Submission to the Australian Council on Intellectual Property
Review of the innovation patent system

Open Source Industry Australia Ltd

Lodged 4 October 2013

Contents

About OSIA

OSIA represents & promotes the Australian open source industry by:

- Ensuring that the Australian business, government and education sectors derive sustainable financial and competitive advantage through the adoption of open source and open standards;
- Helping Australian Governments to achieve world leadership in providing a policy framework supportive of open standards and of the growth and success of the Australian open source industry; and
- Ensuring Australia's global standing as the preferred location from which to procure open source services & products.

OSIA's members are organisations in Australia who invest in or build their future on the unique advantages of open source software. For further information, see the OSIA website at http://osia.com.au.

Contacts

For further information in relation to this document, contact:
OSIA Chairman, Jack Burton <chairman@osia.com.au >;
OSIA Director (Domestic Markets), Mike Hideo <onshore@osia.com.au >;
OSIA Director (Export Markets), Don Christie <export@osia.com.au >; or
OSIA Director (Education), Daniel Jitnah <education@osia.com.au >.

Copyright

1 Executive summary

OSIA welcomes the opportunity to comment on the present review of the innovation patent system and thanks ACIP for providing that opportunity. Patent law, particularly in relation to the patentability of computer software, has been a matter of concern to OSIA for some years.

1.1 Recommendations

Preferred option

Of the three options presented in the ACIP options paper, OSIA's preference is for Option B ("abolish the innovation patent system"), although we note that even Option B would still leave outstanding issues in relation to standard patents on computer software (see below).

Comments on Option C

From OSIA's perspective, for Option C ("change the innovation patent system") to provide any substantial improvement over the status quo, it would need at a minimum to include the proposed reforms described in Section 5.6.8 ("Exclusions") of the ACIP options paper — most importantly, the proposed exclusion for computer software.

OSIA also sees merit in several of the other reforms proposed in Option C, but notes that even a complete implementation of all reforms proposed in Option C would not by itself address the present issues to the extent that Option B would.

Unsuitablity of Option A

In OSIA's view, Option A — the ubiquitous "do nothing" option — is unacceptable. ACIP's own findings to date have identified the existence of a range of deficiencies in the innovation patent system. In this submission, OSIA affirms the existence of some of those, and notes the existence of further deficiencies apparent specifically in relation to the industy sector our members operate in. Given the broad range of deficiencies, and their negative impact on technological progress in Australia, clearly the status quo ought not be left to persist.

1.2 Supplementary recommendations

The options paper refers specifically to innovation patents. However, the question of the patentability or otherwise of computer software is one that is relevant also to standard patents.

OSIA's position on this matter is simple: we are opposed to software patents.

We take this position because software patents damage the interests of the vast majority of the Australian software industry, including and especially our members in the Australia open source software sector.

OSIA calls upon the Commonwealth Government to amend the *Patents Act 1990 (Cth)* with provisions explicitly excluding computer software from the scope of patentable subject matter (in relation both to standard patents, and to innovation patents if those are to remain available).

The recently passed *Patents Act 2013 (NZ)* provides a good model from which to work.

1.3 Broader implications

The improvements brought about in New Zealand by the passage of the *Patents Act 2013 (NZ)*, in particular the exclusion of computer programs from patentability, present an ideal opportunity for Australia to align its patent regime with New Zealand's.

In his third reading speech on the Bill, NZ Commerce Minister Craig Foss mentioned plans to devlop "a single regulatory regime for New Zealand and Australia patent attorneys"[1]. The benefits of introducing a common regulatory regime are clear.

We note that alignment of Australia's patent regime with New Zealand's could well serve to facilitate the development of the common regulatory regime for patent attorneys.

