーの請求はいずれも理由がないとして棄却した。

この判例も示している通り、ソフトウェアが 2000 年問題未対応であっても、そのこと自体だけでは、ソフトウェアの瑕疵担保責任をベンダーに負わせることは難しいといえる。

(3)　ソフトウェアの瑕疵と製造物責任

最後に、ソフトウェアの瑕疵と製造物責任との関係をここで考えてみる。

民法 709 条では不法行為による損害賠償責任は「故意又は過失」という要件を規定しているが、製造物責任法 3 条では「製造物の欠陥」によって生じた損害につき賠償の責任を負うと規定しているだけであるので、製造物責任は無過失責任であるとされている[25)]。

この製造物責任法は平成 6 年に不法行為特別法として制定された法律であるが、ここで問題となるのは、ソフトウェア（正確にはプログラム）は製造物であるかということと、ソフトウェアの不具合を引き起こすバグなどの瑕疵が、製造物責任法でいう欠陥に当たるかということである。

まず、ソフトウェアは製造物であるかということについて検討する。

製造物責任法では「製造物とは、製造又は加工された動産をいう。」（同

法２条１項）と規定しているが、ソフトウェア開発請負契約により開発されたソフトウェアを同法でいう製造物（動産）と見るのは無理がある。なぜなら、動産は不動産以外の有体物（民法 85・86 条）であるが、一般に、ソフトウェアは無体物として認識されるからである。もちろん、ソフトウェアがフロッピーディスクや CD-ROM などの記録媒体に記録された状態になったときは物として認識して取扱われるが、ソフトウェアの実体は記録媒体に記録された中身すなわちデジタル形式の情報であるから、情報であるソフトウェアは動産といえず、あくまでも無体物である。

しかし、この例外として、ソフトウェアが CPU[26]や ROM[27]などとして半導体チップに記録された形態（すなわちコンピュータの一部品）となれば、半導体チップと一体化しているので動産とみることができよう。例えば、今日では特定の機能を有するソフトウェアを ROM 化し、これを一部品として様々な情報機器、携帯電話、情報家電などに組み込んでいるが、このようなソフトウェアは動産として取扱ってよいと思われる。

次に、ソフトウェアの瑕疵は製造物責任法でいう欠陥に当たるのであろうか。

同法でいう「欠陥」とは、「当該製造物の特性、その通常予見される使用形態、その製造業者等が当該製造物を引き渡した時期その他の当該製造物に係る事情を考慮して、当該製造物が通常有すべき安全性を欠いていること」である（同法２条２項）。そして、ここで注意を要することは、同法が賠償責任を負わせている損害について「……欠陥により他人の生命、身体又は財産を侵害したときは、これによって生じた損害を賠償する責めに任ずる。ただし、その損害が当該製造物についてのみ生じたときは、この限りでない。」（同法３条）と規定していることである。つまり、製造物に欠陥があるということは欠陥品であるので、修理したり、正常品と取り替えたりしなければならないが、このような製造物自体についての損害には同法は適用されないのである。

では、どのような場合にソフトウェアの瑕疵が製造物責任の問題となるのであろうか。

上述したように ROM 化されたソフトウェア（例えば、エンジン制御プログラム）を組み込んだ製造物（自動車）が暴走し通行人に怪我させたと仮定した場合、その暴走の原因が自動車（部品として組み込んだ当該ソフトウェア）の欠陥にあったとすると、自動車メーカーは製造物責任を負わなければならないことになる。

このような場合、当該ソフトウェアを開発したベンダーも製造物責任を負わなければならないのであろうか。通常、エンジン制御プログラムのような自動車専門技術を要するソフトウェアの設計は自動車メーカーが行なうことになろう。そうすると、ベンダーは当該自動車メーカーの「設計に関する指示」に従って当該ソフトウェアを開発することになる。このようにして開発されたソフトウェアに欠陥があったとしても、同法 4 条 1 項 2 号の免責事由に該当し、ベンダーは製造物責任を負わなくてよいと解することができる。

以上のことから、ソフトウェアの瑕疵は、ソフトウェアが動産に該当するような特別な使われ方をする場合を除いて、製造物責任の問題は生じることがない。また、Web 対応・分散処理型のアプリケーションソフトの開発請負契約においては、ソフトウェアの瑕疵問題は契約上の債務不履行や瑕疵担保責任の問題として処理すれば十分であり、製造物責任問題として検討する余地は殆どないといえる。

3. 債務不履行に関連する契約問題

ソフトウェア開発請負契約で問題となりやすいのは、これまで述べてきたようにソフトウェアの完成（開発完了）と不具合（瑕疵）の問題である。ソフトウェアの完成の問題は債務不履行の問題として、講学上、履行遅滞、

履行不能及び不完全履行に区分して論じられ、不具合の問題は瑕疵担保責任の問題として論じられている。

つまり、ソフトウェア開発請負契約において、ベンダーは「債務の本旨に従った履行」をしなければ債務不履行となり損害賠償の責任を負わされることになる（民法 415 条）。もし、ベンダーが開発を請け負ったソフトウェアの完成が遅れ、契約所定の期限までに開発完了（完成）したソフトウェアをユーザーに引き渡すことができなければ履行遅滞となり、ユーザーが要求していた内容通りのソフトウェアを開発できなければ履行不能となる。また、開発したソフトウェアが所定の機能を発揮できなければ不完全履行ないしは瑕疵担保責任の問題が生じるのである。そして、これらの債務不履行や瑕疵担保責任の問題が生じると、最終的にはユーザー・ベンダー間で契約解除や損害賠償等の法的紛争へと発展することになる。

ところが、一般に、このような債務不履行や瑕疵担保責任の問題が生じる原因を調べてみると、「責めに帰すべき事由（帰責事由）」が一方的にベンダー側にある場合は少なく、むしろ、ユーザー側にある場合や様々な背景や諸事情が存する場合も多いのである。

このようなソフトウェア開発請負契約に特有の背景・諸事情等が存することにより法的紛争の解決を困難にしているが、これらの法的紛争を予防・解決するために、契約実務上、様々な方策が講じられている。

そこで、以下にこれらの契約実務上の方策の幾つかを取り上げてみたい。

(1) ソフトウェアの瑕疵との関係

一般に、債務不履行のうち履行遅滞や履行不能の場合は、債務不履行の事実（債務の本旨に従った履行をしていないという事実）は客観的に見て明白となっている。しかし、債務不履行のうち不完全履行は、形式的には履行がされているが、債務の本旨に従った完全な履行でない場合をいうので、このような不完全履行の場合は、債務不履行の事実が客観的に明白で

