# OPEN SOURCE AND COMMERCIAL SOFTWARE

AN IN-DEPTH ANALYSIS OF THE ISSUES

# Contents

# Part I : Understanding the Fundamentals



The growing popularity of open source has altered the software industry landscape in a dramatic way in recent years. Few technical subject matters today are as passionately debated as that surrounding open source and commercial software – two prominent models of software licensing. Policy concerns surrounding commercial and open source software have also confounded Governments around the world. In particular, open source is often seen as a possible solution to some of the challenges presently faced in various countries, especially among developing nations. Such challenges include grappling with the piracy problem, desiring greater control over software that is acquired and dealing with broader policy perspectives on how best to develop a thriving domestic software industry. With a view to assist decision-makers discern the issues in this debate, let us examine the key considerations to be taken into account in making software policy decisions.

In this first part of our four-part discussion, we will examine the nature of open source and commercial software development methodologies, the related licensing approaches and the underlying intellectual property foundation. In the second part, we will further review the characteristics of the two software models that are commonly debated to better understand the true nature of these models. In the third part, we will consider the issues that are of particular concern to Asian Governments and economies. In the fourth part, we will consider some approaches and strategies on software procurement and technology standards that Governments are contemplating to address some of the challenges presently faced by consumers and the industry.

## Definitions

In this discussion, let us define the terms as follows:

- "Open Source" is a software-licensing model where the source code of the software is typically made available royalty-free to the users of the software, under terms allowing redistribution, modification and addition, though often with certain restrictions. The support, training, updates and other services for the software may be provided by a range of entities, increasingly under commercial arrangements. Open source programs are often, though not exclusively, developed through a collaborative effort in which a number of persons contribute elements of the final software. Software companies are also contributing paid programmer time and programs developed in-house to the open source community.

- "Commercial Software" is the model where the software developed by a commercial entity is typically licensed for a fee to a customer (either directly or through channels) in object, binary or executable code. The commercial entity often provides support, training, updates and other similar services needed by customers to efficiently use that software. The source code of the software may be made available[1] to certain users of the software through special licensing or other agreements, but is usually

not distributed to the general public, and may not be copied or modified except in a manner provided for in such agreements.

Each of these software models can translate to a viable business strategy for software companies, as well as offering customers real advantages. The models are not mutually exclusive, and companies are increasingly finding ways to embrace both approaches and allow them to co-exist. For example, there have been proprietary operating system platforms that have benefited from the open source development by adopting an open source approach for the lower levels of the system (e.g. device drivers) while keeping the higher levels proprietary (e.g. user interface)[2]. This approach allows greater focus to be placed at the development of the higher-level components, where innovation may bring greater benefits to customers. Conversely, there are software providers who have contributed commercially developed code to the open source community to allow open source solutions to operate on a broader range of platforms[3]. Increased competition and a larger number of viable alternatives of products on the server and desktop platforms have contributed significantly to the IT ecosystem. Software solutions continually innovate, with software providers focusing and improving substantially on emerging issues such as addressing security and reliability concerns.

## Understanding Open Source and Commercial Software

The open source and commercial software approaches have each their own strengths and challenges, and can bring to users a number of benefits along with tradeoffs, depending on the circumstances in which they are deployed.

**The models are not mutually exclusive, and companies are increasingly finding ways to embrace both approaches and allow them to co-exist.**

Products in the form of commercial-off-the-shelf software have been in the market for many years, offering consumers a wide range of computing functionalities and productivity enhancements on a mass scale. Customized commercial software solutions have also met the complex business operating requirements of larger organizations that generic off-the-shelf products may not be able to adequately meet. Driven by the needs of the customers, the vendors of commercial software solutions continually strive towards developing products that are easy to use, rich in functionality, value for money and supported by a services eco-system as demanded by customers who may not be technically savvy, and want to solve their problems with minimum fuss.

Open source licensing has been in existence for decades, primarily in academic and research organizations in the past. More recently, it has

[1] Some examples of commercial software companies disclosing source code include Real Networks with their Helix project, Computer Associates with their Ingres release, SAP with their release of OpenDB and Microsoft with their Shared Source Licensing Program. Such source code releases may be based on terms that allow modification and re-distribution of source code, e.g. Windows CE.
[2] The Apple MacOSX product is one such example. See http://www.apple.com/opensource and http://www.wired.com/news/technology/0,1282,18488,00.html.
[3] IBM has taken this approach to enable Linux to operate on its entire range of hardware platforms. See http://www.intelligententerprise.com/010810/412e_business1_1.jhtml?_requestid=142394. a

attracted greater attention due to the corporate backing or commercial support of open source software in the market.