[1] (27 August 2013) 693 NZPD 12948

2 General observations

2.1 Software as patentable subject matter

2.1.1 Patents & copyright

Computer software stands alone in Australia as the only class of work to be subject to both patents and copyright. The existence and parallel application of two irreconcilably disparate manners of granting limited monopolies in relation to a single class of work naturally gives rise to some level of confusion and uncertainty even within the computing industry — and to a far greater extent in the market beyond.

This dual approach is overly burdensome not just on the industry itself (who as a result face greater compliance costs than should otherwise be necessary, thereby providing a disincentive to innovation, which defeats the original purpose of having a patent regime) but also on the industry's clients: in recent times we have seen software patent-holders strategically elect to assert their patents against end-user organisations (who often have little or no ability to defend such claims), rather than the developers of the allegedly infringing software (who at the very least would be somewhat better placed to defend claims of infringement).

The solution, clearly, is that computer software should be subject only to one regime or the other. OSIA's view is that copyright provides the most appropriate legal mechanism to encourage the development of innovative and useful computer software.

We hold that view for the reasons outlined below, but also because copyright vests automatically in the author of a computer program, with no effort or cost required, whereas the process of patenting a computer program is a complex and expensive one — which puts individuals and SMEs who develop software at a clear disadvantage.

2.1.2 The nature of computer sofware

Computer software, in its purest form, consists solely of algorithms. In the real world today, almost all computer software also contains data, alongside those algorithms. Those data may take many forms, including text, images, audio/video recordings, or raw mathematical, financial or geospatial data.

When considered in isolation, all those forms of data have traditionally been the subject matter of copyright (except perhaps where they consist of mere collections of facts, which nobody ought to "own").

It stands to reason therefore that copyright, not patent law, should apply regardless of whether those forms of data are published alone, or bundled with a set of algorithms – the alternative would be absurd.

Similar reasoning applies to the algorithms themselves, although to those outside the computing profession it may not be quite so obvious.

An algorithm, in essence, is a textual description of an abstract concept – much like a text-book on mathematics, a treatise on philosophy or a theological tome.

Courts and legislatures in various jurisdictions have on occasion attempted to classify the algorithms that comprise computer programs as "mathematical" or "non-mathematical", in order to regard the latter as patentable and the former as not.

However, such a distinction is completely devoid of meaning. The eminent computer scientist Donald Knuth put it best in his open letter to the US Patent & Trademark Office of 1994:

> *I am told that the courts are trying to make a distinction between mathematical algorithms and nonmathematical algorithms. To a computer scientist, this makes no sense, because every algorithm is as mathematical as anything could be. An algorithm is an abstract concept unrelated to physical laws of the universe.*
>
> *Nor is it possible to distinguish between "numerical" and "nonnumerical" algorithms, as if numbers were somehow different from other kinds of precise information. All data are numbers, and all numbers are data. Mathematicians work much more with symbolic entities than with numbers.* [2]

2.1.3 The nature of software development

The computer software industry is quite unlike most other industries today. Computing, as a profession, is still in its infancy, with commercial computing have been a reality for less than 60 years.

Enormous progress has been made in the field of software development during that short time. Much of that progress has been achieved iteratively — through programmers building upon, extending and improving the published ideas of other programmers.

[2] http://progfree.org/Patents/knuth-to-pto.txt

Sometimes that progress is facilitated by an enlightened software developer releasing his software under a license designed to encourage the creation of derivative works (this is certainly the case for the open source software sector, but not always in other sectors of the software industry).

But nearly *always* that progress has been achieved by software developers building upon the *ideas* — the "abstract concepts" of which Knuth writes — of earlier programmers.

This is not, as some outside the computing profession may imagine, a mere manifestation of greed. Rather, it is a direct result of what computer programs are — algorithms — and the process by which a program can be developed for and executed on a computer.

Progress in software development simply cannot occur any other way. Starting from nothing, for computer software to exist and be useful, the following occurs:

1. The physical components that comprise a computer must be designed and fabricated. Most relevant amongst these are one or more processing units — computational devices which are able to execute a well-defined set of primitive instructions. The computer itself, consisting of an organised collection of those physical components, must also be designed and fabricated.

