侵害のパラドックス：侵害の論理についてのエンジニアの見方

Dr. Douglas K. Brotz

ソフトウェア技術者は，著作権，トレード・シークレットおよび商標の侵害の場合には，侵害したとして訴えられた人を「悪者」だと考えるのに対して，ソフトウェア特許の侵害事件の場合に，逆に特許保持者の方を「悪者」だと考えるのはなぜだろうか．この一見矛盾する振舞いが，じつは技術者の倫理を反映した結果であるという点について説明したいと思う．この倫理の基礎にあるのは，「コピー」という概念についての技術者の理解である．これが知的財産権の制度とどう影響し合うかを見る前に，この概念を明確にしておく必要がある．

コピー対同時発明

米国および他の多くの国では，コピーは「悪い」行ないだと教えられている．生徒は学校で独力で学び独力で問題を解くことを奨励される．グループ学習の成果は，時として個人学習に優るけれども，グループ学習は学校では特別に許可された場合にしか認められていない．原則はあくまでも個人学習である．生徒はこのことをよく教え込まれているので，コピーの禁止はなかば本能になっている．こう感じるのは知的財産権法とは何ら関係がないことである．つまり，もはや著作権では保護されていない古い作品をコピーすることも，保護されている作品をコピーするのと同様に「悪い」ことだと考えられているのである．

ある行為が「コピー行為」であるとみなされるためには，コピーしている人が，自分が実際にコピーしているということを認識していなければならない．また，コピー行為には，コピーされている作品が利用可能であること，そしてコピーする人がその実物を実際に使用することが必要である．

これに対して同時発明は，2人あるいはそれ以上の人が，他人の作品の存在を知らずに類似のアイディアや発明を考案したときに起こる．同時発明の結果は，コピーに似ているように見えるかもしれないが，コピー行為ではないことはもちろんである．同時発明にはコピー行為に着せられる汚名はない．それは多くの人が同じ問題に別々に取り組む場合に，当然予想される結果なのである．

著作権，トレード・シークレット，(ある程度までの)商標の侵害は，意図的なコピー行為を含んでいる．確かに，著作権とトレード・シークレットを侵害したとされるためには，侵害者が保護される作品にアクセスし，そして結果が本質的に類似していることが必要である．侵害者は，いつでも自分が行なったことが何であるかを，そしてそれが悪いことであることを知っていたと推定されるのである．このように，侵害者が自分が悪い行為をしていると知りながらも，ともかくそれをやってしまったという場合には，関心のある傍観者は侵害者を非難するのは倫理的に正しいと感じるのである．

特許侵害

特許侵害についての非難は，著作権侵害に対する非難とは全く異質である．行為中に他人の作品について全く知らなくても，特許を侵害したとして非難され，刑法上有罪とされうるのである．同時発明であるという抗弁は特許事件では認められない．特許を侵害したとして非難される者が，著作権侵害者ほどには「悪く」はない，という倫理的感覚は，特にソフトウェア特許の場合に強いのである．

以下で，特許侵害について論じる際に，「故意の侵害」は除外することにする．米国法は特許侵害と「故意の」特許侵害を区別している．「故意の」侵害は，侵害者が単に侵害しただけでなく，侵害時に侵害を確認していたときにのみ成立する．これは典型的には特許文書を

Principal Scientist, Adobe Systems. Inc.

完全に知っており，特許が請求された方法あるいは物を明確に理解した上で，保護されている方法や物を作る，あるいは使用するという意思決定を含むものである．典型的な技術者ならば，故意の侵害が倫理的に「悪い」ということに同意するだろう．この判断は，著作権やトレード・シークレットの侵害が「悪い」のと同じ理由づけにもとづいている．つまり，意図的なコピー行為を含んでいるからである．米国法も同様にこの判断を示している．なぜなら，故意でない特許侵害に対する通常の賠償と異なり，故意の特許侵害に対してははるかに大きな賠償（3倍賠償）を認めるからである．ここでは故意の侵害についての倫理には触れないが，故意の侵害の法理は，後で述べる特許の秘密に関して特に厄介な帰結をもたらす．

