Open Software & Open Standards in South Africa

A Critical Issue for Addressing the Digital Divide

National Advisory Council on Innovation[1]
Open Software Working Group

January 2002

Version 1.0

[1] The National Advisory Council on Innovation (http://www.naci.org.za/) is a body set up by South African Act of Parliament to advise the Minister of Arts Culture Science and Technology, as well as Cabinet as a whole, on science and technology issues.

Executive Summary

South African Context

The dramatic weakening of the Rand in recent weeks highlights the importance of producing South African goods and services to substitute increasingly expensive imports and to generate exports at relatively low cost. While this applies to practically everything, the goal of this document is to focus on software use and development.

A sustained Rand slide will make licenses on imported software (not to mention other imports) prohibitively expensive. Whether or not the Rand enjoys an upswing in future, it makes sense to minimise risk through avoidance, where possible, of dollar-based software license fees and through vigorous encouragement and support of local software development efforts.

Needless to say, lowering software costs by violating proprietary license conditions is not an option. Happily, there are often legal alternatives to proprietary software: non-proprietary "open software"[2] attracting no license fees at all. Furthermore, open software may be freely probed, customised and modified. This is the cheapest way of generating software suited to the country's needs. It is also an ideal jumpstart for entering the software development arena.

South Africa and other developing countries are extremely well placed to compete in the global software development market. Creating software is best done with a relatively inexpensive but well trained labour force. Software development is, and will continue to be, a knowledge and people intensive activity.

Like governments in many countries (developed and developing), it is time for South Africa to promote open software and open standards. This accords with the priority given to the theme at the first meeting (October 2001) of the Presidential International Advisory Council on Information Society and Development.

However, government action cannot be the sole objective. Various people and institutions in South Africa, including small and large companies, are already using open software products (notably Linux and associated software tools) precisely because they already have the freedom to do so rather than because they have been prompted by government policy. The bare minimum is to ensure that this freedom is not curtailed by introduction of inappropriate policy.

That said, government has a key role to play in accelerating universal access and providing leadership for the African continent as a whole. Furthermore, government is the largest procurer of Information and Communication Technology (ICT) in the country, accounting for some 70% of total spend. Given this level of clout, government action is bound to stimulate industry in various ways, such as the provision of open software training and support.

[2] "Open Software" is a shorthand for "Open Source Software", whose formal definition is given in the text.

Open Software and Open Standards

Open software is both an opportunity and an important resource. South Africa now has the opportunity to participate in, and benefit from, the open software movement. South African companies and developers are already a driving force in many open software projects. If open software is able to change the rules in the information technology industry, the country and the companies that better understand it and are more advanced in its use and knowledge will have a clear competitive advantage. This document proposes recommendations designed to help the country to benefit as much as possible from open software, and to remove the barriers to future open software development projects.

Many governments are now developing national policies to promote the use of open software – examples include China, Thailand, Brazil, Argentina, Germany, France and the United Kingdom.

A related issue is the adoption of open standards, which make it possible for open and proprietary software conforming to these standards to inter-operate and exchange data. This is essential for seamless inter-government and government-to-citizen communication. The Internet owes its explosive growth and impact to its foundation on open standards and open software.

Open software has reached a critical mass that has allowed it to enter the mainstream software market and its impact is becoming noticeable in the software industry and in society as a whole. Companies like IBM, SAP, Sun, Intel, Hewlett-Packard and Silicon Graphics are committed to using open software as a core part of their business and are investing significantly in enhancing its already impressive capabilities.

Open software is an especially useful tool to allow developing countries to leapfrog into the information age. It encourages novel development models that have been demonstrated to be particularly well suited to take advantage of the work of developers collaborating across the Internet. In general, it also has a positive impact as an enabler for the creation of new markets and business opportunities.

In summary, the major benefits of open software and open standards include:

- Reduced costs and less dependency on imported technology and skills
- Affordable software for individuals, enterprise and government
- Universal access through mass software rollout without costly licensing implications
- Access to government data without barrier of proprietary software and data formats
- Ability to customise software to local languages and cultures
- Lowered barriers to entry for software businesses
- Participation in global network of software development

Recommendations in Brief

A. ICT in the Public Sector

1. Make Open Standards a non-negotiable base for ICT in the Public Sector.
2. Encourage government agencies and public institutions to use Open Software whenever feasible.
3. Allow Open Software to compete on a “level playing field” with proprietary alternatives in government software procurement.

