

Ms Kay Collins  
Director  
Intellectual Property & Competition Review Committee Secretariat  
Attorney-General's Department  
Robert Garran Offices  
National Circuit  
BARTON ACT 2600

Dear Ms Collins,

Please find attached comments on the Intellectual Property & Competition Review Committee's Interim Report.

I am a fourth year law student at the University of New South Wales. Last year I completed an Honours thesis entitled a Cost-Benefit Analysis of Intellectual Property Protection of Software.

Yours sincerely,

Elizabeth Melih

The Case for Special Treatment for Intellectual Property  
(Discussed on page 33 – 34 of the Committee’s Interim Report)

A thesis has been put forward that new property rights are defined and assigned as a response by agents to new cost-benefit possibilities. This may be the result of “changes in economic values, changes that stem from the development of new technology...changes to which old property rights are poorly attuned.” (Demsetz, H. (1967) “Toward a theory of property rights” *American Economic Review* Vol. 57 pp. 347–359, at 350)

Such a development can be seen to have occurred upon the invention of the printing press, which made the copying of written works easier and more cost effective. This resulted in new property rights being defined to cover written works. From the earliest of times intellectual products were always a special case, not fitting within any of the existing property right regimes that covered tangible goods such as land and chattels. As a result intellectual products, especially of an artistic nature, tended to be sponsored by the wealthy and elite as a means of ensuring their production.

The difficulty in assigning property rights for intellectual products stems from their intangibility and the difficulty that comes when attempts are made to control their use. For an idea, once it is given expression, becomes impossible to control and very closely resembles a public good. Ideas are non-rivalrous, one person’s consumption of the idea in no way diminishes its worth or the stock of it left for others to enjoy. In fact, one person’s “use” of a new idea may increase its value and use through what that person may add to it. For example, if a person produces an apple they may consume that apple thus preventing anyone else enjoying it as it is exhausted by one person’s consumption. On the other hand, if one person produces a recipe for apple pie, everyone in the world may enjoy that particular recipe for apple pie as long as they have access to a copy of it. Another person may come along and adapt this recipe to produce yet another apple pie recipe that varies in some way from the original. The more people use the recipe for apple pies, the more people will learn about it and wish to also try it, leading to even more consumption of this new recipe. While each individual copy of the recipe is itself rivalrous in nature, a cook in one kitchen cannot use the same copy as a cook in another kitchen somewhere else, the recipe itself is not. One cook’s use of the recipe in no way diminishes it, it remains whole and can be used by any number of cooks at the same time provided they each have a copy, and still it will be available for further use.

Further, ideas are non-excludable without legal means. It is generally impossible to prevent others from using an idea once it is made known. Unless the original innovator never tells anyone else about his/her new idea everyone who comes into contact with the idea may “use” it. The very act of learning about the new idea entails the learner thinking about it, using it to generate connections to his/her own experience and creating meaning for him/herself. As a result the idea may influence that person’s thinking and may form the basis of a new idea this person has.

Property rights therefore must be reassigned to build a legal fence, in the absence of the ability to build a physical one, to prevent the unauthorised consumption of intellectual products. Without this legal fence innovators will be unable to appropriate the benefits of their ideas and so the incentives of innovators will be affected (Demsetz 1967). Here we see the same problems and arguments arising that the property rights approach has attempted to address, focusing usually on physical resources. The only difference is that with intellectual products property rights are more difficult to fully define, assign and enforce than with physical resources.

### *Types of Knowledge Produced*

Once it has been recognised that intellectual products are a special case and do not fit into the property right regimes that govern physical goods, consideration must be given to the different types of knowledge society produces. Society’s store of knowledge is constantly being expanded as new discoveries and new ways of viewing old knowledge are being made. The law does not protect all the knowledge that is produced. Some of the knowledge produced can be viewed as general knowledge, of importance and interest but which has no specific commercial use and cannot be exploited for economic gain, such as the discovery of a law of physics like gravity. Knowledge that does gain the protection of the law can be seen as specific knowledge that has been sought and discovered to meet a particular need or create economic value. This type of knowledge often arises from specific research and development projects undertaken to develop a solution to a problem or to fill a niche in the market. This distinction is between applied and theoretical research. What is being examined in this paper is knowledge that belongs to the former category.