ないのが通常である。

ソフトウェア開発請負契約においても不完全履行の事実は客観的に明白でないが、この要因として次の事由が考えられる。

① 契約締結時において、ベンダーの仕事の内容が明確に特定されないことが多いこと。（既述「1.（2）ベンダーの仕事の内容」参照）

② ユーザーとベンダーとの間にソフトウェア内容に対する認識に乖離があることが多いこと。（既述「1.（3）開発完了と仕事の完成」参照）

そして、契約実務では、仕事が完成したかどうかは、開発完了のソフトウェアをユーザーが検収（検査）することで確認しているので、完全履行か不完全履行かの判断は、このユーザーの検収（いわばユーザーの主観）に委ねられている。

なお、不完全履行を引渡債務の場合と行為債務の場合とに分け、更に、引渡債務の場合の態様を瑕疵型と拡大損害型とに分けて検討する方法[28)]があるが、この方法に従えば、ソフトウェア開発請負契約における不完全履行は、引渡債務の瑕疵型に該当し、追完可能な不完全履行として処理できる。

ところで、ソフトウェアにはバグが付きものであり、このバグがソフトウェアの不具合を引き起こすが、この不具合の内容・程度によってソフトウェアの瑕疵や欠陥に該当するかを判断すべきである。（既述「2.（1）プログラム・バグとソフトウェアの瑕疵」参照）

そして、このことを前提に契約実務も、ユーザーは、ベンダーから開発完了したソフトウェアの納入を受けたら一定期間内に遅滞なく検収（検査）を終えることをユーザーに義務付けるのが通例となっている。つまり、ユーザーは、この検収（当該ソフトウェアを実際に稼動させて不具合がないかどうかの確認）を行い、この結果、不具合が見つかれば、ベンダーに対して不完全履行として当該ソフトウェアの補修（追完）の要求をするこ

とになる。

しかし、実際には、ユーザーがこの検収期間中にソフトウェアの不具合を見つけることができず検収完了（検査合格として目的物＝ソフトウェアの引渡完了）したにもかかわらず、契約終了後（当該ソフトウェアを本格稼動している中で）不具合を見つける場合も多い。このようにユーザーが検収完了後に不具合を見つけた場合、契約実務では、その不具合をソフトウェアの隠れた瑕疵とみなして、ベンダーは担保責任を負うとするのが通例である。ただし、この担保期間は、民法 637 条 1 項の「引渡し時から 1 年以内」ではなく、「引渡し時から 6 ヶ月」などのように短縮する場合が多い[29)]。

なお、「本件で主張されているコンピュータ・システムの欠陥は、たとえば、回線切断、画面表示、入力方法、機密保護など、いずれも一般にコンピュータ・システムを構築・運営する上で通常生起することが予想される類型のものである」として債務不履行を否定した裁判事例[30)]もあるとおり、債務不履行（不完全履行）と瑕疵担保責任とは区分できないことが多いと考えられる。

このような事情を背景にソフトウェア開発請負の契約実務においては、不完全履行と瑕疵担保責任の問題は、一連の関連した問題として処理されている。つまり、ソフトウェアの不具合の発見が検収完了の前後によって処理の仕方が異なるだけである。すなわち、検収完了前に不具合が見つかった場合は不完全履行として追完により処理し、検収完了後に見つかった場合は瑕疵担保責任として修補請求により処理しているのである。

(2) ソフトウェアの保証との関係

次に、ソフトウェアの保証と債務不履行との関係について検討する。

保証という言葉は、講学上、民法 446 条以下の人的担保（保証人）を意味するが、ソフトウェア開発請負契約等 IT 関連の契約実務では、次の意

味合いを組み合わせて多義的に使用されている[31)]。

① 当該債務を完全に履行することを誓約すること。これは、一般に「履行保証」と呼ばれている。

② 目的物が第三者の権利を一切侵害していないことを請合うこと。これは、一般に「権利保証」と呼ばれている。

③ 目的物に傷や不具合等の異常がないことを請合うこと。これは、一般に「品質保証」と呼ばれている。

④ 納入後、一定期間であれば正常稼動を請合うこと。これは、一般に「アフターサービス」と呼ばれている。

以上の意味合いを持つ保証は、いずれも契約当事者が約定しなければベンダーの法的義務としての効果が生じない。そして、保証された内容が実現されなければ、その内容の性質により債務不履行ないしは瑕疵担保責任の問題として処理されることになる。上記の保証のうち、特に、ソフトウェア開発請負契約実務において問題となりやすいのは、②の権利保証と③の品質保証である。

なぜなら、①の履行保証は、ベンダーの履行義務そのもので、これを保証するしないにかかわらず法的効果は変わらないからである。しかし、ベンダーの履行に長期間を要するソフトウェア開発請負契約では、ユーザーとしては「ベンダーが途中でやめないで、きちんとソフトウェアを開発してくれるだろうか」などの不安を抱くことはもっともであり、この不安を解消するためにも契約実務では履行保証の規定を盛り込むことが多い。なお、この履行保証に似ている問題として、ユーザーは「ベンダーが倒産等によりソフトウェア開発が頓挫するのではないか」という不安を抱くこともある。この不安に対しては、ソフトウェア完成を担保する制度として映画製作等で利用されている「完成保証」[32)]の利用が考えられる。しかし、ソフトウェア開発請負契約が完成保証会社（第三者）の介入に馴染めない（完成保証を利用できない）という性質を考慮すると、ユーザーはベンダ

ー選定時に、開発実績や経営状況等をよく吟味し、信頼できるベンダーを選定することで、この不安を解消すべきであろう。

また、④のアフターサービスの内容は瑕疵担保責任（通常、当該ソフトウェアのメーカーが行なうので、ベンダーがこれを行なうとは限らない）と似ているが、ソフトウェア開発請負契約においては、別途ソフトウェア保守契約を締結して一定期間内は無償で（期間経過後は有償で）保守する方法を採るのが通例となっているからである。

従って、ソフトウェア開発請負契約の交渉においては、ベンダーが②の権利保証や③の品質保証を行なうかどうかを巡って、ユーザー・ベンダー間で意見が対立しやすい。つまり、当該ソフトウェアについて、ユーザーは両方の保証をベンダーに求めるのが通常であるのに対して、ベンダーはこのような保証を行なえば契約上の義務負担が大きくなるので、これをできるだけ回避したいからである。

それでは、それぞれの保証について更に検討を進める。

まず、権利保証は、ベンダーが開発するソフトウェアについて、正当な権限に基づき開発を行ない、第三者の権利（主として特許権、著作権）を侵害していないことを求める保証である。この保証の内容で重要なことは「第三者の権利を一切侵害していない」ということであり、このような保証は「権利非侵害保証」と呼ばれることもある。