B. Open Software Development

4. Promote documentation, translation and localisation of software, especially for use in the Public Sector.
5. Promote Open Software in pre-commercial research and development projects financed with public funds.
6. Establish an Open Software Development Initiative

C. Training and Support

7. Establish a national capability for testing, evaluation, verification and accreditation of Open Software.
8. Promote education and training on Open Software products
9. Provide incentives for Open Software training and development

D. Legal Issues

10. Oppose patenting of standards, software and algorithms.

South African Context: Scenarios

The following scenarios give a flavour of the challenges to which Information and Communication Technology in South Africa needs to rise.

Sello's woes – a small enterprise scenario

Sello runs a spaza shop in a vibrant part of the township, which turns out to be a mini-gold mine - but he is taking money so fast, he is losing track. His brother has an old PC that he bought from a pawnshop, with no software, and Sello would like to use it as a till. He doesn't care if it cannot do anything fancy, he would be happy to use it to keep track of the takings over a day. A rudimentary spreadsheet will do the trick.

He runs into a big problem: not only is the latest software expensive with all manner of bewildering bells and whistles that he does not need, but it doesn't run on such an old machine. What is he to do? Would it be legal to find and use an old copy of the operating system and spreadsheet? "Certainly not", replies his software dealer, and sternly warns him of the fate that awaits users of illegal software copies.

With free software, it would be perfectly legal and Sello would be in business. With proprietary software, he would be breaking the law.

He harbours vivid memories from years past of the dreaded early morning raid – only this time it would be the software copyright police.

Funeka's awakening - a disadvantaged school scenario

Funeka is a schoolteacher with a mission: to give her dusty, rural school the very best. She launches a campaign to build a computer lab and approaches various businesses for help. To her delight, one company donates 20 computers that are being replaced, but the company will keep all their software licenses for their new machines. She also has to find her own educational software.

Delight turns to horror when she discovers that it will cost many thousands of Rand for software licenses, including licensing the educational software the dealer tells her she needs. To make matters worse, casual inspection reveals that the

content is geared to American schools, using unfamiliar baseball metaphors and the like.

Meantime, Funeka's students have been doing some legwork of their own. They have contacted a young IT company that has offered to network the computers and connect them to the Internet. When the company's network guru calls by and finds computers with no software, she installs Linux and associated free software on all of them, sets up the network and Internet connection and even gives the students a preliminary driving lesson on using the software and surfing the Internet.

While Funeka agonises over raising a software budget, the students spend many days probing, exploring and discovering new things. Within a short time they have learned to do creative projects by searching the Internet and sending email around the world for facts they can't find in the tiny school library. Using tools and examples from other Web sites, they soon start designing their own school Web site and developing content like a Web-based newspaper covering school and local community issues.

When she learns of all this, Funeka is amazed at the creativity of her students, and decides that her original idea of what computers should do is completely wrong. She had thought of the computer as just another passive medium of instruction.

Funeka quickly adapts to this awakening, and promptly arranges a session on the Internet - given by her students to members of staff. They are all amazed that all this has happened without the school having to pay a cent in software licenses.

They also heartily approve when the students explain their plans to design a community resource for guided access to government Web sites. The one concern the students have is that they are often unable to read files downloaded from government sites. The problematic files are in a format that requires proprietary software to read.

Sipho's choice – a research scenario

Sipho has good reason to be pleased with himself; he has just submitted a groundbreaking PhD thesis at a leading South African university. Using advanced concepts in mathematics and physics, his thesis, "QVM: the Quantum Virtual Machine", proposes an ingenious algorithm to speed up the conventional PC beyond the wildest dreams of classical wisdom.

QVM will make light of computer resource hungry fields like environmental and climate modelling, determination of protein structure and function, discovery of new drugs, complex industrial simulation and design *etc*. It will also lead to a host of completely new applications that inevitably accompany such a major computational advance.