The defining, assigning and enforcing of property rights can thus be seen to be of great significance to economic activity. Property rights underlie each and every transaction that is undertaken and only when they are

fully defined and assigned will efficient resource allocation be achieved. It cannot, however, be simply assumed that property rights are fully defined and assigned, as in much of orthodox theory, rather goods must be considered individually and the property rights that support them examined in detail to ensure they are defined in the most appropriate way. This is especially true in the case of intellectual products, which do not fit easily within the property right regimes designed to cover tangible goods.

## Patents and Computer Software

(Discussed on page 41 – 42 of the Committee’s Draft Report)

### *Advantages of Using Patent Protection for Software*

#### 1. The Encouraging of Innovation Today and Innovation Tomorrow

Patents may be seen as sacrificing innovation tomorrow for innovation today through their protection of ideas. This view, however, fails to take into account the fact that patents do not give a blanket protection to ideas in their abstract form. Instead, patents protect an idea that has been applied to produce a particular product or process (Dam, K. (1995) “Some economic considerations in the intellectual property protection of software” *Journal of Legal Studies* Vol. 24 pp. 321–377.). The same idea may still be used in another application as long as it is not used to produce a product or process that copies the patented one. Consider the case of the computer program AutoCAD<sup>1</sup>, which has been developed and patented and makes use of a mathematical formula, a cubic spline, to draw accurate circles on a computer. The result of the patent is that no other software producer may use that formula in the same way to produce a program to perform the same function. The patent in no way prevents anyone else from making use of the same formula for any other application for which it is useful. Indeed, if another software programmer found they could use the same formula to perform some other unrelated, but equally useful, application they could seek a patent for their program and not be guilty of infringing the original patent.

Thus it can be seen that a patent, while allowing the holder to exploit the commercial gains of their innovation, does not restrict subsequent innovators from building upon the ideas of their predecessors. What patents do, instead, is prevent the mere copying of ideas which does not add to society’s store of knowledge but merely reallocates resources away from research and development to copying. Further helping to encourage future innovation is the fact that patents make ideas public. Innovators do not need to keep their ideas secret to enable them to appropriate their commercial value, patent protection gives them the monopoly over their exploitation, so the ideas are available to other innovators as foundations for future innovation. This is especially pertinent for the software industry where new advances are constantly being built upon earlier programs and ideas. By making ideas public patents allow for future

---

<sup>1</sup> See the Australian High Court decision in *Autodesk Inc. v Martin Dyason & Others [(1) (1992)] 173 CLR 330.*

development of programs that are more efficient and better serve users. Therefore patents can be seen as offering a positive incentive to engage in research and development, as innovators are able to appropriate the gains from their innovation, now and in the future.

## 2. The Application of Petty Patents to Software

Petty patents are issued for twelve months and may be extended for a further five years. By placing programs in the public domain sooner, petty patents allow them to become available to subsequent innovators as source material sooner and may stimulate new innovation.

### *Disadvantages of Using Patent Protection for Software*

#### 1. Patent Issue – Property Rights and Uncertainty

Patents are issued on the basis that the subject matter is a novel, useful and innovative product or process, which is non-obvious. These criteria ensure that what is protected is limited only to that which has resulted from research and development and has a potential commercial value. In the case of software, however, the expertise needed to assess applications on these grounds has been found to be lacking. Litigation is often used as a means of challenging patents, which have been issued, and, in many industries they are ruled invalid due to a lack of novelty or non-obviousness. In the software-related patent category this problem appears to be more pronounced with the result that innovative activity in the software industry is adversely affected (Dam 1995).

The issue of a patent is meant to create a property right in the intellectual product imbedded in that patent and create greater certainty in the market for this intellectual product. Greater certainty over property rights increases the incentives for agents to engage in research and development. This is because the payoffs to successful research and development projects become more certain, due to the ability to prevent others from copying the resultant software program.

However, if software patents are frequently challenged, as is the case, this reduces the certainty created by the patent issue. The decision to engage in research and development will become even more difficult; agents will still have to form expectations about the probability of success, of both the research and development project and the commercial viability of any resultant software, and of gaining patent protection. But now they will also have to form expectations about the likelihood of challenging

litigation by rivals. Patent challenges enter into the research and development investment decision as a cost, reducing the number of projects that will be deemed profitable and thus the number undertaken.