2. A program can be written to execute on the computer. Such a program can consist only of some combination of the primitive instructions in the set defined by the processing unit(s), and any more complex instructions that the program itself defines using only those primitive instructions — much like a literary work can consist only of known words in the language in which it is written, or new words defined by the author using only existing words.

3. Originally, all software was written as is described in Step 2. Later, programs called "assemblers" and "compilers" were developed to translate useful, more complex instructions into those primitive instructions, to enable other programmers to write more complex programs quickly and easily, in what are known as "higher-level" computer languages — much like those engaged in other professions, such as law, medicine or accounting, use lanugage specific to their domain of expertise to communicate complex concepts quickly without having to resort to explaining everything from first principles. Today, almost all software is written in this manner — often involving several levels of indirection between the instructions that the programmer writes and those that the processing unit(s) execute.

The above is of course presented simplified, but sufficient detail remains to address the question at hand.

Clearly, the computers and their components created in Step 1 are physical engineering artefacts, which are rightly considered patentable subject matter — that is not in dispute.

But it should be equally clear that the programs (algorithms) created in Steps 2 and/or 3 above are not — if anything, they bear closest resemblance to literary works.

It is instructive to note the implications of the latter two steps.

Since every program written in a high-level computer language must reduce to a set of instructions in a lower-level computer language, and the latter are defined *by another computer program*, it is not possible to produce a computer program in the efficient, productive manner that the industry as a whole uses today, without building upon the *ideas* of another computer programmer.

That paradigm is not by any means limited to the use of compilers and assemblers, although it is simplest to demonstrate in those cases.

In the modern world, no computer program exists in a vacuum. To be considered useful, a program must interact with other programs, written by other programmers — whether interfacing with the other program directly (e.g. across a data network), interfacing with it indirectly (e.g. by reading data from, or writing data for, another program in a defined format), controlling it / being controlled by it (e.g. for process automation purposes), analysing what it is doing (e.g. to audit its security implications), or depending on it in some other way.

All of those things can be done without infringing the other program's copyright. But if the other program is patented, many of those things can be exceedingly difficult, or outright impossible, to accomplish without infringing on its patent.

From that fundamental truth it is simple to see that granting a monopoly on the *idea* embodied in a computer program (as software patents do) must always be counter-productive.

Copyright, on the other hand, provides a far better balance between the public interest in ensuring software developers are free to build upon the ideas embodied in programs written by others, and the public interest in encouraging software developers to publish new and innovative software (by preserving their freedom to set the terms of the license under which their specific implementation of that idea may be distributed).

2.1.4 Impact on the Australian open source software industry

The arguments above hold true with respect to all software, whether open source or not. But software patents also threaten the open source software sector in an additional manner.

The vast majority of companies in the Australian open source software sector do not "sell" the software they develop in the traditional sense. Rather, their business models are based on selling services in relation to the software they develop and distribute — the software is provided under license, and the licenses need to carry terms that confer or give rise to a specific set of rights and obligations in order for the software to be considered open source, but the payment of a license fee is not one of those required obligations. In the vast majority of cases, there is no license fee payable. Even in those rare cases where there is, it is payable only by those who obtain the software directly from the licensor — and the right of the licensee to redistribute the software is one of those rights essential to the very definition of open source software[3].

Whilst any company which employs enough sufficiently skilled computing professionals can also sell services in relation to any open source software, the companies that develop it often carry greater gravitas in the marketplace — which in turn encourages them to develop and release more open source software.

Patent advocates often make much of the fact that many standards-setting bodies require that patent-holders license any patents on subject matter contributed to their standards to all comers on reasonable and non-discriminatory ("RAND") terms. Companies that develop and sell closed-source software might take some solace in such arrangements — the result of which is merely to inflate the license fees they charge.