Consumers today are choosing from a wide range of software choices and vendors, even in areas that traditionally have few competing products. Some select open source because it allows them to freely copy, modify and then re-distribute the source code. Such characteristics appeal to those who want to alter the software source code, for example, in academic settings where experimentation is a primary objective or in settings where a high degree of customization may be required.

Active participation in a development community can enhance partnerships and mindshare among the participants, thus building valuable relationships with a broad spectrum of developers that cut across organizational boundaries. Open source and commercial software developers both try to create these communities through formal and informal sharing. The direct participation of users and developers in the use of open standards – which are distinct from open source software – can enhance interoperability rapidly and are thus used by both groups of developers.

There are today more options available to software users. It is very common to find user environments deploying a mixture of open source and commercial software platforms and applications to meet different demands. The competition between different software providers has made the overall software industry even more responsive in meeting the needs of consumers, and ultimately benefiting consumers with greater choices and better pricing options.

**It is very common to find user environments deploying a mixture of open source and commercial software platforms and applications to meet different demands.**

To better understand open source and commercial software, we will examine the two models from three perspectives – the business, development and licensing.

## Business

Businesses exist and can be sustained because they generate profit through their activities. Shareholders primarily measure business performance by the profit levels. While there are some basic differences between the business models of open source and commercial software providers, providers of both models must each find their means to create sustainable revenue. Creating software for software's sake is not sustainable economically.

The focus of commercial software providers is on the functionality, features and innovativeness of their technology to meet the customer's needs, as their revenue model is based on the customer licensing their software. Customers purchase new versions of software when it provides new functionality, features and value. This incentive drives a tremendous flow of research and development spending into new software, the results of which include higher productivity, lower costs of business, and new tools for learning.

Open source vendors create revenue from supporting services and hardware that they package around open source software and for which they charge fees. For instance, some companies package open source software, sell it with their personal computer or server hardware. The companies sell such hardware and charge the customer separate fees for the service they provide to enable their software and hardware to work together. Another example is a system integrator who earns revenue by creating customized solutions for customers by using existing open source software as the starting point, and charging the customers for the time and resources to do the necessary customizations to meet the specific user requirements. Another model is to allow free downloads of an open source application and to convert this base of users into paying customers for a full-featured version. In other instances, the pooling of development resources in support of an emerging technology can also provide indirect revenue or benefits to a company that provides open source software, such as the sale of the commercial software and/or hardware they offer beside the open source software[4].

It is recognized that in the open source community, there is also a group of contributors who are motivated not by direct revenue generation, but by an altruistic notion that all software should be free and that even though revenues should not be derived from software, the code will be improved by volunteers who willingly make their work available for anyone's use and reference.

[4] A number of major vendors have also increasingly made their software, e.g. development tools, available as open source. See http://www.zdnetasia.com/news/software/0,39044164,39239409,00.htm.

From the customer's perspective, the value that a customer derives from a commercial software product typically correlates with the licensing fee, software functionality and product support. While the customer can correspondingly hold the commercial software vendor directly accountable for the software, there is no "owner" of most open source software and thus it is difficult to assign accountability. A number of open source vendors have approached this concern by providing customers with similar assurances through licensing terms and the payment of corresponding fees[5]. In making their procurement decisions, customers from businesses and governments should weigh each one of the above factors according to their individual needs, concerns and environment.

## Development

Another factor that has historically distinguished open source and commercial software is the approach taken towards the development of software. This is rapidly evolving and converging with each model adopting some practices of the other.

Commercial software development teams historically work within the confines of a single organization or unit for the primary code development. With open source development, typically there is a structure for the involvement by

[5] In instances where third party vendors undertake to provide to customers accountability or assurance for specific implementations of open source software, it is important for the customer to understand the applicable limitations and conditions under which such accountability is provided, e.g. it may not extend to situations where the customer modifies the software himself. This is to be expected as such vendors offer value through testing, support, maintenance and upgrades that are implemented on a stable and known source, and changes made independently by a customer will limit the ability for the vendor to provide guarantees.

a wide variety of players. This dichotomy is converging to a common model of development. Today, commercial software development teams have developed structures to collaboratively develop software with teams that span the globe. There are also open source software solutions originating from a single company or programmers backed by commercial vendors. These open source solutions are contributed to the community and maintained by just one or two key contributors. In both the commercial and open source software development approaches, the common underlying development process is an iteration of design, standards, coding, testing, release and feedback. There is a core set of programmers who write the program and release it to a community for beta testing. The beta testers run the program and report back to the programmers on bugs and patches. The programmers then change the source code to solve the problems identified before the software is released generally.