この権利非侵害保証は、一見、当然のことのように見えるが、ソフトウェアについては主たる権利が特許権や著作権といった知的財産権であるため、現実には、この保証を行なうことを著しく困難にしている事情がある。

つまり、動産（有体物）の物権（所有権等）については、その権利関係等が客観的に明白となっているので、権利処理や権利保証も比較的に行ないやすい。これに対して、ソフトウェア等情報（無体物）に係わる知的財産権は、その存在や誰が権利者かなどといった事実関係や状況が客観的に明白でないことが多く、これを調査しても正確に把握できない場合もある

からである。

例えば、今日では多数のソフトウェア特許が出願されているが、ベンダーが開発しようとするソフトウェアに関連する最新の特許権の状況や権利関係を調べようとしても、特許出願から 1 年 6 月を経過しなければ公開されない（特許法 64 条 1 項）し、公開された特許出願の全てが特許権を付与されるとは限らないので、正確な事実や状況を把握できないのである。更に、Web 対応・分散処理型ソフトウェアの開発に当たっては、フリーソフト等多種の第三者ソフトウェアを利用するのが当たり前となっていることから、これに伴い第三者の権利侵害の危険性も急速に高まっている。

このような事情は、ベンダーがどんなに努力してもコントロールが及ばない領域のものであり、このような事情を背景とする限り、権利非侵害保証の実現可能性は極めて低いといわざるを得ない。そこで、ベンダーは権利非侵害保証を回避したがるのであり、実際にも、このような保証を規定している契約例を殆ど見ることができない。

ところで、ソフトウェア開発請負契約においてベンダーは、ソフトウェア開発の専門家として細心の注意を払いながら仕事を完成させなければならないことは当然である。このため、ベンダーは独自開発手法[33)]を採用するなどにより第三者の著作権を侵害しないよう権利非侵害対応策を講じるのが通常である。このように第三者の権利を侵害しないように開発したにもかかわらず、開発完了したソフトウェアが第三者の権利を侵害していることが判明する場合もあり得る。この場合、それを権利の瑕疵とみなし、ベンダーは瑕疵担保責任を負うとされる可能性が高い。（なお、ソフトウェア開発途中で第三者の権利侵害の事実が判明した場合は、ベンダーは侵害とならない状態にするための対応を当然ながら講じることになるので、債務不履行の問題まで発展する可能性は殆どない。）

このように権利侵害を担保責任の問題と捉えた場合、民法では、売買については、560 条以下に他人の権利を売買の目的とした場合など「権利の

瑕疵」についての担保責任を規定しているが、請負については権利の瑕疵に関する規定がない。しかしながら、ソフトウェアに権利の瑕疵がある場合の担保責任については、民法559条により売買についての561条以下の規定を準用するのが妥当といえよう。そして、ベンダーが開発したソフトウェアが第三者の特許権や著作権を侵害し、ユーザーが当該ソフトウェアを使用できない状態である場合には、民法635条の契約の目的を達することができない状態といえるから、ユーザーは契約を解除できるものと解するのが相当であろう。

一般に、権利者は侵害ソフトウェアの使用差止と損害賠償の請求を行なう場合が多いが、このような請求を受けた場合の対応は容易ではない。そして、通常、この対応のため多大な時間、労力、費用がかかる場合が多い。特に、ソフトウェア開発完了後に権利侵害、とりわけ特許権侵害の事実が判明した場合、当該特許を実施したソフトウェアを現に使用しているユーザーに対しても使用差止請求ができるため、当該ソフトウェアの使用差止請求を受けたユーザーにとっては、ユーザーの事業継続にも多大な影響を及ぼす可能性もある。

このような場合を想定し、契約実務上、権利侵害した場合の対応が非常に重要となる。そこで、ベンダーは権利保証を行なわない場合であっても、当該ソフトウェアが第三者の権利を侵害していることを理由として、当該第三者からユーザーが何らかの請求を受けた場合についてのベンダーの責任（当該紛争処理や損害賠償）について契約書に規定することが多い。ベンダーがこのような責任を負うことで、実質的には権利保証を行なっているのである。

次に、品質保証は、契約実務上、瑕疵担保責任として行なわれることが多い。一般に、品質保証という場合、例えば「このテレビはきめ細かくきれいに映りますよ」「簡単に故障しませんよ」という内容の保証を指すが、ソフトウェアについての品質保証は、性能保証と機能保証とに分けて、契

約実務上、両者を使い分けている。すなわち、性能保証とは「このソフトウェアは○○件のデータを○分以内に処理できますよ」というような内容の保証であり、機能保証は「当該ソフトウェアが所定の機能（例えば、文書作成、表計算など）をきちんと発揮しますよ」というような内容である。

このうち、機能保証は、契約目的の実現に関する保証と解することができるので、この保証を契約上明確に規定していなくても、これはソフトウェア開発の性質からベンダーが当然に負うべき債務の内容といえる。つまり、当該ソフトウェアが所定の機能を発揮できなければ、契約目的が実現できず債務不履行となるからである。しかし、性能保証は、当該ソフトウェアが所定の機能を発揮することを前提に、処理スピードが速い（例えば、○○件の処理を○分以内に行なうことができる）などの性能に関することであるので、契約上規定しなければベンダーの債務とはならない。そして、性能は当該ソフトウェアの使用環境等様々な要因（ハードウェア、通信回線、OS その他のソフトウェアなどの特性）に左右されるので、その実現が不確実とされている。このように実現不確実な内容を債務をすることは、ベンダーとして債務不履行の危険が増すため、契約実務上も性能保証は殆ど行われることはない。

しかし、現実には機能と性能とは表裏一体の関係があり、どこまでが機能でどこから性能なのか峻別することは難しい。そこで、契約実務上は、不具合（その殆どが機能不全である）が発現した場合、これをソフトウェアの瑕疵とみなして、ベンダーの瑕疵担保責任として処理するのが通例となっている。

なお、ソフトウェアの機能ないし性能の不全（欠陥）に関する裁判事例を見ると、裁判所は「基幹業務システムのソフトウェア製作（開発）において、当該ソフトウェアは同種のソフトウェアと比較して同等もしくはこれに近い機能、性能を有すべきところ、右機能ないし性能を有しておらず種々の欠陥があったことは、ソフトウェア製作者として自らが有する高度

