法は保護不能な対象を追い求める言語道断の愚か者なのか？

Andy Johnson-Laird

はじめに

このペーパーは少し変わっている．内容の全部が全部筆者の意見とは限らないからである．このシンポジウムの眼目の1つがプログラムやアルゴリズムの保護に関する米国の技術者の見解の開陳である以上，筆者の意見だけではなく，筆者がインターネットを通じ，また書かれたものを通じて得たソフトウェア技術者およびその他のプログラムビジネスに関係する人たちの意見を反映させた，より一般的な観察を述べることにする．

来た，見た，コピーした

現代は情報化時代である．我々は電子的に結び付けられた社会にいる．デジタル化された情報，画像，音は世界中で瞬時にコピーされ伝達される．知的創造物はいまや巨額の富を生み出す．そして，巨額の富が盗まれている．被害額はソフトウェアだけで世界全体で74億ドル．これにはデジタル化されたオーディオビジュアル作品やデジタルタイプフェースなどの他の形態の知的作品は含まれない．我々はポルノグラフィーすら盗み合っているのだ．

我々人類が「ファイルの支配者」になる代わりに，知的財産権に関して未開人に堕落してしまったのは皮肉なことである．我々は欲しいものを好きな時に手に入れる．とることではなく捕まることが「罪」だという前提で行動している．オリジナルが元のままで残っているなら盗みとはいえない，などといった子供のような理屈をつけている．デジタル情報の提供者は完璧な娼婦のようなものだ．彼らは自分の持ち物を売るが，その後も持ち物は持ち続けているのだから．盗んだ後もそのままあるなら，いったいどこが被害者なんだ？

President, Johnson-Laird, Inc.

インターネットやデジタル化された知的財産権が世界を1つにしたという点はしばしば指摘されてきた．「ネットワーク市民（netizens）」がタイプと転送の速度でコミュニケーションを世界中で展開し，新たなデジタル秩序を採用し，これに適応する能力を試している．しかし，こうした技術が社会の倫理にかつてない試練を与えていることを指摘する者はほとんどいない．

与えられている試練とはこうである．技術革新の結果，我々は他人の知的作品を素早く，安価に，しかも（ほとんど）気づかれないうちにコピーすることができるようになった．我々はこの誘惑に抗しきれるか？ソフトウェアの盗用やソフトウェア訴訟の件数，そしてプログラマーの行動に関する私自身の観察によると，その答は「否」である．

それでは，我々はこうした行動を取り締まる法を作らなければならないのか，たとえその強制はきわめて疑わしいとしても？　我々の行動を規制する法の大半は，強制されて，あるいは強制されると意識されて初めて実効性を持つ．我々は自己強制能力，つまり，それが法であるからという理由で自分の行動を法の範囲内に納める能力を失ってしまっている．我々は自分がつかまるかどうかの評価にもとづいて行動を規制しなければならない．

しかしながら，たとえ強制できない（あるいはごく弱い強制しかできない）法であっても知的財産権の領域を規律するために作られなければならないとすれば，それはどのような法でなければならないか？　既存の法ではだめか？　つまり，我々はこの新しい誘惑の機会を統制することができるか？

ソフトウェア技術者？

このペーパーでは，「ソフトウェア技術者」よりも「プログラマー」という言葉を優先的に用いる．著者自身の経験から判断するに，「ソフトウェア技術者」という

言葉を使うと，プログラマーの地位を高くし過ぎるように思われるからだ．それはあたかもネズミ捕り人を「齧歯（げっし）類技術者」と呼ぶようなものだ．「真の」技術者にふさわしい修練とプロフェッショナリズムを兼ね備えたプログラマーなどほとんどいない．(少なくとも西洋に関する限り）大半のプログラマーはそうした資質からはほど遠く，せいぜいのところが，才能と天分に恵まれた，素晴らしい，創造的な，天才に近い，無誘導ミサイルである．私の意見を疑う人は，なぜコンピュータソフトウェア産業でのみ，原型となる製品がいったんでき上がれば直ちに公衆に販売されるという現象が起きるのかを考えてみるとよい．他の技術分野では，大量生産される製品はまず原型が作られ，それからその原型は廃棄され，その後で初めて大量生産用製品が作られ，販売されるのだ．