Resources will be reallocated into the provision of legal services to both protect and challenge patents increasing the costs of gaining a patent, decreasing their appeal and hence reducing incentives to engage in research and development, which may lead to fewer software innovations or higher prices to consumers. While it is true that incentives will always exist for agents to challenge patent issues to gain an economic advantage over rivals, a well functioning patent system should not require the judiciary to adjudicate on the validity of patents, except in the odd extreme case.

Faced with the prospect of expensive litigation to defend their patents, software innovators will face disincentives to seek patents, an expensive and time-consuming process in itself. Without patent protection innovators will be left to seek other forms of protection for their programs, and if these are found wanting, software innovation may fall to a level below that which maximises society's welfare.

To combat this problem the patent office may require a specialised software department staffed by persons qualified in the information technology industry. The costs of establishing such a department would be substantial and include the training of staff in the areas of both the information technology industry and patent law as well as general administration and related costs. These costs, however, would be more that offset by the reduction in litigation costs associated with patent challenges. Assuming the department was able to achieve greater certainty of property rights for software, patents should become more attractive to software producers increasing patent applications. The greater certainty over property rights should also encourage innovation in the industry further increasing patent applications. In this way the department should be able to recoup its establishment costs and fund its own administration.

The benefits associated with establishing such a department should also be substantial. Most significant is the reduction in uncertainty surrounding property rights in software that should encourage increased innovation. This is because the probability of appropriating the benefits of their software innovation will have increased, for software producers, through the increased ability to prevent copying by rivals that results from the more certain property rights. Incentives to invest in research

and development are also created by the reduced likelihood of litigation as this represents a fall in the costs of research and development rendering more projects profitable.

## 2. Costs Involved in Patent Applications

The time and costs involved in applying for, gaining, and maintaining patent protection serve as a disincentive to innovators to seek patent protection. The application process is a lengthy one due to the requirements of novelty, inventiveness and non-obviousness that must be met. Patent officials must search the prior art base, as well as patents already issued, in order to establish these criteria are met. This requires time and fees are payable for these services. Further, innovators cannot market their innovation prior to application for a patent. Therefore, an applicant must have the capital available to fund a patent application before they have sold a single unit of their innovation. This makes patents less attractive as a means of protecting innovations especially for smaller firms who lack the necessary funds needed for a patent application. Considering that the majority of software companies in the US, which enjoys a very large domestic market and relatively low rate of software piracy, fail to earn a profit in their first year, and for many profitability is achieved only after four or more years (PriceWaterhouse Coopers (1999) "Software and the global economy"), the expense of patent applications and the inability to market prior to application represent significant disincentives to patent application.

As a result, larger better-established firms, with greater access to both internal and external funds, may make greater use of patents. These firms may then exploit the monopoly power gained from patent rights to gain a greater share of the market through the selling of complementary programs or even give-aways of add-on programs. In the short-run this may seem to benefit consumers but in the long-run it may lock them into using inferior products and reduce choice while increasing prices by stifling competition as rival firms are unable to gain a foothold in the market. While the incumbent firm will have to engage in some research and development to retain its hold on the market this may take the form of only minor improvements to the existing programs and not involve any new innovation. This would see the developments of multiple upgrades that do not represent any real improvement for consumers but merely assure greater sales and increased profits further advantaging the incumbent firm.

With the added uncertainty that even if a patent is granted the innovation may fail to make commercial gains, a loss larger firms may more easily bear than smaller firms, having the funds to cover such losses and still continue to operate, provides strong disincentives to innovators to seek patent protection. Especially in the area of software, the time taken to grant a patent may be long enough that the program no longer has any worthwhile commercial value, due to the entrance of a rival program on the market or the movement of technology in a direction which makes the patented program obsolete.

The effect of these time and cost factors may be to discourage program innovators from seeking patent protection for their software inventions. If they also find that other forms of protection, such as copyright or trade secrets, are inadequate to protect their commercial interests and that without protection of some form they cannot appropriate the benefits of their invention, the result may be disincentives against engaging in research and development. As a result, innovation in the industry may fall.