の専門的知識経験に基づき右目的の実現に努めるべき債務を負うとして、機能を欠く債務不履行の部分がある」と認定した判例がある[34]。このことから、ベンダーが開発するソフトウェアにつき性能保証を行なわない場合といえども、当該ソフトウェアが同種ソフトウェアと同程度の機能・性能を発揮しなければ、ベンダーは債務不履行や瑕疵担保責任を負わされる可能性が大きい。

(3) 契約解除との関係

一般に、ソフトウェア開発請負契約においても、ベンダーが債務不履行となれば、ユーザーは契約解除をすることができる（民法 541 条・543 条）。また、ソフトウェアに重大な瑕疵があり契約目的を達することができないときも同様である（同法 635 条）。

しかし、実際の取引契約実務を見ると、これまで論述してきたようなソフトウェア開発請負特有の性質、背景・事情などを受けて、契約解除はそう簡単に行なわれることはない。つまり、ユーザー・ベンダー間の信頼関係が完全に破綻したような場合やベンダーが履行不能に陥った場合でないと、ユーザーは契約解除を行なわないのが通常である。

この第一の理由としては、債務不履行に至った原因が必ずしもベンダーだけに帰責事由がある場合だけに限らず、契約解除の要件を満たさないことも多いからである。また、ソフトウェアの瑕疵については、ユーザー（注文者）が提供した材料の性質やユーザーがベンダーに与えた指図によって生じた場合は担保責任の規定が適用されないからである（民法 636 条）。

裁判事例を見ても契約解除を認めたものは少ない。すなわち、ベンダーの債務不履行については、履行不能の場合（ソフトウェア開発を継続する意思がなく、客観的に履行不能であるとき）[35]だけであり、また、瑕疵担保責任については、契約目的を達せないほどの重大な瑕疵があった場合

（ソフトウェアに不具合が多く使い物にならない状態）[36]だけである。

第二の理由としては、一般に、契約が解除されると、民法 545 条によりユーザー・ベンダーは共に原状回復義務を負うことになるが、特にソフトウェア開発請負契約では、実際には原状回復が困難な場合が多いので、ユーザーはできるだけこれを避けたいからである。そして、原状回復は契約締結前の状態に戻すことであるから、ユーザーにとっては、もう少しでソフトウェアの開発完了（ベンダーの完全履行）が見込まれる状態であっても、当該ソフトウェアを受領できない（一切使用できない）ことを意味することになる。そこで、ベンダーに多少の債務不履行（開発の遅れ等）が生じたり、ソフトウェアに多少の瑕疵（不具合）が見つかったとしても、履行期（開発完了・納入時期）を伸ばしたり、追完請求ないしは修補請求を行なうことで当該ソフトウェアが使える状態になれば契約目的を達することができる。これに対し、契約を解除してしまうと、ユーザーは別のベンダーを探してソフトウェア開発をやり直さなければならず、このための時間・費用等で大きな負担となる。

いずれにせよ、このようなベンダーの債務不履行やソフトウェアの瑕疵によりユーザーに損害が生じた場合であっても、上述したような特段の場合（履行不能やソフトウェアが使い物にならない場合）を除いて契約解除は行なわずに、損害賠償問題として処理すれば済むことである。

(4)　損害賠償との関係

ソフトウェア開発請負契約において、ユーザーは、ベンダーの債務不履行やソフトウェアの瑕疵によって蒙った損害については、ベンダーに対して損害賠償の請求をすることができる（民法 545 条 3 項・634 条 2 項）。

なお、ベンダーの不法行為や不当利得などによりユーザーが損害を蒙った場合も、ユーザーはベンダーに対して損害賠償の請求をすることができる（民法 704 条・709 条）が、ここでは、このような場合を除外して検討

する。

前項（3）でも述べた通り、ソフトウェア開発請負契約においては、実際には、債務不履行問題やソフトウェアの瑕疵問題の解決は、損害賠償問題として最終的に解決される場合が殆どである。このことは、本稿で引用した幾つかの裁判事例を見ても、その殆どが損害賠償請求事件となっていることからも裏付けることができる。

ところで、債務不履行による損害賠償について民法は 415 条～422 条に詳細な規定を置いているが、一般に、損害賠償を検討する場合、どの範囲までの損害をベンダーに負担させるかという損害賠償の範囲について一番問題となりやすい。

この損害賠償の範囲については、民法 416 条が 1 項で「債務の不履行に対する損害賠償の請求は、これによって通常生ずべき損害の賠償をさせることをその目的とする」と原則を規定し、同条 2 項で「特別の事情によって生じた損害であっても、当事者がその事情を予見し、又は予見することができたときは、債権者は、その賠償を請求することができる」と規定している。

そして、請負人の瑕疵担保責任による損害賠償について、民法 634 条 2 項は「注文者は、瑕疵の修補に代えて、又はその修補とともに、損害賠償の請求をすることができる。この場合は第 533 条の規定を準用する。」と規定し、請負人に対し損害賠償責任を認めているが、この損害賠償の範囲について、民法は特段の規定を置いていない。また、瑕疵担保責任による損害賠償の範囲は、一般に履行利益（瑕疵があるために生じた目的物の経済的価値の減少に対する損害）に及ぶとされている[37)]。

しかし一方では、民法 636 条本文は「前二条の規定は、仕事の目的物の瑕疵が注文者の供した材料の性質又は注文者の与えた指図によって生じたときは、適用しない。」と規定し、同条ただし書は「ただし、請負人がその材料又は指図が不適当であることを知りながら告げなかったときは、こ

の限りでない。」と規定している。

実際のソフトウェア開発請負契約においては、ユーザーが材料（例えば、ユーザーの前システムのプログラムやコンテンツなど）や指示（例えば、入出力の画面・帳票などに関するデザインやデータの桁数など）をベンダーに与える場合が頻繁に行なわれている。このようなユーザー（注文者）の提供した材料や指図によって当該ソフトウェアに瑕疵が生じた場合には、民法 634 条 2 項は適用されず、ユーザーは損害賠償の請求をすることができないことになる。

そして、裁判事例[38]を見ると、民法 636 条本文の規定によりベンダーの瑕疵担保責任を否定した判例があるので、この概要を以下簡単に紹介しておく。

A 社（原告）は、C 社（訴外）が開発したインターネット上で顧客管理を行なうアプリケーションソフトを用いたサービス提供業務を C 社の代理店として行っていたが、A 社がこの契約に違反したため C 社から契約解除された。そこで、A 社は B 社（被告）に対し、C 社のソフトと同機能のソフト製作を依頼し、B 社は A 社の指図どおりにソフトを製作し納入した。ところが、被告 B 社が製作したソフトは、C 社のソフトのプログラム及びユーザーインターフェース画面を無断で複製又は改変したものであったので、原告 A 社は C 社から著作権侵害の警告を受けた。そこで、A 社は B 社に対し、当該ソフトは法律上の瑕疵があると主張し、瑕疵担保責任に基づく損害賠償を請求した。