Sipho cannot wait to publish a paper in a high impact international journal giving full details of QVM principles and design. He also intends to place a full software implementation on the Internet, allowing anyone to download and use it on a standard PC. No license fee, no royalties. They can use the software as they please –learn from it, modify it - as long as they do not repackage and sell it for private commercial gain *and* attempt to stop others from using the free distribution.

His friends are horrified - he could license QVM to a global computer company and make a fortune. The university is horrified - it could license QVM to a global computer company and make a fortune. His supervisor is horrified…

But Sipho stands his ground. He firmly believes in the freedom (or should that be obligation?) to publish academic work supported using public funds – software included. His own research benefited immensely from the use of software distributed under similar conditions.

He is also mindful of a moral obligation to seek the greatest economic gain for the country from publicly funded research. But this only strengthens his resolve. He is convinced that greater benefit can accrue to South Africa's scientific and economic fortunes through his suggested route than by surrendering such a major scientific breakthrough wholesale to any single company, whether it is foreign (almost certainly) or local.

"Is he very foolish or simply ahead of the game, like he is in his research?" his friends puzzle. "Is he really acting in the country's best interest or is he a well-

meaning but naïve academic?" wonders the inquiring public. "Should a man like this even be allowed a choice on the matter?" fumes the university's deputy vice chancellor for research.

Language translation – a true open software story

South Africa has eleven official languages. This is a logistical nightmare when it comes to communicating decisions and policy to the nation, but think also of the learner whose English is not very good but who is fluent in Xhosa. Most of our population is excluded from computer technology simply because they do not have the required language skills. A solution is to improve English literacy. Although this is an admirable goal in itself, providing African language software is both possible and simple.

If it is so easy then why are there no African language software tutorials, online manuals and associated software? The main reasons are that there is little commercial interest in multilingual products and is not competitive. Furthermore, the packaged software market largely uses inflexible proprietary standards that do not make it easy for users to add their own enhancements. By contrast, a local NGO called Translate (http://www.translate.org.za/) has already released a Xhosa version of some software after only 3 months of work.

They were successful mainly because they translated KDE, an open software desktop akin to Microsoft Windows in functionality. KDE itself is sensitive to language issues and is currently translated into 42 languages, far in excess of any of the popular commercial packages. It took Translate six weeks of work to translate enough of KDE into Xhosa to make it ready for release. Another six weeks were spent for other minor components and documentation. It was so easy to include Xhosa in KDE because there is a spirit of co-operation and collaboration in open software projects. As a result it enjoys some of the richest translation tools and is multi-lingual from the ground up. Information is freely discussed and shared, which means that KDE, like most other open software, is rapidly being enhanced by thousands of volunteer programmers around the world.

Commentary

Sello's story illustrates the folly of seeking the best and latest. It is often overkill for many people's purposes. Producers of proprietary software simply have no interest in maintaining older versions. This severely restricts the freedom to mix and match to suit one's needs, particularly on older hardware. This cannot be in the best interest of a developing country.

Funeka' story demonstrates the need to think clearly about the problem one is intending to address. In the school context, self-initiated exploration, formal instruction and school administration are 3 distinct areas. Think carefully before making a huge investment in expensive software. It also touches on the importance of promoting access to government information through the use of non-proprietary data formats.

It is one thing to make use of currently available open software, but when should software developed with public funds be open source? Whose judgement call should it be? Sipho's story raises this question and a number of related issues. These issues are discussed in some detail in a recent article "Public money, private code"[3].

The benefits of the language translation story are self-evident.

What follows is a discussion of open software open standards against the background of the scenarios above.

[3] http://www.salon.com/tech/feature/2002/01/04/university_open_source/index.html

Introduction

Information and Communication Technology (ICT) is making rapid inroads into virtually all aspects of life, affecting the way we work, live and play. Computer chips can be found in computers, industrial machinery, cell-phones, cars, household appliances and so forth. Software contains the instructions that the chip processes in order to give these products an appropriate behaviour. Software also enables a device to exchange information with other devices connected to it via the Internet or other network. Software is thus key to the generic function of ICT - sharing and processing information.