### 3. The Duration of Standard Patents

The length of protection granted by patent has potential implications for future innovation in the software industry. Under the current legislation patents are issued, subject to payment of renewal fees, for twenty years. In an industry where product life is relatively short and new products are constantly being produced such extended life for patents can be argued to hinder future innovation and be of little use when the commercial value of a program is exhausted well before the expiration of the twenty year period.

### 4. Piracy

Piracy occurs when another agent makes copies of a particular software program and then sells these copies as originals (Dam 1995). This practice has implications for incentives as well as consumer welfare. If such copying is allowed to occur unchecked software innovators will have less incentive to innovate, as they will be unable to prevent others from appropriating the benefits of their innovation. As a result innovation may decrease and so choice for consumers will become limited.

Further, consumers will purchase software they falsely believe to be the genuine product of the innovator and so will suffer losses in the form of

the ancillary services they will not have access to. Resources will be expended by consumers in an attempt to ensure the product they are purchasing is genuine, resources that should not be required for this purpose.

Patents should be able help to prevent piracy, as it would allow innovators to seek injunctions against infringers. With penalties for infringement in place incentives to pirate would be reduced leading to a decrease in piracy. This argument relies on the ability of innovators to successfully detect and prosecute for infringement which, in turn, relies on the strength and competence of the legal system in place. If innovators are successful in detecting and prosecuting infringers and penalties are sufficiently high then patents would provide a means of deterring piracy. As a result resources would be allocated more efficiently as they would not be employed wastefully in the production of copied software, a practice that would lead to overproduction and thus a misallocation of resources.

While copyright, the dominant form of protection for software currently, should be a means of preventing piracy the extent to which this practice occurs suggest that it is not an adequate measure. According to an International Planning and Research Corporation study, in 1997, four out of ten business software applications were pirated resulting in losses of \$11.4 billion<sup>2</sup> world-wide, this figure being an increase on the 1996 figure of \$11.2 billion. Figures for Australia and New Zealand, in 1996, show that PC business software piracy rates are approximately 32.6 percent costing the industry \$157.7 million (PriceWaterhouse Coopers 1999). The fact that copyright has been unable to prevent acts of piracy suggests that patents too may not be adequate to address this issue.

### *A Software-Specific Patent*

The duration of a standard patent, and even the extension period for a petty patent, is currently longer than the commercial life of most software programs. The initial twelve months of the petty patent, however, may be insufficient to cover the commercial life of a program so programmers face the prospect of further expense in having the period extended. What is required to provide incentives for research and development in software is the setting of an optimal patent length for programs. By having a patent issued specially for software the duration can be set such

---

<sup>2</sup> Figures quoted are in US dollars.

that the programmer can exploit the commercial value of his/her program and when this is exhausted the program can become available to subsequent programmers as source material.

The optimal duration of a software-specific patent could be modelled as the maximising of social welfare as the difference between the incentive to invest created by patents and the social costs of the increased investment in research and development. A longer patent life would increase monopoly profits and the incentives to engage in research and development but would also increase the dead weight loss associated with the monopoly created. The optimal patent life would be such that it yielded the creation of a number of software programs where the benefits to society from the last program produced exactly equalled the cost of producing that program.

Fees for such a patent could be set so that innovators know with certainty the cost of obtaining protection. As the legislation currently stands a programmer may apply for a petty patent, and depending on the market response to the invention, may or may not seek to extend the patent. The same is true for a standard patent; renewal may or may not be sought depending on market acceptance of the innovation. Thus, when deciding whether to engage in research and development, not only must programmers form expectations about the possible returns from any program created but also form expectations of differing costs of gaining patent protection contingent on the returns earned. This makes assessment of the profitability of investment in research and development even more uncertain and may provide disincentives to engage in research and development.

A software-specific patent could counter these disincentives by making the costs of protection more certain and giving a definitive duration for protection. The optimal duration for a software-specific patent would be set such that it maximises current innovation subject to resource constraints, capital, both physical and human, and technology, and future innovation discounted back to present values. The optimal duration for a software-specific patent is likely to fall somewhere between the twelve months of the petty patent and the twenty years of the standard patent.

By introducing a software-specific patent innovation in this industry will be encouraged now and in the future. Programmers will have greater incentives to engage in research and development, as there is greater certainty over property rights in software and the duration of protection

offered. As well, commercially exploited programs will become available to subsequent innovators, as source material, sooner.