これに対し、裁判所は、原告 A 社の主張は失当であるのみならず、仮に法律上の瑕疵があるとしても、被告 B 社は A 社の指図どおりに製作したものであるから、その瑕疵は「注文者の与えた指図により生じた」ということができ、被告 B 社は民法 636 条本文の規定により、瑕疵担保責任を負わないと判示した。

この判例は、第三者の権利侵害をしているソフトウェアは「法律上の瑕

疵」があると主張した点とその瑕疵が「注文者の与えた指図により生じた」と裁判所が民法 636 条本文の規定に基づき瑕疵担保責任を否定した点が、他の判例にない特徴である。

その他のソフトウェア開発請負契約におけるベンダーの瑕疵担保責任に基づく損害賠償請求に関する裁判事例を見ると、民法 635 条本文が規定する「契約をした目的を達することができない」というようなソフトウェアに重大な瑕疵がある場合[39]を除き、損害賠償請求が棄却されている[40]。いずれにせよ、これらの判例からは瑕疵担保責任に基づく損害賠償の範囲は明らかにされていない。

そこで契約実務上、債務不履行による損害賠償の範囲が問題となりやすいので、この問題について、もう少し検討する。

判例・通説は、民法 416 条を「相当因果関係」の原則を定めた規定であるとし、不法行為にもこの規定の適用を認めている[41]。しかし、ソフトウェア開発請負契約の実務においては、民法 416 条 1 項でいう「通常生ずべき損害（通常損害）」と同条 2 項でいう「特別の事情によって生じた損害（特別損害）」とはどのような損害を指すのか、更に、特別損害で要求される「当事者がその事情を予見し、又は予見することができた」という予見可能性とはどのようなことかが問題となる。

このような損害賠償の範囲の解釈については、平井宣雄教授は「金銭賠償主義を採る日本民法の下では損害は、実務上、保護範囲と金銭的な評価との区別が意識されず、また必ずしも常に意識される必要がない。……416 条の対象となるのは、契約解除によって買主がどのような不利益な地位にあるか（転売によって受ける利益を失ったとか、転買主へ違約金を支払う地位に置かれたとか）という問題であって、市価と契約価格との差額は、その不利益な地位を金銭に表示するための方法の一つにすぎないと考えるべきである。……保護範囲として画定される不利益の事実は、416 条によって予見可能だと認められる範囲である。予見可能性が存在しない場

合には、損害賠償の範囲は『通常生ずべき損害』に限定される。……右に述べたことは、予見可能性が賠償の範囲を決定する究極の要件となっていることの反面にすぎない。」[42]との見解を示されている。これに対し内田貴教授は「416 条が損害賠償の範囲についての原則を定めているが、この規定をめぐっては、判例の理解も含めて、長い論争の歴史がある。……通常損害と特別損害を区別する意味は、予見可能性を立証する必要性の有無にある。……しかし、通常損害と特別損害の判定は、具体的な事案において必ずしも容易なわけではない。……そこで問題は、何が通常損害で、何が特別損害かである。これは一般的に決まらず、契約類型ごとに、当事者は商人かどうか、目的物は何か、等を考慮して個別具体的に判断するほかないだろう。」と解されている[43]。

一方、ソフトウェア開発に関する損害賠償の範囲に関する裁判事例を見ると、ソフト開発の下請会社が作業時間を水増しした金額で元請会社がそのまま注文者に請求したことにより、元請会社は注文者から信用を失い元請契約を解除されたので、元請会社が取引機会を失ったことによる逸失利益につき下請会社に対し損害賠償請求をしたところ、相当因果関係が否定された判例がある[44]。そして、数多くの損害賠償請求事件についての裁判では、原告（通常はユーザーが多い）は損害額を幅広く見積もって通常損害と特別損害を区別せずに請求するのが一般的である。これに対し裁判所の判断は、仮に損害賠償を認める場合においても、原告の請求額と比較すると、裁判所が算定した賠償額（金銭的な評価）はかなり低い額となるのが通常である。このような実態となっているのは、原告として因果関係や予見可能性を含めた損害額についての立証が非常に難しいからだといえる。

結局、通常損害や特別損害がどのようなものかなど民法 416 条の解釈を議論しても実務上の利益はないものと思われる。むしろ、この損害賠償の範囲は、実際のソフトウェア開発請負契約の紛争案件ごとに判断されるべきである。

このような損害賠償の裁判実態などを受けて、取引契約実務においては、契約違反等により相手方に損害を与えた場合に備えて、裁判による損害賠償の範囲の立証の困難さを避けるため、あらかじめ契約書に違約金（損害賠償額の予定）条項を盛り込んでおくことが多い。この違約金条項は、民法 420 条 1 項前段により有効とされているからである。そして、違約金額は、契約金額の一部で設定されるのが一般的である。しかし、ソフトウェア開発請負契約においては、当該ソフトウェアがどのような目的に使われるかによって契約金額の数倍という非常に高額の損害が生じることも予想される。例えば、基幹業務用ソフトウェアについては、ユーザーがこれを使えない状態になると、基幹業務そのものの処理ができず、これによる損害は多大な額となることが容易に想定できる。そうすると、開発対象のソフトウェアの目的や用途に応じて、あらかじめ想定される損害額を見積もり、これを違約金とすることが考えられるが、この結果、余りにも高額の違約金となる場合は公序良俗違反として民法 90 条により無効とされよう。

違約金を巡る裁判事例としては、ソフトウェアの共同開発を途中で取りやめた場合について、共同開発者がこれまで行った全作業につき適正な評価をして相当と認められる金額の精算をすることを認めた判例[45)]があるが、ソフトウェア開発請負契約については、この判例はあまり参考とならない。

そこで、損害賠償に関し、契約実務上、何らかの方策が求められる。

ソフトウェア開発事業の業界団体である（社）情報サービス産業協会（JISA）は、平成 6 年 12 月に「ソフトウェア開発委託モデル契約書（第 3 版）」を公表し、更に、平成 14 年 5 月にも Web 対応・分散処理型ソフトウェアについての「ソフトウェア開発委託モデル契約書」を公表している。

この JISA モデル契約書の損害賠償条項を見ると「本契約の履行に関し、相手方の責に帰すべき事由により現実に被った通常の損害に限り、相手方に対して契約金額の限度内で損害賠償を請求することができる。」旨規定している[46)]。なぜ、このような条項にしたかについても、JISA は当該モ