ソフトウェア・エンジニアリングの本質

ソフトウェア・エンジニアリングは，数学の伝統に由来している．計算，分類，配列といった基本的なソフトウェアの操作は，いずれも数学的なものである．数学の場合と同じく，ソフトウェアの構造は完全に思考上のものである．こうした思考上の構造は，書くことによって表現され，人の創造力の及ぶ限りの精緻さを備え，物理的限界には制約されないものである．職業としてのソフトウェア・エンジニアリングの魅力の多くは，物理的には想像し得ないほど巨大な「マシン」を構築する機会を技術者に与える点である．（もちろん，ハードウェアなしのソフトウェアは，パン焼きのレシピそのものがパン焼き機ではないのと同様，機械とはいえない．ソフトウェア技術者が構築する「マシン」は，現実にはソフトウェアとそれをのせるハードウェアとの組合せとして存在する）

盗作とは異なり，同時発明は数学では通常のこととして受け入れられている．ニュートンとライプニッツのよる微積分の同時発見はいうに及ばず，数学の歴史は同時発明の例に満ちている．その場合，片方だけがその功績を讃えられるではなく，むしろ両方がその偉大な業績を認められる．もっと卑近な例を挙げれば，多くの数学雑誌が，読者への出題と返送された解法から採用された解答を載せている．通常，特に際立った解法が，同様の解法を返送した人たちの名前とともに紹介されるが，これらの人たちは盗作だと非難されるのではなく，同じアイディアにもとづきながらも細部にわたる表現が異なる場合には，よくやったと認められるのである．

典型的には，大きなプログラムを構築する作業は，何百もの小さな問題を解くと同時に，それら何百もの解答を，それらがすべて同時に働かせるよう構成するという，より大きな問題を解く行為である．個々の小さな問題に対しては，さまざまの異なったテクニックを適用することが可能である．またプログラム全体を統合するためには，いくつかの全体構造統合手法が用いられる．同じ問題に別個に取り組むソフトウェア技術者が，多くの類似の基本的アイディアを考えついたり，使ったりすることはありうるが，別個に開発された大システムが，コード全体を通じてよく似ていることはまず有り得ない．

大ソフトウェア・プロジェクトの成果は，通常はオブジェクト・コードでのみ利用可能である．ビジネスの世界では，ソース・コードの詳細は秘匿され，その内容は多大な労力なしには知りえないものである．ソース・コードの詳細な知識なしに，機能の類似した大きなプログラムを構築することは，もともとのプログラムを作成するのと同じくらい時間のかかる大仕事である．

物理的対象に対する特許とソフトウェア特許

世界の特許の歴史は，主として新しい機器の説明とその保護によって成り立っている．ある特定の機器の分野で働く技術者は，既存の機器をよく知っており，その作動の原理は部品を観察することで容易に分かるものと想定されている．特許制度は，ライバルが特許を取得した発明品を見て，そのデザインをコピーすることから発明者を保護する制度である．この場合，コピーする者は，発明品をコピーするためにわざわざ特許文書を見る必要はない．つまり物理的物体は容易にコピーできるのである．

ところがソフトウェアの場合はそうではない．すでに述べたように，ソフトウェアの構造に関する原則は，ソフトウェアの機能を表面的に観察するだけでは容易には分らない．ソフトウェア特許の侵害を試みる者が必要な要素を見つけるためには，特許文書を読んでその内容を理解するか，あるいはプログラムをリバース・エンジニアリングするという，気の遠くなるような労力を割かなければならない．実際，ソフトウェアの開発が，他人のコードを詳細に検討することによって行なわれることはほとんどない．ほとんどいつの場合でも，必要な機能を得るために新しいコードをデザインするか，再使用可能なコードの組合せを新しく配