The many disadvantages of applying patent protection to software outweigh the few advantages proving patents to be an inadequate form of protection. Introducing a software-specific patent, an exercise that would entail considerable expense but ultimately yield many potential benefits, may solve these problems. If intervention in the software industry is to continue to rely on existing forms of intellectual property protection the introduction of a software-specific patent provides a more efficient mode of protection than copyright or general patent legislation.

## Computer Software and Copyright Protection (Discussed on page 83 – 84 of the Committee’s Interim Report)

### *Advantages of Using Copyright Protection for Software*

Copyright offers some advantages for the protection of software programs in the form of the speed, ease and costs associated with gaining protection. Under the current legislation copyright protection is automatically granted to a work, as defined under the Act, a category to which computer programs belong. There is no requirement for registration, any fees to be paid to establish or maintain protection and no need to hire copyright experts unless a dispute arises. Due to these factors copyright can be seen as an effective way of promoting innovation, as innovators need not add resources needed to gain protection for any successful innovations when calculating expected costs and returns of engaging in research and development.

However, when the protection that is offered is inadequate, as evidenced by the many disadvantages to using copyright for software, discussed below, the fact that it is a quick and easy method of gaining protection is of small comfort. If there were registration requirements and fees it is likely that copyright would not be as widely used as currently is the case, PriceWaterhouse Coopers (1999) report that eighty-three percent of respondents to a US survey used copyright for software protection in 1997.

### *Disadvantages of Using Copyright Protection for Software*

#### 1. The Creative Nature of Copyright and The Utilitarian Nature of Software

Copyright was first envisaged as protection for creative works, such as works of fiction, art and drama, and so is ill suited to the protection of software that is primarily a utilitarian work. By placing software in the category of a literary work, under copyright law, it has been effectively placed on the margin of protection. The working of the merger doctrine, whereby a work is deemed unprotected if the idea and expression merge, further compounds this problem. For utilitarian works such as software it is often difficult to separate the idea and expression leaving software with even more marginal protection under copyright.

#### 2. The Idea/Expression Dichotomy and Its Effects on Innovation Today and Innovation Tomorrow

Kenneth Dam (1995) states that copyright law should be drafted in such a way as to achieve an efficient balance between the innovation of today and the innovation of tomorrow. He believes that the doctrine of copyright law that separates ideas and expression ensures this goal is achieved by protecting only the expression of ideas while leaving the ideas themselves unprotected. This leaves every subsequent generation of computer software creators to freely borrow from and use the ideas of their predecessors. Any computer programmer may write a word processing program but none may simply copy the source code of Word.

The strength of this argument rests on the interpretation the courts place upon the ideas/expression dichotomy. If courts take a narrow view of the meaning of expression the protection offered might be inadequate to ensure innovators can appropriate the benefits of their software innovation, the very reason for copyright law in the first place. This is the problem being encountered by software programmers today due to the utilitarian nature of software and has seen them turn to other forms of protection, such as patents.

On the other hand, if the courts expand the meaning of expression such that almost everything is protected, and thus allow expression to encroach into what is really the realm of ideas, future innovation will be compromised. With little or no common property in ideas subsequent innovators will have difficulty proving they are not in breach of copyright and so incentives for subsequent innovation will decrease (Dam 1995).

### 3. The Idea/Expression Dichotomy and Non-Literal Elements of Software Programs

The idea/expression dichotomy is very important when considering the non-literal elements of a software program. The non-literal elements comprise the program structure, which is the way in which the program code is organised and ordered, and the 'look and feel' of the program, or the user interface, which refers to the presentation of the program as perceived by users (Hunter, D. (1993) "Limitations on the proposals for reform of non-literal copyright and reverse engineering of computer software" *Journal of Law and Information Science* Vol. 4 No. 2 pp. 243–264.). In its draft report into computer software protection the Copyright Law Review Committee found that these non-literal elements "lend themselves to protection by copyright law in the same manner and to the same extent as traditional literary works." (CLRC 1993, pp 111-12) Despite the Committee's finding non-literal elements are even further

along the margin of copyright protection than the literal elements due to the fact they are often fall foul of the merger doctrine.