Experience in the industry over the years indicates that a proper framework that facilitates software development by multiple teams or contributors and their diverse perspectives can lead to a rapid rate of innovation, optimization, vulnerability-fixing and timeliness to market. Both open source and commercial software development teams adopt such structures in many of today's software development projects.

For the programmers, both the commercial and open source development environments contribute towards their individual growth and skill development. Foundational computer science concepts have been articulated in textbooks for many years and such texts are continually updated resulting in a wealth of published information available to students. Traditional methods of teaching based on such texts have produced good programmers over the many years since software was first written. The key is in the effectiveness of the educators and professors in teaching students the relevant concepts in a way that the students learn to write their own code to implement those concepts, and not in the students having access to source code merely to copy from. Skilled programmers, whether developing with the open source or commercial software models, are able to gain recognition in their own right for their contribution to software development as they solve unique, difficult or complex problems.

## Licensing

The primary underlying difference between the open source and commercial software models is in the licensing of software. Commercial software providers typically adopt the traditional software licensing approach where permission to use the software is granted to a customer in return for a fee. The customer is usually permitted to use, reproduce or adapt the software only according to the terms of the license.

Open source software is made available under a variety of licensing approaches with certain common features such as the right to modify and the right to redistribute the software. The copyright within the software is the foundation of the licensing contract, just as it is the case with commercial software. Rights and permissions are granted subject to conditions. In general, these conditions restrict how the software may be further changed or distributed, rather than impose a requirement that a fee be paid for it. There are

two principal open source licensing approaches – the GNU General Public License (GPL)[6] and the Berkeley Software Distribution (BSD) License[7].

Under the GPL, all derivative works of the software and subsequent versions down the chain must be licensed and distributed on the same terms as the original software. Source code subject to the GPL permanently remains subject to GPL. This permanent nature of the GPL, as intended by the authors of the GPL, constrains the options available to developers building on GPL software in creating, distributing or commercializing products using existing GPL source code. There are also other potential challenges faced by developers, for instance in determining when software developed for a GPL software platform is considered a derivative work that is subject to the GPL.

In contrast, under the BSD License, developers have the freedom to integrate the licensed software with the developers' own source code to create new products with few restrictions. The BSD License, for example, allows programmers to use, modify and redistribute source code and binaries of the original software. However, unlike the GPL approach, programs containing code subject to the BSD License do not have to be distributed under the BSD License. Derivative works can be distributed either in an open source manner, or under a more traditional commercial license. Where a purchaser acquires the software only for his own use and does not intend to build on and redistribute the modified software, the distinction between GPL and BSD is not as essential.

[6] See http://www.gnu.org/copyleft/gpl.html.
[7] See http://www.opensource.org/licenses/bsd-license.php.

The distinction is however important to a software developer that writes code for commercial purposes. The GPL prohibits charging money for the distribution of source code, other than to cover the administrative cost of copying and shipping. While the GPL permits the open source software to be sold to a customer for a fee, the license and the access to the source code allows customer to freely redistribute or modify the code without further payment to the original party the code is acquired from.

Charging fees for system setup, system management, support, maintenance and other related services is permitted under the GPL. It is on this basis that commercial support services for Linux – which is licensed under the GPL – are offered by companies and used as one of their revenue sources.

## Intellectual Property

In the knowledge economy, one of the most important assets that we can leverage is our intellectual assets – data, information, knowledge and other intellectual creations such as software and other inventions. The continued growth of the overall software economy is founded on the protection of such intellectual property rights. Without such protection, software owners lack the incentive and legal basis for commercializing their creations, and the software industry cannot be an engine of economic growth.

Although open source software is often available for free download, it does not follow that there is no need for intellectual property rights protection under the open source software model. On the contrary, open source licenses impose terms and

conditions based on elements of intellectual property protection (as illustrated in the Netfilter example that follows). In addition, the use of the source code from open source developers is often permitted on the condition that there is an appropriate attribution to the author of the original source code.