列し直すことによって行なわれるのである．

特許は秘密である

世界の特許制度は，その開示とひきかえに，新発明に一定期間の保護を与えるという仕組みとして正当化されているが，開示の面が十分に実行されているとは言い難い．実際のところ，現役のソフトウェア技術者にとっては，すべての特許が秘密であった方がよいくらいである．これにはいくつかの理由がある．

一般に，ソフトウェア技術者は与えられ課題に対して，コンピュータ・プログラミングの一般的な知識を適用してソフトウェアを組む．ほとんどのソフトウェアは，特定の機能に関連した特定かつ独特のメカニズムに関係している．私の経験では，ある人が作ったプログラムのすべての部分と互換性を達成するためには，大プログラムの半分以上のコードを知る必要がある．その結果，他人のプログラムを読むには，細部にわたる表現と，自分が知りたいアイディアを見分けるための異常な努力を払わなくてはならない．そのため，ほとんどのソフトウェア技術者は，別個に自分のソフトウェアを組むことになる．一般的な参考資料を見てアルゴリズムを理解することはありうるが，典型的なソフトウェア技術者が特異な記述で書かれた特許文書を参考にすることはまずない．

さらに重要なことに，特許の解釈に習熟していないものにとって，特許のクレーム（要求条項）で実際に何が保護されているのかを理解するのは不可能に近い．特許弁護士は，法廷で保護を受けられるようにデザインした特許クレームを書いて金を稼ぐが，ほとんどのソフトウェア技術者は，高価な特許弁護士の助言なしには特許クレームが何を意味しているのか理解することはできない．さらに，特許保持者が誰かを特許侵害だと非難する場合，その保持者が主張するクレームの意味は，人がそれを単純に読んで理解する意味とはずいぶん異なることがよくある．それゆえ，特許文書は侵害を避けようとするプログラマーには価値がないのである．

特許文書の文言もわけのわからなさのために，既存の特許の公開がプログラマーにとっては無意味なものになっていることに加えて，故意の侵害の法理の意図せざる結果として，既存の特許の公開は法的にも無益なものとなってしまっている．故意の侵害に対する罰則は非常に厳しい．そして，故意の侵害が成立するためには，侵害者が特許を理解していたということの立証が必要であるから，米国における通常の法律的助言は，「他人の特許を読むな」ということになってしまうのである！　このように，故意の侵害を避けるためにも，特許文書を無視することが有効なのである．残念ながら，それは特許が公開という約束を果たさないということでもある．

特許保持者は「悪者」である

このように実務では，ソフトウェア分野での特許侵害の主張は，非難されるプログラマーにとっては青天の霹靂である．コピーせずに，問題に対して独自の解法を開発したプログラマーが，他人の財産を盗んだと非難されるのだ！　しかも，この訴えの権利は，伝統的にソフトウェア分野を理解せず，範囲の不明瞭な独占権を与えている政府機関との秘密のやりとりによって獲得されたものなのだ．以上の理由から，著作権，トレード・シークレットや商標の場合とは逆に，特許侵害（故意の侵害は除く）の場合には，典型的なプログラマーの倫理は，原告が「悪」で被告が「善」と結論しやすいのである．

米国では，原告イコール「悪者」というイメージは，故意の侵害に対して与えられた救済措置によって一層増幅されている．というのは，巨額の損害賠償を目当てに，ほとんどの特許侵害訴訟が故意の侵害を主張する．単純な侵害の訴訟がすでに進行している場合，より強い故意の侵害の主張を成り立たせるために，貧弱な証拠が提出される．通常，この追加証拠は特許文書からでっち上げた空想物語でしかない．自分で問題を解いただけで，何か悪いことをしたと非難されるだけでも不愉快な技術者にとって，こうした追加的な（そして通常はでたらめの）故意の模倣行為であるとの非難は，原告を一層憎悪する理由となるのである．