技術の現状

複製のコストが最初から作り出すコストよりも低くなった時点で，我々が今直面している倫理的なジレンマが始まった．たとえば，装飾を施したラテン語で丹念に書かれた中世の彩色手稿を考えてみよう．それを作るコストと筆写するコストは同じである．今度はラテン語ではなくコンピュータ言語で書かれた，mputという現代のコピーに関する言葉を考えてみよう．

この単純なコンピュータコマンドは，打ち込みにほんの2，3秒しかかからないが，これでデジタル表示された世界中の彩色手稿をコピーすることができる．国境も，ベルヌ条約も，トレードシークレット，著作権や特許に関する法も無視して，米国西海岸のコンピュータから完璧なコピーを東京や台北やトロントのコンピュータに瞬時に送ることができる．もう1つだけ例を出すと，世界中に発信されたデジタル情報がたとえマイクロソフトの最新のオペレーティングシステムのソースコードの違法なコピーであったとして，コピーした人間は匿名のままでいることもできるのである．現代の技術を持ってすれば，これらすべてが実行可能である．

コンピュータと世界全域をカバーするインターネットの技術は，いともたやすいコピーを可能にし，何の摩擦もない，自由な世界大の情報伝達メカニズムをもたらしたのである．

我々コンピュータ時代の人間は，いまや世界中のどこの国にでも，我々のデジタルの腕を差し伸べることができるあらゆるものを伝達することができる．もはや問題はそれが可能かどうかではない．それは可能である．そして，いったんなされてしまえば，それを取り消すことはできない．真の問題は，我々はそれをしないようにすることができるのかである．

プログラマーは何を望んでいるのか？

私の判断によれば，米国のプログラマーが望んでいるのは以下の6つである．

1．過去の経験を評価した雇用関係
2．自分の，そして他人の先行業績を「採用し，応用し，改良する」ことができること
3．ごく簡単な手段で自分の作品を盗まれないこと．
4．横やりを入れず，明確な指針を提供してくれる法
5．活発な競争
6．良いソフトを書いて，世界中から注文が殺到すること

これらの中には現在の法とは抵触するものもある．が，ともあれもっと詳しく述べてみたい．

1．過去の経験を評価した雇用関係

プログラマーの職場移動はきわめて頻繁である．そもそも職場に物理的に存在しなくてもよいし，気が向いたらいつでも勤め先を変えてしまう．あるプロジェクトから得られた特別の知識は多くの場合複製不能で，それはプログラマーの頭脳の中にのみ存在する．プログラマーのよくいう格言に「ソフトウェアを書くのは難しい．だからそれを理解するのも当然難しい」というのがある．「コードが文書化されたら，いったいどうやってそれがUNIXだってわかるんだい？」というのもある．

こうした雇用環境の下では，雇い主は通常，現在のプロジェクトとは無関係に，多くの経験を持ったプログラマーを雇いたがる．そうした経験を持ったプログラマーは競争企業で同種のプロジェクトに従事している確率が高い．誘惑を感じるのはプログラマーの側も同じである．そうしたプログラマーは自分の頭の中にある価値のゆえに競争企業に接近する．

このプログラマーが転職を決意したとして，新しい会社に持ってゆき，すぐに使えるのは何か？　あからさまにこの質問をするプログラマーは多くはないし，新しい雇い主がたずねることはもっと少ない．

当然のことだが，知的財産権弁護士はこの種の質問が得意である．もっとも，その答は当人の国籍によっ

て異なる（それが管轄と準拠法を決定する）．

プログラマーの多くは，もし自分があるソースコードを書いたならば，それは自分が持っていって良いものだと感じている．だって，自分で思いついて，コードを作り，それを完成させるために夜も週末も残業してやっと仕上げたんだ．だから当然それは自分のものだろう？