Copyright of non-literal elements has important incentive effects making the distinction between ideas and expression crucial. Program structure may be thought of as either an idea or its expression, it is more abstract than code such that it may be defined as merely the idea for the program but it may also be the expression of that idea. For developers of software programs, especially in today's world of object orientated languages, computer aided software engineering tools and structured programming techniques, their program structures represent important expression which they wish to protect. Without protection others may copy their programs and apply them to other systems or else produce products in competition to them without needing to invest in research and development, as they can simply copy the original programmer's structure (Hunter 1993).

Both of these represent significant problems as they provide disincentives to the creation of new software programs. As copyright has not proved adequate in the protection of literal elements of software it is likely that it will also prove inadequate if extended to the non-literal elements. The working of the merger doctrine is likely to ensure that much of what is contained within the non-literal elements will be found to merge expression and idea and thus not gain protection.

#### 4. The Duration of Copyright Protection

The length of protection granted by copyright has potential implications for future innovation in the software industry. Under the current legislation copyright subsists in a work, the category to which computer programs currently belong, for fifty years after the death of the author. In an industry where product life is relatively short and new products are constantly being produced such extended life for copyright can be argued to hinder future innovation. By revising down the duration of protection the incentives for innovation created by copyright will not be compromised but may actually be enhanced. By placing programs in the public domain sooner they become available to subsequent innovators as source material sooner and may stimulate new innovation.

As the law currently stands copyright may subsist in a program, which has become of only limited commercial value but, if allowed to be freely copied, may produce a new program with a greater commercial value. While it may be argued that infringement of copyright may not be sought to be punished by the copyright holder, in such a situation, if a

subsequent innovator were to use such a program to create a new one the original innovator is unlikely to simply allow this second programmer to appropriate all the benefits. By setting a shorter length for copyright for software both innovators could be given the opportunity to appropriate the benefits of their innovations.

If the duration of copyright in software were set at an optimal level the original innovator would gain the maximum benefits from his/her innovation. Upon expiration of this optimal time subsequent innovators could freely use the original program to create new programs, there would be little incentive to simply copy and sell the original program for its commercial value would have been exhausted. Instead, innovation would occur which builds upon that of the original program. Copyright for this new program would then belong to the subsequent programmer allowing them to appropriate the benefits associated with their innovative activities. Thus it can be seen that if the duration of copyright in software is adjusted to a more optimal level future innovation may be encouraged.

The optimal duration for copyright protection of software could be determined as an optimisation problem, where social welfare is being sought to be maximised. Total welfare would be an increasing function of the number of programs produced and the consumer and producer surplus per program and a decreasing function of the total costs of creating a program including the costs associated with the copyright system (Landes, W. M. and Posner, R. A. (1989) "An economic analysis of copyright law" *Journal of Legal Studies* Vol. 18 pp. 325–363.). The number of works produced is a function of the copyright protection afforded and varying the duration of protection will alter the number produced. The cost of the copyright system is also a function of the duration of protection such that solving the equations for the number of works and costs of copyright simultaneously should yield a value for the duration that gives the highest number works at the lowest cost. This can then be used to determine total welfare. Comparisons between various times and total welfare will enable the optimal duration of copyright protection to be determined.

The problems associated with copyright of software stemming from it being designed with creative works in mind, as opposed to software's utilitarian nature, its over-long duration and the current piracy rates indicate that it is not the most efficient method for software protection. If the benefits of faster growth in the software industry are to be enjoyed alternative forms of protection for software need to be considered.

A Sui Generis Law for Software Protection

Considering the numerous disadvantages of relying on copyright and/or patents to protect software and encourage its production an alternative form of protection must be sought. One alternative is to provide a sui generis law for software protection, that is, a law that provides protection only to software and removes software from the areas of copyright and patent.

#### *Advantages of a Sui Generis Law*

Providing a sui generis law for software, tailored to the specific needs of the industry and its product, has the advantages of removing all the disincentives created under the current modes of protection and reducing the uncertainties inherent in them. Property rights should become more certain and the scope of the protection offered should be clearer allowing agents to make better informed research and development decisions.

The combined effect of these benefits should be an increase in software innovation, more in line with the level that maximises social welfare. The potential job creation and growth in the software industry is already quite high. Some of the increased employment opportunities and revenue will come from increased research and development and the introduction of new programs. To achieve a reduction in piracy levels requires a new and improved method of protecting software such as a sui generis law created specifically to meet the needs of the industry.