Unlike the physical device that it may lend functionality to, software is a coded embodiment of ideas and knowledge. Like all knowledge (and unlike physical artefacts), to share it is not to lose it, whether or not a fee is involved. Indeed, to share knowledge is to enhance it because it enables others to build further upon it. The principle of open disclosure and rigorous peer scrutiny has underpinned the advancement of human knowledge for centuries.

This is a principle that today underpins the development of **open software** – software that is free of proprietary restrictions (see formal definition below). An intimately related issue in ICT is that of **open standards** for communication. Open software implementations of specified standards are available to anyone without incurring prohibitive licence fees or other proprietary restrictions. The Internet, and its associated applications such as the World Wide Web and e-mail, is the most visible triumph of such openness in ICT.

The other side of the openness coin is that an individual or organisation may not wish to freely disclose software that is commercially strategic or sensitive in some way. This choice has no less merit than the choice to be open. One cannot decree that all software must be open any more than one might insist that all knowledge must be shared freely. Thus, one should have no qualms about the co-existence of open software alongside closed or proprietary software. However, it is necessary to take issue with any attempt to restrict non-proprietary software use or development (through sweeping software patents, say) just as vigorously as it is necessary to defend freedom of expression and the open exchange of ideas.

Formal Definition and Brief History

The term “open software” is used as a shorthand for “open source software” (OSS), hence the terms will be used interchangeably[4].

OSS is typically developed through public collaboration, it is available to anyone (usually at little or no cost), it does not require proprietary license fees and it may be freely re-distributed. Users also have access to the human readable version of the software called the “source code”, revealing the inner workings of the software and

[4] “Open software” is preferable because it does not make constant reference to the rather technical concept of “source” and it has a symmetry with “open standards”

allowing its modification, hence the term "open source"[5]. The use, modification and re-distribution of the source code is governed by rules specified in associated non-proprietary open source licenses.

Access to source code has the potential to empower people in ways that proprietary software simply does not allow. It offers people the freedom to probe, modify, learn from and customise the software to suit their needs. Hence open source software is also known as free software, where "free" refers, first and foremost, to this freedom rather than to "free" in the monetary sense[6].

Free software has existed since the invention of the first computers. However, its production, distribution and use was limited to a few engineers, scientists and others who had access to the then expensive computing facilities.

More recently, the initial development of the Internet was propelled by the use of open software and the adoption of related open standards. For example, the most basic Internet applications such as e-mail, FTP, Gopher and the World Wide Web all make use of open software for their successful deployment and overall adoption. More than half of all web servers use open software[7].

As increasingly rich standards are being developed to cater for a variety of Web services, it becomes all the more important that associated protocols and tools remain freely accessible if the Web is to retain its non-proprietary pedigree.

Adoption and Impact

There is a groundswell the world over of adoption of open software by individuals, small and large enterprises, schools and other public and private institutions. This reflects a growing acceptance of and confidence in open software and open standards, and thus contributes to their further development. It is no idle matter that IT industry big names, notably IBM[8], have expressed such confidence and commitment.

Where the open software message has been promoted in South Africa, it has mostly been by individuals in academia, NGOs and small software companies. It is timely for such bottom-up initiatives to be complemented by initiatives led by government policy and action.

Indeed, many governments are now developing national policies to promote the use of open software. Examples include China, Thailand, Brazil, Argentina, Germany, France and the United Kingdom. Information technology professionals are also encouraging their governments to adopt policies that support open software, New Zealand being a

[5] See http://www.opensource.org for a full definition.
[6] See http://www.gnu.org/ for fuller discussion.
[7] Netcraft's survey http://www.netcraft.com/survey shows that some 60% of web sites use the open source Apache Web server
[8] "We believe very, very strongly that open standards and open-source software are absolutely critical foundations for the IT business going into the future." IBM, http://www.ibm.com/news/us/2001/08/15.html

notable case in point[9]. Some of these countries recommend that software used by government and its associated agencies must be open software, unless proprietary software is the only available option.

The initiatives in Germany, France and the UK are a result of, or are closely allied to, the European Commission initiative "eEurope – An Information Society for all". The initiative's action plan set the target: "During 2001 the European Commission and Member States will promote the use of open source software in the public sector and e-government best practice through exchange of experiences across the Union".