ソースコードは持ってゆけないという考え方に直面すると，それならそれで，コードを書き直して，前のやつよりも優秀で小型で高速で高性能なコードをこしらえてやろうと考えるプログラマーは多いだろう．

しかし，これは合法的だろうか？　新しいプロジェクト用に新しい雇い主の下でこうした書き直しをするのは，元のプロジェクト・雇い主を侵害することになり，訴訟は避けられないのではないだろうか？

ここにジレンマが生まれる．あるプログラマーの知識に魅力を感じてその人を雇いたいと思っても，雇った後にその知識を利用することはできないのだ．かりに利用すれば訴訟が待っている．

2．自分の，そして他人の先行業績を「採用し，応用し，改良する」ことができること

他の有益な技能と同じく，プログラミングは人から教わることができない．それは学習によってしか習得できない．そして，学習は自らの成功と失敗，そして他人のプログラム作品の吟味を通じてなされる．

プログラマーは配管工や大工と似ている．彼らはその職業生活を通じて自分のお得意の道具の詰まった工具箱を持って世界を渡り歩く．工具箱の中身は，学校で教わったものもあれば，誰かのところで仕事をしていて身につけたものもある．雑誌や教科書に載っていたのを覚えたものもある．

しかし，現在の知的財産法の下では，こうした学習プロセスは雇い主や他人の知的財産権の侵害とみなされてしまう．工具箱は違法に取得された知的な道具で一杯ということになる．トレードシークレットを侵害したものもあれば，著作権を侵害したものもある．特許を侵害したものもある．

プログラマーが新しい課題を与えられた場合に問題が顕在化する．この問題を解決するためにいかなる知的基盤に立てばよいのか？　彼（女）の工具箱の中身を使っても罪に問われることはないだろうか？　それとも，それは訴訟を招くだろうか？　彼（女）は，前の雇い主の下でその問題を解いた解き方を使いたいと思う．しかし，今の雇い主にはそれを認める法的権限はないし，彼（女）がそれを開示することは法的には許されない．たとえ最初の職場でその課題に取り組んだ理由が，そのプログラマーの過去の経験のゆえであったとしてもだ．

自分の経験の再利用を否定されることはプログラマーにとってどうしても納得のゆかないことである．優れたプログラマーは，経験にもとづいて広範囲の問題の解決法を知っている．この哲学に反するいかなる立法も，実際には無視されるだろう．その結果，現在の雇い主は，訴訟が提起されて初めて自分が実はピンの抜かれた手榴弾を持っていたことに気づくのである．

3．ごく簡単な手段で自分の作品を盗まれないこと

プログラマーの中には自分の作品をごく簡単な手段で盗まれないようにしたいと考える者もいれば，自分の作品を誰でも自由に利用できるようにしたいと考える者もいる．どちらにせよ，プログラマーは自分の好きな方を選びたいと思っている．しかし，現実世界の法はそうはさせてくれない．

根本の問題は，補償の問題であるように思われる．補償といっても必ずしも金銭の補償とは限らない．むしろ，プログラマーが選択する形態の栄誉による補償である．

プログラマーの多くは「雇われ作品」という考え方をしない．プログラマーの作品を所有するのは雇い主だとは考えない．こうしたプログラマーにとって，書くことは所有を生み出す．いったん書かれば，そのプログラムは法が何といおうが雇用契約が何といおうがそれを書いた者の物のように感じられるのだ．

だから，誰かがソースコードやユーザーインターフェースや一般的なアイディアをコピーしたら，そのプログラマーは自分の作品をコピーされた画家や彫刻家と同じ憤りを感じる．そのプログラマーは，子供を誘拐された親と同じように，自分の労働の成果を誘拐されたと憤慨する．

4．横やりを入れず，明確な指針を提供してくれる法

現在の米国の法は，プログラマーの期待よりもはるかに不明確である．プログラマーはこうこうとした光と漆黒の闇の世界に生きている．真実と偽りのみが存在し，中間の灰色を彼（女）らは認めない．プログラ