リバース・エンジニアリング

ここで少し，リバース・エンジニアリングの問題と，それと技術者の倫理的理解とがどう関連するかを考えてみたい．リバース・エンジニアリングは，技術者が新しいシステムを開発するのではなく，特定のシステムがどう働くのかを発見しようとする努力である．ソフトウェアの分野では，システムが文書もなくオブジェクト・コードでのみ利用可能な場合は，リバース・エンジニアリングには極めて精緻な道具と技術を必要とする．シミュレーション，論理分析，暗号解読やデコンパイレーションは，使われる道具や技術の一部で

しかない．

リバース・エンジニアリングそのものについては，私は何ら悪いことだとは思わない．作用の仕方を理解することは知識を増大させ，将来のより良い技術製品を生み出すことにつながる．もちろん，あらゆる強力な道具と同様に，それは良い目的にも悪い目的にも使われる．しかし，それ自体としては，リバース・エンジニアリングは情報収集の１つの方法に過ぎない．リバース・エンジニアリング(あるいは，デコンパイレーションなどの特定のリバース・エンジニアリング）について，いくつかの法律的提案がなされている．それらは，すべてのリバース・エンジニアリングを非合法化するものから，明確にそれを保護するものまで多岐にわたっている．これらの提案が，道具の使われ方よりも道具そのものに焦点を当てるならば，それは方向を誤っていると思う．

知的財産権制度の適切さ

法制度が社会にどのくらいよい貢献をするかを測る１つの目安は，それがどれだけ善に報い悪を罰するかである．そこでさまざまな知的財産権制度が，ソフトウェア技術者の価値体系にどれだけ合致しているかを要約してみたい．

著作権法は，技術者の価値体系に非常によく合致している．それは最もあからさまな作品の盗用から作者を保護する．これは「コピーしてはいけない」という，すっかり染み込んだ観念に合致する．しかもこの場合，技術者は自分がいつ法を逸脱するかを認識しているはずである．

コモン・ロー諸国（英国，米国など）では，法は法典の意味と判例という２つの原理に立脚する．著作権法に関しては多くの判定の蓄積がある．この法の体系は，技術者の倫理体系とよく調和している．技術者がある行為（システム全体を１行１行コピーする行為）を悪いと確信する場合，法もそのように判断する．技術者が良い行為と判断する行為（盗作ではなく，同時発見）は，法も同様に判断する．技術者が灰色領域だと感じる場合，法も同様である（どの程度のコピーが侵害とみなされるか．構造，処理の流れ，構成(structure, sequence and organization (SSO)）は保護されるべきか)．こうした合致は，著作権法の論理がソフトウェア技術者の倫理と合致することを示している．

この短い文章で，私はトレード・シークレット法について触れることはできなかった．コピーから保護するという点で，著作権法同様，技術者の倫理に合致しているとだけ指摘しておく．

これに対して，ソフトウェア技術者は商標法にはあまり関心がない．それはマーケティング部門の人たちの専門であろう．商標の侵害が通常故意のコピーを意味する点では，この法は技術者の倫理と合致する．

ソフトウェア技術者の倫理は，特許法の現行の解釈とは激しく衝突する．特定のアイディアやアルゴリズムが特許保持者の排他的財産であるという概念は，あらゆる数学の歴史と教育に逆行する．それは誰もフェルマーの最終定理を他の定理の証明に，それも17年間も使えないというようなものだ！　ソフトウェア技術者は，ソフトウェア特許の「開示」からは何の利益も得ることはできない．また，単に自分で作品を作っただけで盗んだことになるという考えは，技術者の世界観には全く合わないものである．

リバース・エンジニアリングについての最善策は，特定のケースにおいてリバース・エンジニアリングがどう使われているのかを見ることであろう．米国法の下では，著作権の侵害は，保護される作品へのアクセス，および作られたものと元の作品との本質的な類似を証明することによって証明される．リバース・エンジニアリング行為は，作品へのアクセスの証拠となる．もしその存在が証明された場合には，本質的類似の側面が，リバース・エンジニアリングが良い目的のためになされたのか，悪い目的でなされたのかを公正に決定することになるだろう．

（訳：中川淳司，今野　浩）