In 2000, a working group of the Commission published a document on free (libre) software covering an analysis of the phenomenon and making recommendations on "how to help Europe to benefit from open source software" [10].

In December 2001, as a follow up to the Commission's initiative, the UK published its document on proposed use of open source software within UK Government[11], along with a comprehensive analysis of the impact of open source software[12]. This complements an earlier document on open standards and specifications for e-Government[13].

The UK documents represent the most comprehensive set of documents by a government on an analysis of open software and open standards, with associated recommendations and a proposed plan of action. Most of what needs to be said about open software and open standards – history, development models, licensing models, economic impact, future prospects - can be found in these documents and references therein as well as related documents such as the report by the European Commission Working Group.

Strengths and Weaknesses

As the foregoing discussion suggests, many people now believe that the future impact of open source software in the ICT industry and in society in general will be so huge that the current rules by which the software industry behaves will completely change. But what precisely are the strengths and weaknesses of the open software model?

As pointed out in [12], software falls into two broad categories:

- Software infrastructure - the plumbing of ICT systems and the Internet - includes operating systems, databases, Web servers and other components that enable software applications to run. This category is taken to include Web and system middleware - the increasingly important intermediate layer between low level infrastructure (operating systems) and user level applications.

[9] See http://www.openz.org/
[10] "Free Software/Open Source: Information Society Opportunities for Europe?", http://eu.conecta.it/
[11] "Open Source Software: Use Within UK Government", http://www.govtalk.gov.uk/
[12] "Analysis of the Impact of Open Source Software", http://www.govtalk.gov.uk/library
[13] "e-Government Interoperability Framework", http://www.govtalk.gov.uk/library

- Software applications, including generic desktop business applications such as word processors, spreadsheets, financial and management systems, as well as various other (often highly specialised) applications that an organisation may need to run its business.

There has been significant penetration of open software in the infrastructure category. Reference [12] comments that "OSS is indeed the start of a fundamental change in the software infrastructure marketplace, and is not a hype bubble that will burst". The authors proceed to predict that "within five years, 50% of the volume of the software infrastructure market could be taken by OSS".

Open software has not yet had a significant impact in the generic desktop applications category, where Microsoft Office, running on the Microsoft Windows operating system, is the *de facto* standard.

Nevertheless, there are already many open source office suites, such as KOffice, Gnome Office and OpenOffice/StarOffice, which already offer equivalent functionality and ease of use. Appendix A in [12] gives a detailed discussion of OSS alternatives to MS Office. Hence the significant obstacle to adoption of OSS desktops is neither technical nor is it any longer a matter of ease of use - it largely has to do with familiarity and to some extent compatibility.

It is not possible to ensure full and sustained compatibility with proprietary MS Office document formats that evolve from time to time. The matter of compatibility needs to be embedded in the wider context of interoperability of different systems across a network. Proprietary document formats impede public access to online government documents.

This strongly suggests a commitment to open standards for interoperability in government use, together with a commitment to the use of non-proprietary formats for document exchange. Indeed, this is a fundamental recommendation of this document.

Familiarity breeds resistance, at least for the user who is not too disaffected by the current offering. Newcomers to computers – students and others – will be more receptive to a different, open model. Furthermore, because the source code is available, the OSS desktop can be freely customised to suit local needs. An obvious candidate is support for local languages. This alone makes a compelling case for the OSS desktop in South Africa. Accordingly, it would make sense to pilot the OSS desktop in various public sector institutions as a possible precursor to more widespread use.

The language support problem does not arise in a country that is entirely English speaking (or other "mainstream" language officially supported by MS Office or other packaged commercial software application). Furthermore, in a developed country, familiarity with MS Office is more entrenched than in a developing country. Against this background, [12] recommends against the OSS desktop in the UK, subject to reassessment by the end of 2002, but adds: "However, OSS on the desktop may soon become a significant player on the desktop in the developing world". That said, the OSS

desktop is being adopted or piloted by some institutions in the developed world. For example, the US Defense Information Agency recently committed to StarOffice on the desktop[14].