#### *Disadvantages of a Sui Generis Law*

A potential disadvantage of adopting a sui generis law for software protection is the international repercussions that would accompany such a move. As a signatory to the Berne Convention and TRIPS and as a member of the WTO Australia has an obligation to meet the requirements of these agreements.

The fact that Australia is a signatory to international agreements that specifically provide for the protection of software under copyright means that if Australia, by itself, decided to enact legislation that removed copyright and patent protection and replaced it with a sui generis law she would be in breach of these agreements. The effects of such a breach could see Australia subject to appeals to the WTO and action by it as well as refusal by other nations to trade.

Apart from these implications for Australia's international status a move to introduce a sui generis law for software protection would also impact upon individual software innovators. Where before they could rely on automatic protection in all Berne Convention member states this would no longer be the case if copyright in software was abolished in Australia. This could potentially have a negative effect on the export earnings of local software producers and, in turn, on the incentives to engage in research and development.

The decision to introduce a sui generis law to protect software in Australia must, therefore, consider not only the domestic effects but also the international implications of such a move. Otherwise the potential exists for the advantages gained domestically to be undermined by the intentional disadvantages. Even if the local industry was found to have a relatively insignificant export component these international disadvantages could be quite high if the introduction of a sui generis law reduces the potential for future expansion into export markets and/or results in a negative international response.

To offer true advantages over existing forms of protection requires a sui generis law to be adopted within the WTO and as part of international agreements such as TRIPS. Australia, as a small open economy, cannot introduce such a law alone and expect to reap any real benefits.

### *Policy Suggestions for a Sui Generis Law*

The minimum required of a sui generis law for software protection, to justify its adoption, is that it addresses the problems inherent in copyright and patent. The following is a potential model of such a law but other versions are possible. Before a sui generis law is introduced a comparison between alternative models should be undertaken to ensure the one adopted yields the maximum benefits for the software industry and economy as a whole. A comparison of this type, however, is beyond the scope of this submission.

One model of a sui generis law could see the adoption of an instrument closely resembling a patent. This software patent, for want of a better name, could be of shorter duration, than standard patents, with a more streamlined application process. The requirement for registration would be retained, with the first to apply rule operating, that is, the first person to apply would gain protection rather than the first person to produce the new program. Registration fees could be kept to a minimal as the novelty requirement could be reduced to a comparison of only registered

programs. This would mean that applications could be more speedily processed, especially if a dedicated department administered the system.

The increased speed and reduced fees should make the software patent accessible to all innovators, such that restricting novelty to only prior registered programs should not unfairly prejudice any innovator as all innovators would have equal access to registration. A standard application form could even be adopted, which was simple to complete and could be easily completed and lodged upon production of a new program making registration even more attractive. The ability to market could be set at the date of application giving innovators quick access to the market.

By altering the criteria for the grant of the patent to more closely resemble the purpose of software the problems associated with the conflict between the utilitarian nature of software and the creative nature of copyright could be addressed. Software would no longer be classed as a literary work rather it would be recognised as a utilitarian work. As the patent would apply only to software the requirement for a post solution outcome, under standard patent law applied to software, could also be removed. This will not jeopardise innovation in other sectors or remove mathematical algorithms from general use, as all that is protected is the particular piece of software. So long as subsequent programs are not identical copies of registered ones they can use the same algorithm, as can agents in all other applications to which the algorithm can be put. The non-literal elements could also gain protection, through recognition of the utilitarian nature of software, with a copy of the program's structure and a description of its user interface to be attached to the application.

The information that accompanies an application, therefore, would ensure that the details of new programs became public and available to produce new programs. This ensures the constant improvement in software through the layering of one innovators work upon that of his/her predecessors.

Finally, the patent defence of independent discovery could be retained so that innovators are not penalised by the working of the first to apply rule. This represents an advantage over copyright where no such defence exists.

This discussion of a sui generis law for software represents a canvassing of only some of the issues involved. Other models are possible that may be superior to the one suggested here but what this model illustrates is that a sui generis law has the potential to address the disadvantages inherent in copyright and patent and hence increase the incentives to create software. The actual advantages and disadvantages of any sui generis law adopted will depend upon the specific form it took and on the international environment into which it was introduced.