But what of companies who develop and distribute open source software? A royalty payment cannot be calculated on a installed base of indeterminate size — so a traditional patent license based on royalties is not workable for open source software. The alternative is a royalty-free license offered in exchange for a one-time fee. To fund such a fee, an open source software company would need to inflate the prices it charges for its *services*, whereas its competitors providing the same services would not (unless they also distributed the software). That would substantially dilute (if not remove altogether) the natural advantage the developing company enjoys in the services market — thereby *discouraging* them from investing in further development (which is in direct conflict with the original purpose of patents).

So when an open source software company is approached by a software patent-holder (even one with RAND obligations) asserting infringement, it has only two options — go to court and invalidate the patent; or exit the market altogether. For open source software companies, the traditional third option (enter into licensing negotations) is simply not feasible.

Larger and well-resourced software companies amass patent portfolios to threaten others and/or enter into cross-licensing agreements. The premise behind this tactic is simple: if you sue me, then I will sue you. This stalemate option is not available to SMEs who simply cannot afford to buy the protection their more well-resourced competitors can. They, more rightly, put their resources back into innovation and, in turn, the broader economy.

The open source software model is an enabler for the vast majority of new digital businesses in Australia and around the globe. Software patents undermine this model and thus restrict the economic opportunity that has been ushered in by the dominance of open source software.

2.1.5 International perspectives

The New Zealand experience with Microsoft's XML document patents

The New Zealand Open Source Society (NZOSS) has had first hand experience of large multinational companies using patents to protect their monopolies in New Zealand. Since 2004 NZOSS has been involved with action opposing two patents covering the use of XML for word processing documents. If it were not for this opposition to these patents the multinational software company concerned would have been able to threaten all potential rivals with patent infringement.

Microsoft's office productivity software is one of their primary revenue streams. Open source software like LibreOffice[4] is now threatening that revenue flow. One of the primary means Microsoft has of preventing people moving to other products is to control the file format used to save documents.

Microsoft applied for two patents around XML use in word processors in New Zealand. Several competing word processors were already storing XML documents when Microsoft filed it's patents in 2002. NZOSS opposed both of these patents on the basis they were trying to patent things that clearly had prior art and were obvious. One patent was restricted as a result and Microsoft was forced to withdraw the second, which it did in all jurisdictions[5].

Had these patents not been opposed Microsoft would have been able to threaten software developers offering word processing applications that used XML to store documents. These patents could have prevented other

[3] Open Source Initiative, *Open Source Definition*, Criterion 1. See http://opensource.org/osd

[4] See http://libreoffice.org

[5] See http://nzoss.org.nz/content/nzoss-wins-patent-opposition

companies from delivering applications capable of interoperation with Microsoft's XML document formats. In summary patents are being used by large monopolies to protect themselves from competition; the precise opposite of the initial intent of patents. No-one has the resources to oppose all software patents that are overtly broad or obvious. By excluding software patents we will be denying multinational companies the ability to unfairly protect their monopolies using legal mechanisms.

It is to be noted that the Australian Government is aware of this and have been encouraging the use of open formats by software vendors (AGIMO stipulates the open ODF format, for example[6]). However, if Microsoft had gained patents sufficiently broad it would have been able to prevent any competing product from using the same general approach; that is using XML for the purpose it was designed, storing interoperable documents.

Gowers Review of Intellectual Property

In 2006 the UK Government commissioned an independent review of intellectual property Law[7]. This was an extensive review addressing many issues around software. In relation to the European approach to software patents the report says:

> *Currently, only if computer programs have a 'technical effect' can they be patented in Europe. However, there is little authoritative definition of this phrase, and its scope is widely contested. This has led to some computer programs being patentable in one country, such as the USA, but not patentable in the Member States of the EU. Recent attempts to agree a Directive to harmonise the law relating to software patents was rejected by the European Parliament in response to fears about the negative implications that a US-style system for software patents would have on innovation.*[8]

With regard to the effect of software patents on the innovation in software the report identifies issues with patents being used to protect incumbent manufacturers:

> *Cohen et al. show that 65 per cent of firms in complex technologies use patents as a trading strategy to licence a technology (while only 12 per cent use patents as a fence). In simple technologies, 28 per cent of firms use patents as a trading strategy to licence a technology, while 46 per cent use patents to fence off an area to competitors and provide more intellectual space to innovate. This strategy can be problematic in areas where thousands of patents are used in the design of new products especially in the electronics and software industries.*[9]

The report provides an example of how broad software patents can stunt competition:

> *Blackboard, a US maker of online learning management systems, recently took the academic community by surprise when it announced it had been granted a broad patent in the USA. The patent covers 44 claims related to learning management systems and implicated infringement by many other products on the market. On the same day that it publicly disclosed its patent, Blackboard started a patent infringement suit in a Texas court against Desire2Learn. Many companies that have been working on educational software are now concerned that Blackboard will either sue for infringement or enforce complex and expensive licensing agreements.*[10]

This could impact the most successful learning management system in the world, the Australian made, open source system, Moodle[11].

The Gower report made the following recommendation in relation to software patents:

> *Maintain policy of not extending patent rights beyond their present limits within the areas of software, business methods and genes.*[12]

[6] See http://agict.gov.au/blog/2013/05/28/views-sought-annual-review-common-operating-environment-policy

[7] Andrew Gowers, *Gowers review of intellectual property*, HM Treasury, 2006. Available at: http://www.hm-treasury.gov.uk/d/pbr06_gowers_report_755.pdf

[8] *Id.*, s. 2.35, p. 33.

[9] *Id.*, s. 3.19, p. 38.

[10] *Ibid.*, Box 3.1.

[11] See http://moodle.com

[12] Gowers, *op. cit.*, Rec. 17, p. 7.

2.1.6 Patent trolls and submarine patents

A patent troll is an organisation which produces no actual products, but rather operates by purchasing patents and then approaching companies that do manufacture products and negotiating license agreements. The permissive patent system in the US has lead to many patents being awarded that are of dubious quality.

For a manufacturer of a specific tangible product it may be possible to search for related patents in order to avoid infringement or to license the patent. Software however is generally far more complex than manufactured products. In addition software can be written by individuals and SMEs who do not have the resources to ensure that their software does not infringe any of the thousands of software patents.

An example of this was the case of Canadian firm Research In Motion (RIM), makers of the Blackberry, infringing a patent belonging to NTP for mobile email. The maker of the Blackberry e-mail device has reached a $612.5M (£349M) settlement to end a dispute that could have closed the service in the US. RIM stated that the deal with American patent-holder NTP was a "full and final settlement". NTP, which had claimed RIM stole its technology, had tried to get the Blackberry service shut down in the US[13].

In addition to patent trolls there are companies who make products, but release patented ideas and encourage them to be introduced into standards without disclosing the patents. Once the standard is adopted they then are able to force those who implement the standard to license their patent. This form of undisclosed patent is known as a "submarine patent", because they are designed to ambush companies. Once again small software companies who implement standards in good faith would potentially be targets of this kind of unethical behaviour.

2.2 Innovation patents

From OSIA's perspective, it is clear that the innovation patent regime has failed to achieve its stated aim of "stimulating innovation by Australian small to medium business enterprises".

Rather, innovation patents, for the most part, have served as a *disincentive* to innovation in SMEs, most notably in the field of computer software (see Section 2.1 on page 4) but also more generally.

ACIP's issues paper and options paper identify a variety of valid reasons for the counter-productive effect that innovation patents have had.

OSIA views two of those reasons in particular as fundamental to the failure of the innovation patent regime.

2.2.1 Grant without examination

The purpose of granting innovation patents without examination was clearly to make it easier for Australian SMEs to enter the patent system. Whilst the aim was a worthy one, grants without examination have given rise a rather insidious side effect.