マーは，ことを明確にせず，混乱のみを招く「法」を学ぶことを忌み嫌う．

米国の著作権法は何を保護しているのか？　アイディア自身ではなくアイディアの表現．ただし，外的な制約のために表現とアイディアが混合(merge)している場合は除く．プログラマーの行為の多くは外的制約に服している．訓練，プログラムの対象であるコンピュータ，使われる気取った語法の人工語，課題を単純化するのに用いられる他の既成の部品などなど．

それでは，とプログラマーは問う．「表現とアイディアの境界はどこに引かれるのか？」答はこんな感じだ．「仮差止命令の聴聞や公判廷の裁判官がここだというところだ」

著作権侵害はどのようにして認定されるのか？　実質的な類似性があり，オリジナル作品へのアクセスが認められ，コピーしたと訴えられた者の努力が足りなかった場合だ．

それでは，とプログラマーは問う．「たとえばX-windowシステムの機能を利用するコードを書いていて侵害しないようにするためには何をしなければならないのか？　これを使うと，新しいコードは自分が前の雇い主の下で書いたコードと類似してしまうのだが」答はこんな感じだ．「あなたが前に書いたどのコードとも不必要な類似性を持たせないようにしなさい．さもないと，専門家の手にかかってコピーした証拠にされてしまいますよ」

トレードシークレット法で何が保護されるのか？前の雇い主が秘密だと宣言し，秘密にしておき，世界で一般的に知られていないものだ．

それでは，とプログラマーは問う．「前の職場で思いついた，処理速度を早めるアイディアを再使用することはできるだろうか？」答はこんな感じだ．「できるかもしれないし，できないかもしれない．判断の分かれ目は，前の雇い主がそれに気づいて今の雇い主を訴えるかどうかだ」

特許法で何が保護されるのか？　前の雇い主が米国特許審査官を納得させることができた，その技術分野で通常の技能を持つ者にとって自明でなく，誰かが先にやっていないものだ．

それでは，とプログラマーは同じ質問をする．ただし，今度は特許についてだ．「前の職場で思いついた，処理速度を早めるアイディアを再使用することはできるだろうか？」答はやはりこんな感じだろう．「できるかもしれないし，できないかもしれない．もし前の雇い主が出願して特許を取得すれば，その後は今の雇い主は特許侵害に問われるだろう」

明らかに，いずれの場合も，プログラマーは自分にとっては論理必然的な行為を阻害し，何が合法で何がそうでないかについて一向に指針を提供してくれない法に直面する．プログラマーが，白と黒しかない世界に法が作り出そうとしている灰色部分にいらだち，混乱するのも当然である．

5．活発な競争

プログラマーは活発な競争を望んでいる．最も強く優れたプログラムのみが生き延びられるジャングルの掟を望んでいる．たとえそれがもう1つのジャングルの掟―最低の変換コストという経済の法則とぶつかることになってもだ．現状は維持されるべきだという旧弊な考え方とは無縁である．プログラマーは次の「キラーアプリケーション」，人間のコンピュータの使い方を根底から変えてしまうアプリケーションプログラム（スプレッドシートやインターネットのWorld Wide Webのような）を夢見ている．

プログラマーは次のキラーアプリケーションの創造者になる自由を望んでいる．

そして，もちろん，プログラマーは法がそれを作り出す彼らの権利を支援し，いったんそれができ上がればそのことで彼らが報われるようその権利を保護し，他人が簡単にそれをコピーして報酬の一部を横取りしてしまわないようにしてくれることを望んでいる．

米国のプログラマーが望んでいることが一貫していなければならないとは，誰も言わなかった．

6．良いソフトを書いて，世界中から注文が殺到すること

企業家精神に富むプログラマーはキラーアプリケーションを夢見る．それは彼（女）の才能を体現する創造物であり，世界を一挙に変え，その金銭欲をはるかに上回る金銭的報酬をもたらし，ビル・ゲイツ並みの本当の金持ちにしてくれる．