In the 2004 German case of Netfilter Project against Sitecom Germany GmbH[8], a three-judge German court recognized the GPL by requiring the defendant company to disclose the source code of its product that relied on and incorporated components of open source software written by the plaintiff. It is worth noting that in this case, the plaintiff's grounds for the action were founded on copyright. The plaintiff was not in a position to take legal action against other alleged violators of the GPL, since in those cases, the plaintiff did not own the copyright in the works that were used by the other violators. This illustrates that in both open source and commercial software models, the need for intellectual property rights remains the foundation for the license conditions to operate. Hence, irrespective of the software models adopted, the existence of a sound intellectual property rights regime is essential for the software industry.

## Policy Considerations

As we will see further in the second part of this discussion, the arguments advocating any particular software development model may, on closer examination, not be as unequivocal as they first appear to be. Choices and procurement policies should be made based on value-for-money and fit-for-purpose considerations, and should not be made or preferred based on broad categorizations that do not ultimately support sound objective principles. The benefits and appropriateness of the considerations relating to the open source and commercial software models arises according to the customers' needs in particular situations and not as generic "truths".

**Technology innovation is best accomplished by a healthy, competitive and diverse marketplace that allows software companies to develop and grow according to their own strengths and capabilities.**

At the broader policy level, it should be recognized that there is a need for choices in software selection and implementation, and for skills development to support a growing diversity of software development models. Technology innovation is best accomplished by a healthy, competitive and diverse marketplace that allows software companies to develop and grow according to their own strengths and capabilities. Procurement policies of organizations and Governments have a key role to play in maintaining this diversity that is essential to the growth of the software industry. Fair and open competition, and not procurement preferences, should determine which products earn the confidence of consumers. Rigorous competition ensures that technology providers have the incentive to invest and produce the best products for the market, which in turn means broader consumer choice among many innovative technologies.

[8] See http://www.netfilter.org/news/2004-04-15-sitecom-gpl.html and http://news.com.com/2100-7344-5198117.html.

## Part II : Technical Concerns – Cost, Security and Flexibility

In part two of this discussion, we will take a closer look at some of the specific arguments that the proponents of open source and commercial software make in relation to the comparative strengths of different software models.

### The Debate

The ongoing debate between commercial software and open source has sometimes centered on whether one approach to the software licensing and development model is inherently superior to the other. In fact, these models each serve the specific needs and circumstances of the individual customer environment where the software is to be deployed. Such needs and circumstances determine what factors are relevant and applicable, and whether certain advantages and disadvantages of the open source or commercial software model should be given more weight and consideration.

#### Cost

Whether open source software is cheaper than commercial software for a particular customer should be determined in the context of the lifetime costs of a product. While some open source proponents say that open source software is cheaper than commercial software, proponents of the commercial model also point out that the total cost of ownership can be less for commercial software than for open source software with similar functionality.

**Technology decision-makers should weigh the full range of costs, including lifetime costs and migration costs, when evaluating their own choices in this area.**

In terms of the initial purchase price, open source solutions may be cheaper than commercial software. However, in making buying decisions, consumers must also consider the cost of software during its entire lifecycle, rather than the one-time purchase price. Just as consumers weigh the long term costs of buying cheap mobile phones with recurrent subscription packages as opposed to more expensive phones with lower recurrent costs, they should not determine the cost of software merely based upon the initial purchase price. They must also take into account long-term support and maintenance needs, in addition to other less tangible issues such as usability of the product and productivity gains. Purchasers should also consider the cost of retraining users familiar with one product to become competent in an alternative product. Such retraining costs may be quite significant when one takes into account the total time spent by the users undergoing such retraining and the initial lower productivity levels while the users familiarize themselves with the alternative product.

Not surprisingly, there are numerous competing studies and surveys in this area, each considering a different set of cost factors, and correspondingly drawing different conclusions as to the relative costs of one model versus the other. What is clear, however, is that technology decision-makers should weigh the full range of costs, including lifetime costs and migration costs, when evaluating their own choices in this area.

## Security

It has been argued that open source solutions, whose source code is available for public scrutiny, are inherently more secure than commercial software solutions, whose source code is not published. There are also those who argue that where source code is published, it is easier to find and exploit flaws in software, and also others who say that source code access is irrelevant to software security. The viewpoints are wide-ranging.