Since an innovation patent can be granted without examination, all applications for innovation patents are granted (assuming they are submitted in the required form and the prescribed fee is paid).

As a result, literally anything can now be patented, regardless of the patentability of the subject matter, the inventiveness, novelty, utility or existence of prior art. For example, on 24 May 2001, Patent #2001100012 was granted on, of all things, the wheel.

The *Patents Act* seeks to guard against abuse of the grant-without-examination process by requiring (post-grant) examination of an innovation patent before the patent can be enforced.

In relation to the most outrageously absurd patents, it is easy for anyone to see that they would fail to be certified when subjected to post-grant examination and therefore could not be enforced — for example, in the case of #2001100012 several millenia of prior art are evident, even to the layman.

At the other end of the scale, at least theoretically, it may be possible for a particular unexamined innovation patent to exist such that it was easy for anyone to see that it *would* be certified upon post-grant examination — although such a case would be unlikely to arise in practice, as if the invention were so obviously patent-worthy, the inventor would no doubt seek a standard patent on it instead.

Unfortunately, the vast majority of unexamined innovations patents occupy the middle ground — neither so lacking as to be clearly uncertifiable, nor so notable as to have no chance whatsoever of failing to be certified, at least in the eyes of anyone ordinarily skilled in the art.

That is not surprising — after all, the existence of that middle ground is why we have patent examination in the first place.

The trouble is that the existence of those unexamined innovation patents that occupy the middle ground gives rise to near-total uncertainty amongst others practising in the relevant field as to whether the subject matter is or isn't subject to an enforceable patent.

[13] *Settlement ends Blackberry case* http://news.bbc.co.uk/2/hi/business/4773006.stm

For risk management purposes, other organisations considering innovating in that area — particularly SMEs, who are less able to withstand potential infringement proceedings — are likely to assume that the unexamined patent will stand, and therefore refrain from innovating in that area.

Some such patents may go on to be examined and certified, so will indeed be enforceable. But others will not or cannot — thereby giving rise to circumstances in which an unexamined innovation patent acts as a disincentive to innovation: the complete opposite of the innovation patent's intended effect.

2.2.2 Bar too low to be meaningful

A major feature of the innovation patent system is the lower patentability threshhold of innovation patents in comparison to standard patents ("innnovative step" rather than "inventive step").

Although the interpretation may not yet be completely settled, it appears from the judicial decisions reported to date that the main point of difference between the criteria for establishing an "innovative step" and those for establishing an "inventive step" is that the former do not include a test of non-obviousness.

IP Australia themselves have stated that innovation patents allow "even clearly obvious enhancements to be patented" which leaves the current system open to inappropriate use [14] and raised concern about their potential to facilitate both evergreening and the creation of "patent thickets".

OSIA questions what public good could possibly arise from granting a patent on something obvious. We see no valid reason for doing so. Rather, we note that doing so can and does lead to effects contrary to the purpose of the patent system — i.e. rather than encouraging invention, it serves instead to stifle it — by facilitating the creation of patent thickets, which serve only as a disincentive to progress the relevant field.

Innovation is a necessary and useful component of progress in many fields of endeavour. OSIA's position is simply that innovation without invention ought not consitute sufficient grounds for the grant of a patent.

[14] IP Australia, *Innovation Patents — Raising The Step Consultation Paper*, 2012, http://www.ipaustralia.gov.au/pdfs/Innovation_Patent_System_Consultation.doc, p. 4.

3 Observations on Option C

Section 5.6 of the ACIP options paper described a series of potental reforms to the innovation patent regime, as Option C. Whilst most of the mooted reforms have merit, OSIA's position is that Option C — even in its entirety — does not go far enough towards solving the problems evident in the innovation patent regime today.

At the very least, in addition to the reforms proposed in Option C, computer software should also be excluded from the scope of subject matter on which standard patents may be granted. In OSIA's view, that one measure, combined with the abolition of the innovation patent system, would be the simplest and most effective means to address the deficiencies in Australian patent law today.