デル契約における主要条項の策定趣旨をホームページに掲載している。これを見ると「ソフトウェア開発に関連して損害が発生する場面では、多数の関係者の副次的な要素が絡み合い、原因の特定や損害との因果関係の究明が困難であり、相互に莫大な額の損害を主張し、問題解決が長期化することが多いものと考えられる。この場合、合理的な範囲で賠償限度額を設定することにより、因果関係が稀薄な損害に関する賠償請求に見切りをつけ、問題解決を迅速に図ることは請求者にとっても決して不利益にはならない。なお、一般的に、取引関係に入る場合、契約金額を上回るリスクを想定することは極めて少ないと考えられるため、契約金額を損害賠償の限度額とすることは合理的と考えられる。また、同様の問題を抱える他業界においても、賠償限度額を設ける慣行が定着している。」と説明している[47)]。

この JISA モデル契約書は、ソフトウェア開発委託取引の業界約款ではないので、このように損害賠償の限度額を設定する方策は未だ取引慣行とはなっていない。しかし、このようなモデル契約書の普及は、ソフトウェア開発委託取引について適正な取引慣行を確立するために貢献するばかりか、損害賠償問題を含む種々の法的紛争の予防と早期解決にも役立つことが期待できる。

おわりに

本稿では、前稿「ソフトウェア開発委託契約紛争事例の研究（1)」で明らかにした法的紛争の主要争点、すなわちソフトウェア開発請負契約における債務不履行および瑕疵担保責任を中心に、裁判事例や実務契約対応を織り交ぜながら考察した。

この結果、これらを請求原因とする裁判では、最終的には契約解除ないしは損害賠償の請求として争われることが多いが、特に損害賠償請求の場

合、相当因果関係（事実関係）と損害額の立証の困難さがあることがあることが判明した。

このことから、当事者の話合いで法的紛争を解決することを第一としているわが国では、特に、ソフトウェア技術の知識を伴うソフトウェア開発請負契約を巡る紛争解決について、裁判が利用されるケースはほんの氷山の一角であろう。

また、ソフトウェア開発請負契約は、もともと非典型契約であり便宜的に請負契約に当てはめているにすぎない。そこで、ソフトウェア開発請負契約を巡って紛争が生じたときは、通常の有体物の製作を目的とする請負契約に比べると大きく異なった特性を有しているので、これらの通常の請負契約に適用される債務不履行、瑕疵担保責任、損害賠償などについての解釈論や法理を、そのまま適用して処理することが困難な場合もあると思われる。更に、ソフトウェア開発請負契約を巡る紛争においては、一般にソフトウェア技術も絡み複雑な事実関係となっているケースも多いので、これらを紐解きながらケースバイケースで判断しながら処理せざるを得ないのではないだろうか。このため、このような紛争処理解決には多大な時間と労力を要する場合が多いことが予想される。

しかし一方では、当事者は、このように解決に多大な労力等を要する紛争ができるだけ生じないようにすること、すなわち紛争予防に努めることが何よりも肝要である。この点について、取引契約実務界では、JISA モデル契約書の策定・公表などに代表されるように、紛争予防などを目的としたソフトウェア開発請負契約書の標準化や適正な取引の慣行化への努力がなされている。

一般に、ソフトウェア開発請負契約における債務不履行が「完成すべき仕事」の内容を具体明確化しないままに開発作業を進めることに起因する場合が多いことやユーザーの参画・協力がなければ開発作業を円滑に進められないことなどの実態を勘案するならば、このようなソフトウェア開発

請負契約についての取引実務界の取り組みは、適正な取引の慣行化や紛争予防などに大きな貢献をすることが期待できる。

注

1)「Request for Proposal」略して「RFP」とも呼ばれている。この提案依頼書は、ユーザーが要求するソフトウェアをベンダーに開発してもらうためには重要であるが、情報システム技術を有しないユーザーが自らこれを作成することは困難である。そこで、ユーザーは、基幹システム開発などの場合にはシステムコンサルタントに委託して作成することがあるが、このような特別な場合を除き、きちんとした提案依頼書が提示されることは少ない。

2)　官公庁がソフトウェア開発を発注する場合は、従来は競争入札で発注先・発注額を決定していたが、近年はシステム提案書と見積書を総合的に勘案して発注先を決めるという「総合評価方式」を採用するケースが増加しているといわれる。このように官公庁発注方式は、ここに示したユーザー（注文者）が民間企業の場合の①〜③までのプロセスに似ているが、官公庁発注の場合には④のプロセスは殆ど経ることがなく、見積書の記載金額で契約締結されることになる。なお、この②のプロセスでユーザーに提出される見積金額は、あくまでも概算金額であり、④のプロセスで契約金額は決定されることになる。

3)　内田貴「民法Ⅱ債権各論」（東京大学出版会、第 20 刷 2003 年）25 頁、潮見佳男「契約法理の現代化」（有斐閣、2004 年）5 頁、なお、最高裁昭和 59 年 9 月 18 日判決（判例時報 1137 号 51 頁）では「契約準備段階における信義則上の注意義務違反を理由とする損害賠償責任を肯定した原審の判断は是認することができる」と認めている。

4)　村井武・平井宣雄「交渉に基づく契約の成立」（NBL702 号 6 頁・703 号 29 頁）では、企業間契約は、殆どが交渉に基づき成立し、契約内容の複雑化や書面化などの特質について検討している。

5)　①東京地裁平成 3 年 2 月 22 日判決（判例タイムズ 770 号 218 頁）、②東京地裁平成 12 年 11 月 24 日判決。①の判例では、システム開発を請け負っ

た原告が代金（開発費用）を請求したので、被告が前渡金の返還請求（反訴）を行なったところ、裁判所はコンピュータが完成していないとして反訴を認め、代金請求を否定した。また、②の判例では、被告から会計システムに関するソフトウェア開発を請け負った原告が被告に対して請負代金残額を請求した事案において、裁判所は未だソフトウェアが完成していないとして、この請求を棄却した。

6) 開発手法の変化については、前稿「ソフトウェア開発委託契約紛争事例の研究（1）」で論述した。この詳細は、（社）情報サービス産業協会法的問題委員会契約部会編（内布光監修）「新しいソフトウェア開発委託取引の契約と実務」（商事法務、第2刷 2003年）12頁。

7) 東京地裁平成7年6月12日判決（判例時報1546号29頁、判例タイムズ895号239頁）。この判例では、システム開発委託業務のプログラム・ステップ数が当初見積もった3万3,500ステップから実際には10万1,459ステップと大幅に超えた場合において、商法512条に基づく報酬請求権（当初委託代金の増額変更）が争われたが、裁判所は増額変更を否定した。