残念ながら，現実は，キラーアプリケーションはもはや現われそうにない．コンピュータプログラミング自体は製品の価格の10％でしかない．残りの90％はマーケティング，広告，流通とサポートのコストである．別の観点から言うと，販売戦略次第で平凡なプログラムがどんどん売れもすれば，すばらしいプログラムもマーケティングがまずければ売れ残りのパン同然

である．うそだと思うなら，今最もよく売れているオペレーティングシステムやワープロのプログラムを見てみればよい．

しかし，だからといって未来の企業家がやる気をなくしてはいないし，またそうであってはならない．なぜなら，彼（女）らこそは，米国議会が「その著作や発明について一定期間排他的権利を保障することにより，科学および有用な技術の促進を図る権限を有する」とされた「著作者や発明者」であるからである（合衆国憲法第一条第8節）．

米国のプログラマーは，訴訟というアメリカの悪夢を見ることなしにアメリカンドリームを夢見たいと望んでいる．彼（女）らは，自分の創造力のみにもとづいて成功し失敗する自由を持ちたいと望んでいる．彼（女）の考えたことが誰か他の者の考えだとか，彼（女）の書いたソフトが前の雇用関係から生まれた作品であり，前の雇い主のトレードシークレットであるとか，他人が特許を得た発明だとか，裁判所に決めてもらうことなど望んでいない．

米国のプログラマーが望んでいることが現行法と一致しなければならないとは，誰も言わなかった．

社会は何を望んでいるのか？

技術者の必要と希望をバランスさせることは社会全体の必要と希望をバランスさせることにつながる．これこそが，合衆国憲法の起草者たちが「科学および有用な技術の促進を図る」メカニズムを作るに動機となったことである．

社会は何を必要とし何を望んでいるのか？

社会が望んでいるのは，操作しやすく，信頼できる―作動し続ける―ソフトウェアである．これまでそのいずれも与えられなかった．米国のプログラマーは，その責任の一端は現行の知的財産権法にあると考えている．

著作権法は，類似性を避けるためにプログラマーに新しく考案し書くことを強いている．侵害として訴えられることを避けるために，全く別のグラフィックユーザーインターフェースを使わなければならない．たとえ第三者ソフト（たとえばX-windowシステムやマイクロソフトウィンドウズといった）を用いて書かれたものであっても，問題のプログラムが類似の機能を持つ場合にはそうしなければならない．さもないと，その類似性が盗用，部分的コピーの証拠とされてしまう．

こうした書き直しや強制された差異化はソフトウェアに対して2つの深刻な影響を与える．そしてそのいずれも社会全体，あるいは少なくとも社会の中でコンピュータを使わなければならない部分にとってはマイナスであると米国のプログラマーは考えている．第1に，書き直しの際に新たな間違いが組み込まれ，「バグ」の多い，エラーを多発するソフトウェアを生み出す．第2に，プログラムのユーザーは，2つのプログラムについて同じ特徴を備えた共通の知的なモデルを構築することができない．たとえば，Word Perfectとウィンドウズ版Wordのパラグラフナンバリングに関する同じ特徴は，根本的に異なっている（ここでは，これらのソフトのパラグラフナンバリングがあまりに機能しないので，精神異常者が作ったのだとコンピュータ業界でいわれているという事実はさておく）．この不一致が混乱といらだちを引き起こしている．共通の機能が共通の知的なモデルにもとづいていない．そのため，学習し使いこなすのが難しくなる．ユーザーは思考にソフトウェアを合わせるのではなくソフトウェアに思考を合わせなければならない．ユーザーではなく敗者（loser）だ．

要するに，現行著作権法は差異のための差異を強いている．類似性は制約条件からしばしば必然的に生じるが，プログラマー以外の人間にこのことを説明するのは難しい．結局，多くのプログラマーは，類似性を残してその結果生じるであろう事態を避けるために，多大の犠牲を払って類似性をなくする道を選択するのである．