3.1 Raise the level of innovation

This approach appears well intentioned, but avoids the crux of the matter. Issues around the level at which the "bar" is set for defining what is or isn't "innovation" exist only as a direct result of innovation patents being introduced with that test (as opposed to the more established test of "invention" applied to standard patents).

It is difficult to see any public policy benefit in granting a patent on anything involving less of an inventive step than that required to secure the grant of a standard patent. Since we already have standard patents for that purpose, OSIA sees no benefit — and substantial detriment (see Section 2.2 on page 8) — in retaining the innovation patent regime.

3.2 Limit the monopoly

Limiting the monopoly granted by an innovation patent to a single embodiment would indeed serve to reduce the disincentive to progress that innovation patents currently provide.

However, in relation to computer software (which is OSIA's chief interest), the exact same objective can already be achieved through copyright.

Copyright on a work of software extends only to the specific implementation in the copyrighted work. A monopoly granted by patent on application with identical scope to the monopoly already granted by copyright automatically would serve no useful purpose.

3.3 Change processes — formalities check, compulsory certification

The introduction of a formalities check is an obvious reform, which ought to have been in place since the inception of the scheme.

Compulsory certification would provide more substantive benefit, only if required pre-grant. Compulsory post-grant certification would still leave innovative organisations other than the grantee with the same unacceptable degree of uncertainty that exists today — albeit for a lesser period than at present.

Concerns around this measure potentially acting as a disinscentive to SMEs innovating are unwarranted. The risk arising from the present uncertainty is a far greater disincentive.

3.4 Change the name of the right

OSIA is not convinced that changing the name given to as yet uncertified innovation patents would have any measurable positive effect at all.

The detrimental uncertaintly that arises from unexamined innovation patents arises from the risk of their *potential* enforcability post-examination, not from their name pre-examination.

Changing the name will reduce neither the uncertainty nor the disincentive to innovation that it gives rise to.

3.5 Education

OSIA sees the question of where IP Australia publishes its information about how to file for appropriate IP rights as a red herring.

IP Australia already publishes this information on its web site. It is reasonable to assume that any Australian organisation today who may be likely to seek a patent or trade mark is aware that IP Australia is the relevant government agency and that their web site would be a logical place to find relevant information. It should also be noted that most such organisations will continue to seek independent legal advice as and where necessary.

3.6 Exclusions

In OSIA's view, this is by far the most useful of the reforms mooted in the ACIP options paper.

OSIA supports the proposed exclusion of computer software from patentable subject matter. This exclusion should be applied to all patents, not just innovation patents — see Section 2.1 on page 4.

OSIA also supports the proposed exclusion of methods and processes. It is instructive to reflect that, had those sorts of patents been available during the formative years of the computing industry and had a patent been granted on, for example, "structured programming"[15], progress in the industry would likely have stalled for two decades, as a result of which the "PC revolution" of the early 1980s (which was fuelled mainly by general-purpose business software — the so-called "killer applications" of the day, rather than the actual computers on which that software ran) would not have occurred.

OSIA does not take a position either way on the proposed exclusions for chemical compositions and pharmaceuticals as those are not fields of endeavour in which our members innovate.

3.7 Limit access to the innovation system

OSIA does not see substantial merit in the proposal to limit access to the innovation patent system to inividuals & SMEs, nor in the mooted limiting of access to Australian nationals.

The negative effects of the present innovation patent regime for Australian SMEs — aside from the negative effects of patents in general (innovation or otherwise) on computer software — arise chiefly from the uncertainty arising from uncertified innovation patents. That uncertainty will persist so long as the innovation patent system does, regardless of whether the holders of uncertificed innovation patents are individuals, SMEs, large corporates, government agencies or foreign entities.

[15] a major advance in the process of computer programming, emerging from the work of Böhm & Jacopini and later Dijkstra, in the late 1960s