8) 後藤巻則「消費者契約の法理論」（弘文堂、平成14年）、前掲注3・潮見佳男「契約法理の現代化」、及び、中田裕康／山本和彦／塩谷國昭編「説明責任・情報提供義務をめぐる判例と理論」（臨時増刊判例タイムズ1178号）がこの問題を論じている。

9) 前掲注8)・後藤巻則「消費者契約の法理論」166頁は、消費者契約における事業者の義務についてフランス契約法の法理から展開しているが、情報提供義務・助言義務は、契約当事者の協力義務として根拠づけるとともに、協力義務の法的根拠は信義則であるとしている。一方、前掲注3・潮見佳男「契約法理の現代化」では、『証券取引の領域での説明義務・情報提供義務の考え方は、ドイツの「契約締結上の過失」理論（とりわけ、法定の保護義務論）からの示唆を受けて、わが国でも比較的早くから展開した。』（86頁）としながら、『最近の民法上の議論では、フランス情報提供義務論ならびに合意の瑕疵理論の影響のもと、説明義務・情報提供義務を、詐欺や不当勧誘に代表される「行為態様の不当性」に結びつけ、契約の有効性に対する評価の観点から捉えようとする一連の動きがあらわれている。』（106頁）としている。

10)　営業秘密は、不正競争防止法 2 条 4 項で定義されている通り「秘密として管理されている生産方法、販売方法その他の事業活動に有用な技術上又は営業上の情報であって、公然と知られていないもの」をいうが、同法は、この営業秘密に関し、平成 16 年の改正で民事訴訟による保護を強化し、更に、平成 17 年の改正（平成 17 年 11 月 1 日施行）で刑事罰の強化・拡大を図っている。この 17 年改正により、ベンダーの従業員等がユーザーが提供した営業秘密を不正に使用・開示すると、ベンダー自身（法人）も 1 億 5 千万円以下の罰金刑が科せられることになった（同法 22 条）。

11)　東京地裁平成 9 年 9 月 24 日判決（判例タイムズ 967 号 168 頁）。この判例では、コンピュータ（ハードウェア及びソフトウェア）の売買契約における作業日程の遅延について、買主である企業にも、一つの企業体として事業を行い、その事業のためコンピュータを導入する以上、自らも積極的に売主との打合せに応じ、売主に協力すべき信義則上の義務があると判示している。

12)　プロトタイプモデルでは、ベンダーがユーザーの要求をイメージして、とりあえずソフトウェアのプロトタイプ（原型）を作ってしまう。そして、そのプロトタイプのソフトウェアを、ユーザーからの細かな要求・指示を受けて修正・改変しながら最終的にユーザーが求める完成品に近づけていく開発手法であるから、ユーザーは開発作業そのものに積極的に参画しなければならないことが明らかである。この詳細は、前掲注 6)「新しいソフトウェア開発委託取引の契約と実務」14 頁以下参照。

13)　この点に関して、(社）情報サービス産業協会・平成 14 年 5 月発行報告書『新しいソフトウェア開発委託取引のあり方（ソフトウェア開発委託モデル契約書と解説)』(No. 14―J001）掲載のモデル契約書「第 2 章本件業務の推進体制（第 8 条～第 11 条)」で規定している。なお、このモデル契約の主要条項の趣旨については、次の URL 参照；http://www.jisa.or.jp/legal/contract_policy2002.html

14)　東京地裁平成 10 年 12 月 21 日判決（判例時報 1681 号 121 頁・判例タイムズ 1045 号 194 頁)。この判例では、自動車教習所における運転を擬似体験させる装置の共同開発契約の締結を目指した協定に基づいて原告は担当分の開発を終えたが被告は開発できず契約締結に至らなかった事案において、裁判所は契約締結上の過失に基づき原告の被告に対する開発費用の損害賠償請

求を認めた。

15） 前掲注 3）・内田貴「民法Ⅱ債権各論」123 頁、258 頁。笠井修「保証責任と契約法理論」（弘文社、平成 11 年）245 頁

16） 従来のコンピュータには、このソフトウェアの互換性（コンパチブル）の問題があったが、近年のパソコンがオープンシステム化したため、この問題は解消した。つまり、従来はコンピュータメーカーごとに OS が異なっていたため、例えば A 社製汎用コンピュータ用のアプリケーション・ソフトを B 社製汎用コンピュータで使用することができず、B 社製コンピュータに適合するようコードの変換（コンバージョン）が必要であった。しかし、今日の各メーカーのパソコンは Windows など事実上標準となった OS を搭載しているので、このような互換性や適合性の問題はあまり生じてこない。

17） IT 用語辞典 e―Words（http://e-words.jp/w/E38390E382B0.html）によれば、プログラム・バグとは「プログラムに含まれる誤りや不具合のこと。ソフトウェアの正常な動作を妨げる邪魔物であるプログラムの誤りを小さな虫になぞらえたのが語源。人間が作成する以上、よほど小規模のものでない限り、バグのまったくないプログラムを作成するのは不可能である。このため、ソフトウェアの開発過程ではバグを取り除く作業（「デバッグ」と呼ばれる）が非常に重要となる。」とある。

18） コンピュータ訴訟研究会「コンピュータ紛争Ⅱ―ケース研究と予防対策」（尚文社、2000 年）271 頁では、「周知のごとく、私たちが目にする書物や各種の説明資料は、校正作業を経たものがほとんどであるが、それでも、誤字や脱字が皆無ではない。……ソフトウェアのコーディング作業は、人間が頭脳を駆使して行なう頭脳労働である点では、原本の翻訳や執筆と極めて類似している。それがゆえにバグの発生の仕方も似てくるものと考えられる。いわば、これがソフトウェアにバグが入り込み、存在する理由である。」とある。

19） 朝日親和会計社編「改訂版・ソフトウェア取引の実務」（中央経済社、平成 2 年）47 頁では、「請負型のプログラム開発の場合で、特約のない場合を例にとると、以下のような範囲で損害賠償責任が生じるものが考えられる」として「誤り（バグ）の存在は不完全履行」の一事由としている。

20） 北川善太郎編「コンピュータシステムと取引法―システム契約の法政策

的検討―」（三省堂、1987 年）166 頁では、「ソフト開発者によって開発されたプログラムの瑕疵（バグ）が原因となって事故が生じた場合には、ソフトウェアの瑕疵が独自の法的責問題性を帯びてくる。それが、パッケージ化された商品であるときは、情報の瑕疵による製造物責任の一環として論じることができよう。」と述べている。