Security of software is an industry-wide concern, and vulnerabilities will always be present in software. Vulnerabilities affect all complex software programs and are not more or less prevalent for software developed under an open source or commercial software model. The issue is in how to minimize and remedy the vulnerabilities, not which licensing model leads to more secure software. Criminal attacks against software remain a critical consideration in the minds of all users, and it is therefore important to understand the underlying principles in this area.

The security of different commercial and open source software offerings varies considerably. While the design of security-related features matters significantly, total security depends just as much on how well the software is deployed, configured, updated and maintained, including whether product vulnerabilities are discovered and resolved through appropriate and timely updates. These variables are contingent on the customer taking due care, and not on the licensing or development model. The three key factors related to the security of software are the quality of the developers, the techniques and tools used by the development team to reduce vulnerabilities, and the strength of the relationship between the customer and the software provider.

In considering whether having source code available for public scrutiny makes the code more secure, one needs to understand the genesis behind this argument. It has often been said by those concerned about security that security cannot exist by reason that how it works (i.e. the security mechanism) is kept under wraps – “security by obscurity is no security”. This belief has its roots in the early days from researchers developing encryption technologies for scrambling data against interception. It follows that one should not take for granted assertions that a “black-box” product is secure because no one knows how it works. However, it does not mean that because the source code of a product is known, it must therefore be secure. Good security is not dependent on whether the mechanism (in this case, the source code) is known or published, but on how the mechanism is designed, implemented and managed by qualified security specialists. A product’s security depends on whether qualified persons have reviewed and tested the security mechanism to minimize the number of vulnerabilities that might be exploited by criminals, not on whether the source code is available.

In the open source community, where voluminous quantities of source code are available, it is not realistic to expect that every single line of the code has been developed according to a methodology that is known to reduce vulnerabilities or that it has been scrutinized by a wide range of security experts.

New packages and add-on software are developed and distributed regularly, but they may not have undergone the same level of scrutiny as that of other more established packages. It has been found that it is difficult to attract volunteer "eyeballs" that are qualified for (or interested in doing) the difficult job of code review for security vulnerabilities[1]. Users who obtain open source packages are likely to compile, install and use the software even before taking a look at the source code, even though it may have been provided with the software. Only a very small minority of the community will indeed scrutinize the source code of every program before any use. Even then, few code reviewers or writers are trained or skilled in identifying security vulnerabilities by merely reading the code. A good programmer does not necessarily make a good security specialist. Some open source solutions have vulnerabilities that have remained undiscovered[2] for years notwithstanding public availability of the code. It has also been shown[3] that having access to source code does not prevent a backdoor from being hidden and remain undetected in software. Given such a scenario, it appears premature to declare that open source is superior in security.

**A poorly maintained product offers little security, regardless of the software development model used to create the product, or the rigor to which the software was tested.**

The concern over the lack of access to source code is increasingly also being addressed by commercial software vendors who make available such source code for specific purposes. For instance, a number of Asian governments and security agencies have had access to the source code of commercial software products, and have undertaken security review of such products. This access provides the government with the opportunity to rigorously review the source code of the product, as they might also do so with the source code of an open source solution. With time, there are also increasing instances of the source code of other commercial software products being released publicly. The disclosure and availability of source code for the purposes of undertaking a security review is hence no longer a compelling basis to prejudge commercial software and open source solutions[4]. It should also be noted that the availability of source code can also be a double-edged sword – both trusted and untrusted entities have access to the code, and this can be a security risk. Hence, different developers choose alternative approaches, each attempting to achieve greater security.

The speed through which vulnerabilities are

[1] The Sardonix Project was backed by the U.S. Defense Advanced Research Projects Agency (DARPA) and aimed to set up a public website to meticulously track which open source code have been audited for security buys. There were no volunteer auditors even after two years. See http://www.securityfocus.com/news/7947.

[2] Websites such as http://www.viruslist.com provide information on vulnerabilities on a variety of commercial and open source platforms.

[3] See the classic paper by Ken Thompson, one of the fathers of UNIX, "Reflections on Trust" (http://www.acm.org/classics/sep95), where he notes that no amount of source-level verification or scrutiny will protect against untrusted code. The open source process cannot find clever subversions, no matter how many people look at the source code.