In the light of the preceding discussion, another fundamental recommendation is that government ICT procurement policy should allow open software solutions to compete "on a level playing field" with proprietary software - at both the infrastructure and the application level. Clearly, if there are no established open software alternatives (such as certain business process related solutions) the issue of choice does not arise.

Software Development

As suggested by the UK Report, it is reasonable for government to seek full rights to bespoke software rather than allow the contractor to claim full rights. Where appropriate, government can then choose to release the software under an open source license and hence open the contractor's work to general scrutiny. Maintenance and upgrade might then be conducted by different contractors and hence minimise supplier lock-in.

Groundbreaking software often originates in academic research. In the section "Improving the competitiveness of UK industry", [12] notes that "Open source has been the *de-facto* standard for the exploitation of academic software in the US for many years. It is hard to over-state the beneficial effect that this has had on the technology and the wider computer industry." It recommends that open source software should be the "default exploitation route for Government R&D software" in the UK.

This is a well-founded recommendation, one that South Africa also ought to adopt for government funded software R&D conducted at universities and research councils. Such a move would be in the interest of South Africa's industrial competitiveness.

It may make sense for government agencies to develop software as open software from the outset, and take best advantage of the open software development model of voluntary collaboration.

Development Model

Open software has an unusual but very powerful development model. The bulk of the development effort has traditionally been provided by a large group of volunteers from all corners of the globe connected by the Internet. This is possible exactly because access to the sources of the software is unrestricted and modern computer networks allow for very efficient distribution and scrutiny of the latest versions of the software by people all over the world. This has proven to be a way of producing very robust software that is well known for its reliability. Eric Raymond refers to such a distributed volunteer model as a "Bazaar" as opposed to the "Cathedral" in an orthodox hierarchical development model[15].

[14] See "http://www.zdnet.com/zdnn/stories/news/0,4586,2781914,00.html"
[15] "The Cathedral and the Bazaar", Eric Raymond, http://www.tuxedo.org/~esr/writings/cathedral-bazaar/

This volunteer model has arisen in the developed world, especially the USA, where there is an implicit subsidy provided by wealthy institutions, well-funded universities and corporations. The question is how this model translates to poorer countries and how it should be modified if need be without killing the goose that lays the golden egg.

South Africa has a tradition of public support for enterprises in the public and national interest. This tradition should be extended to the development of open software that addresses national needs. This suggests a hybrid development model involving distributed volunteers anywhere in the world and explicit financial support (perhaps by making use of the Universal Service Fund) for others in the country with the enthusiasm and aptitude to be involved in chosen development projects. Well-conceived projects can be an excellent mechanism for capacity building in ICT and assimilating the discipline of collaborative software development.

A complement to development is evaluation and certification of open software. The hybrid development model might include a co-ordinating agency with longer-term employees to carry out this function on behalf of bodies such as the State Information Technology Agency (SITA). It is expected that the training and support role would be fulfilled by a groundswell of companies that a government commitment to open software would undoubtedly give rise to.

It would also make sense for Government to co-operate with other governments and international agencies to build software components and systems that are of common interest. An example is the UK government's involvement in the development of XML schemas (open web-based information exchange standards) as part of an e-government interoperability framework (e-GIF), as described in [13]. Governments could also share experiences on various projects, such as OSS desktop deployment pilots.

Industry and Local Skills

Although less competitive with countries like India, software development in South Africa is extremely cheap compared with Europe and the US. The trend towards supporting open software in Europe creates useful opportunities, given our favourable time zone for Europe. Open software, by its nature, tends to be developed by a large number of relatively small cost-sensitive companies, so the possibility of South African developers teaming up with such companies is real. The local skills base therefore already exists to some extent, which could quickly be mobilised to achieve growth. Developing a local skills base is the strongest possible response to the digital divide.

Proprietary software does not create the same opportunities. A large fraction of the market is sewn up by a small number of companies based in North America. Most of them already have their development operations set in a specific pattern and are not looking for new help, but simply to resell support via a small number of approved distributors.

Proprietary software also has expensive long-term maintenance and support implications. Producers of proprietary software have no interest in maintaining older versions. In a large organisation, such as government, computers of many different ages are in operation. When a software producer decides to discontinue a particular version, users find themselves forced to replace their oldest equipment or software, whether the equipment is still in good condition or not. With open software, the potential exists to continue to support old versions.