21）　これと同旨の考え方は、前掲注 6)「新しいソフトウェア開発委託取引の契約と実務」153 頁

22）　重大な瑕疵と認めた判例として、①東京地裁平成 5 年 1 月 28 日判決（判例時報 1473 号 80 頁）は、トレース作業処理等のプログラム作成請負契約について、作成されたプログラムに重大な（不具合が多く使い物にならない）瑕疵があり、契約を締結した目的を達成できないとして、民法 635 条により契約解除と損害賠償を認めた。②広島地裁平成 11 年 10 月 27 日判決（判例時報 1699 号 101 頁）は、基幹業務システムのソフトウェア開発において、同種のソフトと同等の機能・性能を有しないなど種々の欠陥について債務不履行を認定した。③東京地裁平成 14 年 4 月 22 日判決（判例タイムズ 1127 号 161 頁）は、請負開発された販売管理システムを本格稼動後、重大な瑕疵（システムの設計自体に問題があること）が発見されたが、納得いく修補ができず、このシステムの継続使用を断念せざるを得なくなったので、ベンダー側に対する損害賠償請求を容認した。

23）　東京地裁平成 9 年 2 月 18 日判決（判例タイムズ 964 号 172 頁）は、プログラムのバグがソフトウェアの欠陥であるといえるための要件を示したものといえる。なお、このほかソフトウェアの瑕疵を否定した判例として、①那覇地裁平成 12 年 5 月 10 日判決（判例タイムズ 1094 号 117 頁）では、入力時間や報告書様式がユーザーの要求通りになっていないとしても、これは機能上の問題ではなく使い勝手の問題であるからソフトウェアの瑕疵とは認められないとした。②東京地裁八王子支部平成 15 年 11 月 5 日判決（判例時報 1857 号 73 頁）では、ソフトウェアが概ね基本設計書の通りに構築されている以上、入力作業や入力ミスの確認作業などが煩雑で時間がかかりすぎることはソフトウェアに瑕疵があるとまでは認めがたいとした。

24）　東京地裁平成 12 年 12 月 26 日判決（平成 11 年（ワ）第 6128 号、(財) ソフトウェア情報センター「ソフトウェア契約関連判例に関する調査報告書

―平成 13 年度版」43 頁）

25） 加藤雅信「新民法大系Ⅴ　事務管理・不当利得・不法行為」（有斐閣、平成 14 年）141 頁では『一般に、条文上、過失責任であることは、「故意、過失」等の文言を要件に明示することで示される。無過失責任の規定の仕方としては、製造物責任法が典型的であって、通常、条文上無過失責任であることを積極的にうたわず、単に要件のなかで「故意、過失」に言及しないことで、無過失責任であることを示す途がとられる。』とある。

26） Central Processing Unit の略で、中央処理装置のこと。コンピュータの中で、各装置の制御やデータの計算・加工等を行なう中枢部分。

27） Read Only Memory の略で、読み出し専用記録装置のこと。一度書き込まれた情報を読み出すための記憶装置で、書き換える必要のない情報や書き換えられては困る情報を記録させる。

28） 内田貴「民法Ⅲ〔第 2 版〕債権総論・担保物権」（東京大学出版会、第 2 版第 1 刷 2004 年）125 頁

29） 前掲注 13）・『新しいソフトウェア開発委託取引のあり方（ソフトウェア開発委託モデル契約書と解説）』（No. 14―J001）98 頁参照

30） 東京地裁平成 6 年 1 月 28 日判決（判例時報 1515 号 101 頁）

31） 内布光「IT 社会における企業取引法」（商事法務、2003 年）186 頁

32） 完成保証は、映画の製作資金を提供する金融機関等に対し、完成保証会社が定められた期限までに、予算内での映画が完成されることを保証する制度。詳細は、独立行政法人経済産業研究所「日本型映画完成保証に関する調査研究報告書」（平成 16 年 3 月）参照。URL は以下；http://www.meti.go.jp/policy/media_contents/

33） 田村義之「著作権法概説〔第 2 版〕」（有斐閣、第 2 刷 2003 年）55 頁；ベンダーによって様々な独自開発手法が編み出されているが、著作権非侵害対応策としては「クリーンルーム」方式による独自開発手法が一般的。著作権がソフトウェア（プログラム）のアイデアは保護せず、表現だけを保護することに着目し、既存の同種のソフトウェアに類似しないようにソフトウェアを開発するための方式。

34） 広島地裁平成 11 年 10 月 27 日判決（判例時報 1699 号 101 頁）

35） 東京地裁平成 12 年 2 月 25 日判決（平成 9 年（ワ）第 20237 号・第 27509

号、(財) ソフトウェア情報センター「ソフトウェア契約関連判例に関する調査報告書―平成 12 年度版」54 頁)

36) 前掲注 22)・①の判例

37) 幾代通・広中俊雄編集「新版注釈民法 (16) 債権 (7)」(有斐閣、初版第 8 刷平成 15 年) 151 頁、前掲注 3)・内田貴「民法Ⅱ債権各論」258 頁、

38) 東京地裁平成 15 年 5 月 28 日判決 (平成 14 年 (ワ) 第 15745 号、なお、本判決は公刊されていない。)

39) 前掲注 22) の①判例参照

40) 東京地裁平成 8 年 7 月 11 日判決 (判例時報 1599 号 99 頁)、なお、前掲注 23) の判例参照

41) 平井宣雄「損害賠償法の理論」(東京大学出版会、第 12 刷 1997 年) 3 頁

42) 前掲注 41)・平井宣雄「損害賠償法の理論」169 頁

43) 前掲注 28)・内田貴「民法Ⅲ〔第 2 版〕債権総論・担保物権」155 頁

44) 東京高裁平成 5 年 1 月 25 日判決 (判例タイムズ 857 号 190 頁)

45) 東京地裁平成 12 年 9 月 21 日判決 (平成 11 年 (ワ) 第 6181 号、判例マスタ)；ソフトウェア会社 X と外国語翻訳・通訳会社 Y とが、通産省講演の実習訓練プログラム運用支援のシステム開発事業を共同で行なうことに合意し開発を進めていたが、その後この関係を解消した場合において、XY はこの事業について行なった全作業につき適正な評価をして相当と認められる金額の精算をする合意が合ったと認定された事例。

46) 前掲注 13)・『新しいソフトウェア開発委託取引のあり方 (ソフトウェア開発委託モデル契約書と解説)』掲載のモデル契約書第 38 条

47) モデル契約の主要条項の趣旨の URL；http://www.jisa.or.jp/legal/contract_policy2002.html