米国のプログラマーにとっては，トレードシークレットもやはり問題である．なぜなら，それは経験を将来に生かすことを阻害するからである．多くのプログラマーが苦い経験を通じて思い知ったとおり，「巨人の肩に乗って先を見る」には，前の雇い主がその巨人でなければならないのだ．

トレードシークレットの問題は，競争相手が花形プログラマーを（まさに花形プログラマーであったそのために）引き抜いたことで怒り心頭に発した前の雇い主がトレードシークレットの指定をしていなくてもトレードシークレット侵害として訴えることである．社会は長期的な損害を被る．なぜなら，新しい雇い主とプログラマーは，予防策として，前の雇い主が後からあれは実はトレードシークレットだったと訴えることを警戒して，苦労して獲得した経験を決して活用しようとはしないからである．

特許がソフトウェア関連発明をカバーすることでも社会は損害を被ると，米国のプログラマーは主張する．現在の特許は，あまりにも単純で，普通の人間なら誰でも思いつくから，それ自体としては先行技術としては発見されないようなプログラムに関するアイディアにもしばしば与えられている．その結果，多くのプログラマーが怒って叫ぶ．「あんなのに特許を出すなんて！　あれは当たり前じゃないか．僕は1963年に思いついていたよ！」

ソフトウェア特許に関する最も重大な問題は，まだ若い，駆け出しのソフトウェア会社が，大変に創造的で革新的であるにもかかわらず，テロリストが飛行機に仕掛けた爆弾が爆発するというのと同じくらい恐ろしい警告文付きで出された特許によって吹き飛ばされてしまうことである．創造性と独創性には優れていても資金力に乏しい駆け出しの会社は，こうした状況にあっては離陸することができない．法廷で特許の有効性を争ったり，ライセンスを買ったり，交渉材料として自前の特許を申請したりすることはできないのだ．

結　論

米国のプログラマーは，法の変更を望んでいる．法が自分たちの二元的な世界にもっと合致し，自分たちがソフトウェアを作り出し，自分たちや他人の作品から学習するやり方にもっと合致するように変えられることを望んでいる．それは次のような変更でなければならない．

1．世界中で適用されること．なぜなら，インターネットが世界中に広がったため，たとえば，リバースエンジニアリングをヨーロッパでは禁止しながらインドや米国では許すということは無意味であるからだ．インターネットで一番問題なのはおそらく，それが地理的距離を無意味にしたのに，法は依然として管轄権と領域の境界の概念に大きく依存していることであろう．

2．法の対象とする技術と調和すること．著作権は伝統的な知的作品には合っているかもしれない．しかし，テキストとマシンの作用の両方を具現するソフトウェアであれ，ごく可塑的なデジタル化されたオーディオビジュアル作品であれ，デジタル作品には合っていない．同様に，特許法は機械類には合っているかもしれない．しかし，用いられるアイディアがしばしば他人のそれである，急激に変化するソフトウェアの世界には合っていない．

すでにインターネットや取引の世界では起きていることだが，根本的な問題は，プログラマーは懲罰を受ける恐れなしに法のまわりで自由に活動できることである．懲罰には摘発と把握が必要であるが，多くの場合，法も法律家も法を執行する人たちもそのいずれもできないでいる．ソフトウェアを法で有効に保護するためには，法とプログラマーの倫理が一致しなければならない．そのためには，法をプログラマーの尊敬を得られるように変えるとともに，プログラマーとエンドユーザーの倫理を改めて，摘発と懲罰を恐れて他人の知的財産を侵害しないのではなく，それが悪いこと，だからそうしないと考えるようにしなければならない．

そうした変革なしでは，米国のプログラマーは今後も現行の知的財産権法とその執行者を，そもそも保護できないものを保護しようとしているあきれてものも言えない連中だとみなし，相変わらず彼らの追求をかわし続けるだろう．この追いかけっこの勝敗は明らかである．なぜなら，法は常にプログラマーに遅れをとり，せいぜいのところ，無能な連中や敢えてつかまってやろうとする連中しか捕まえられないだろうからである．

（訳：中川淳司）