Because open software by definition includes access to the source code, old versions can be maintained and enhanced. If necessary, government can use local contractors to modify the open software to fix problems with old versions, so they remain usable. For example, an old version of a database could have a serious flaw that is corrected in a later version. If the later version requires a more up to date computer to run, a user with an old computer may not be able to correct the flaw. On the other hand, if an open word processor were used, the flaw could be corrected in the old version.

An important outcome of developing a local skills-base in maintenance and support is that these skills are exportable, without the skills leaving the country. Many support and maintenance problems can be handled by phone or through the Internet, and our position is highly advantageous relative to Europe, in terms of time zone, and a population well versed in English.

Intellectual Property Issues

Intellectual property has to be respected by any nation that seeks to develop an Information Technology industry. The purpose of Intellectual Property protection has always been to encourage innovation and this is certainly the intention of this document. Intellectual property protection is a service provided by the state in order to encourage innovation.

The free publication of software source is analogous to the way the scientific community publishes its research results in order to encourage rapid scientific progress. Thus in the software arena, open software is an effective innovation vehicle.

Patents were introduced in the nineteenth century precisely to encourage inventors to make their work public while guaranteeing them rewards for a limited period. When applied to software, something that is not (yet) possible in South Africa, the effect is precisely the opposite. Large corporations in the developed world use software patents to prevent access to the market by new entrants. This is mainly because software patents correspond to the patenting of ideas rather than inventions in the traditional sense: new software ideas and algorithms are general and universal, in correspondence to the universal nature of computers. Additionally, patents are granted for extremely long periods relative to the IT innovation cycles, which are typically of the order of 18 months. This contrasts with the pharmaceutical industry where the innovation-exploitation cycles are much longer. Lastly, software patents have frequently been granted for extremely simply notions leading to a plethora of notorious junk patents.

These days it is very difficult to write any major piece of software that does not infringe on a number of silly patents that have so far been granted in the US. The effect is that large company A, who owns a large stock of patents, can come to an amicable agreement with large company B who has a similarly large stock. However a startup from developing country C has no such stock of patents and any software it writes is dead before it emerges.

Copyright is an appropriate mechanism for protecting software. It relates to the actual code written. Patents should not be granted on software and algorithms.

Return on Investment

Finally, it is worth commenting on the issue of total cost of ownership (TCO). TCO tries to capture all the real costs of the software's life cycle including initial acquisition, upgrade, training, maintenance, and decommissioning. It is often argued that software licenses (typically, the cost of initial acquisition and upgrades) can be a small part of the TCO. Hence by attacking proprietary licenses, open software advocates are failing to see the big picture.

But acquisition and upgrade is precisely the dollar-based cost component for imported software that South Africa would do well to minimise. As discussed above, training and maintenance ought to be fulfilled by a local skills base.

The issue has to go beyond TCO to return on investment (ROI). The desirable ROI in software has to be an enhanced national skills base for software development, training and maintenance while reducing the country's dependence on imported software. Open software is the logical way forward.

Conclusion

The trade-off between the proprietary and open approaches amounts to choosing between relying on foreign skills and developing local skills. The proprietary approach requires higher up-front costs and, in many cases, higher long-term costs. The open approach requires a much lower up-front cost (anywhere from zero, to the cost of buying a CD, to avoid a slow Internet download), but requires a bigger investment in a local skills base to enable local software development.

In terms of a national strategy, the choice is clear.

If South Africa chooses the proprietary route, the cost in many cases will be higher, and much of the expenditure goes out of the country. The country becomes dependent on foreign companies for much of our technological requirements, and hostage to currency fluctuations.

If South Africa chooses the open route, the cost will often be lower, and much of the cost will remain in the country. Further, South Africa can break dependence on foreign companies, and potentially become a player in the world software development and

software services markets.

Recommendations

Most of the following recommendations call for government intervention. Where appropriate however, initiatives may involve government agencies, public institutions, non-governmental organisations, the private sector or public-private partnerships (PPP).

A. ICT in the Public Sector

1. Make Open Standards a non-negotiable base for ICT in the Public Sector.

ICT standards need to be open (available without restriction to any developer or user) to ensure inter-operability (seamless sharing of data and information) between applications and between users. The Internet and World Wide Web are founded on Open Standards. A closely related issue is the adoption of neutral (non-proprietary) data formats for document exchange in the Public Sector.

Benefits:

a) Promote inter-operability within government agencies as well as between government and the public.

b) Promote universal access to online government services without prohibitive costs, license restrictions or similar barriers.

c) Minimise the risk of lock-in to specific vendors of ICT products and services.

d) Lower barriers to entry for local developers seeking to offer ICT solutions for use in the Public Sector.

2. Encourage government agencies and public institutions to use Open Software whenever feasible.

Open Software is available to anyone (usually at little or no cost), it does not require proprietary license fees and it may be freely re-distributed. There is an intimate link between Open Software and Open Standards. Set up public sector pilot programmes on the use of Open Software on the desktop.

Benefits:

a) All the benefits of the Open Software model, such as direct access to software without proprietary license obligations.

b) Cost effective transfer of software technology across national borders.

c) Stimulate an indigenous software industry based on Open Software.

3. Allow Open Software to compete on a "level playing field" with proprietary alternatives in government software procurement.

Tenders and Requests for Proposals from government agencies such as the State Information Technology Agency (SITA) should include provisions explicitly allowing the desired objective to be carried out using Open Software. This must be supported by a capability to evaluate Open Software offerings (see below).

Benefits: As above

B. Open Software Development

4. Promote documentation, translation and localisation of software, especially for use in the Public Sector.

Availability of key ICT applications and services in South Africa's official languages is absolutely fundamental to the notion of universal access. Proprietary solutions restrict the freedom to conduct the necessary localisation. An Open Software base may be the only way forward. The same may hold for other localisation needs.

Benefits:

a) Promote universal access.

b) Support for industry, particularly small and medium enterprises.

5. Promote Open Software in pre-commercial research and development projects financed with public funds.

This is particularly important for software developed to serve a national interest. The outcome would be available without restriction to a broader community, for further development and use in both non-commercial and commercial products and services.

Government should seek full rights to bespoke software and consider releasing it under an Open Software license where appropriate.

Benefits: As above, as well as

a) Education and training

6. Establish an Open Software Development Initiative

This would likely need to be a hybrid of a central funded agency and a "bazaar" of distributed developers in South Africa and beyond, focusing on software development that addresses African needs.

Benefits: Potentially, virtually everything listed thus far, as well as

a) Encourage growth of critical mass of human resources related to Open Software development. This can stimulate commercial enterprise and benefits to society.

C. Training and Support

7. Establish a national capability for testing, evaluation, verification and accreditation of Open Software.

This is an important complement to development efforts. The responsible agency could also be a repository of Open Software and Open Standards and provide guidance and advice on available solutions. It would need to be actively involved in global standards setting bodies.

Benefits:

a) Open Software evaluation service for government software procurement.

b) Support to the broader community.

8. Promote education and training on Open Software products

One aspect of this is a general education that lays emphasis on principles rather than specific software products. Hidden details of implementation and other proprietary restrictions can be a hindrance to understanding. Hence Open Software should be given preference over proprietary offerings.

The other aspect is the shortage of trained people to use and support Open Software solutions. It requires, amongst other initiatives, formally accreditation training in key Open Software products (such as Linux certification). Some training might be provided by specified agencies, but if the recommendations above are implemented they would stimulate a groundswell of reputable training and certification centres.

Benefits:

a) Build capacity and stimulate SMEs founded on Open Software development.

9. Provide incentives for Open Software training and development

Possible incentives might include credits to companies and their employees for enrolment in Open Software training programmes and development projects.

There may be access to funds from the IT training levy, the Universal Service Fund which is tasked with improving ICT access *etc*.

Benefits: As above.

D. Legal Issues

10. Oppose patenting of standards, software and algorithms.

Open Software makes use of copyright law and distribution licenses. However, broadly defined patents on software threaten software development and Open Software in particular. The developing world is particularly vulnerable in this regard.