[4] It should also be noted that the software landscape comprises of different levels of disclosure of source code. Just as some commercial software providers may decide to fully disclose their source code in some circumstances, core teams of open source developments can also choose not to disclose certain aspects about their code in other circumstances. In both software models, there are instances where the software developers do not make full disclosure, e.g. where they disclose vulnerabilities after a patch is available.

addressed and remedied is also a point of contention among proponents of different software development models. Neither development model necessarily addresses vulnerabilities faster than the other. More importantly, the assumption that a quick fix is a good fix is not necessarily the case. For some customers, it is important that updates or patches for identified vulnerabilities be thoroughly tested in different environments and configurations before they are rolled out, lest they lead to other system stability issues. For others, it is essential for the customer to have accountability from the software provider for the quality of the updates or patches provided[5]. The speed of providing an update or patch is not always the most important criterion, just as it is not always the case that either software model has a faster turnaround time on updates or patches than the other. Conclusions drawn from comparative studies of the security of products indicate that assumptions generally made about the speed of fixes available for different operating systems may not necessarily be accurate[6]. The duration before fixes are made available may also be dependent on a range of factors not related to the software model, including the necessary testing that needs to be done before the fix can be released.

**The security of any software product and implementation is not pre-determined by the method of development or distribution.**

Another misconception in this area of security focuses on the assertion that because the criminal exploitation of vulnerabilities in proprietary software is sometimes more widely felt than the exploitation of vulnerabilities in open source software, then proprietary software must therefore be less secure. Here, it should be borne in mind that the impact of software exploits has more to do with the popularity of the software, than with its design or the method of software development. The program that is most commonly used is frequently the program that attracts the most interest among cyber criminals, and will be the platform that is more likely to be attacked, as hackers have a greater incentive to hit a larger target than a smaller one.

Security issues result from a combination of factors, including the software design and implementation as well as user behavior and usage, coupled with the skill and expertise of the user in installing, deploying and maintaining the software. Anecdotal experiences relating to the number or perceived impact of attacks targeting commercial software are not necessarily testimony to the notion that open source solutions are less vulnerable or that commercial software solutions are more vulnerable. A technological product can only be as secure and as reliable to the extent that the necessary care was taken to properly install and maintain the product. A poorly maintained product offers little security, regardless of the software development

[5] I.e. some customers may not want just any third party to provide that update or patch and have neither the third party nor the original software provider bear any responsibility for the patch or fix, but they want to hold the original provider accountable.

[6] It is sometimes assumed that open source developers respond more quickly to vulnerabilities than commercial developers. However, studies done (see for example the report by Forrester: “Is Linux More Secure Than Windows?” (at http://www.forrester.com/Research/Document/Excerpt/0,7211,33941,00.html and http://www.linuxworld.com.au/index.php/id;554502920;fp;2;fpid;1) have concluded that on average, Linux distributors has taken longer than Microsoft to patch security vulnerabilities.

model used to create the product, or the rigor to which the software was tested. Simultaneously, the standardization around a platform can simplify and accelerate the security updating processes.

Therefore, one cannot draw any reliable conclusions about whether a product is more or less secure on the basis of the software's development or licensing model.

The security of any software product and implementation is not pre-determined by the method of development or distribution, but by the proper design of security features, and equally important by the correct deployment, configuration and maintenance of the software by the customer. Today, we see software licensed under open source and commercial models that has been developed with security in mind. Developers are using methodologies known to reduce vulnerabilities through up front assessments, rigorous and organized testing, and post release response centers that assess vulnerabilities and provide updates. How the software is licensed is not a significant factor in this process. Each product and implementation should be assessed on its own merits and strengths.

In recent years, software developers across the industry have redoubled their efforts to address consumer concerns and security, and security issues have featured prominently in the decisions and choices made by software companies, as well as consumer organizations, There is a strong motivation and incentive for commercial software developers to undertake hard and resource-intensive testing of their code and updates or patches for security, and this is increasingly true of open source software developers and distributors who are interested in commercializing their products. Ultimately, good code is good code, regardless of whether the source is open or not.

## Flexibility

The argument that open source solutions are more flexible for customers than commercial software stems from the ability of a customer to examine the source code and make the necessary alterations to the code. This also allows technically-savvy customers to potentially identify any problems in the system and make their own changes or fixes to the software to rectify the problem.

There are resources available for both commercial software and open source solutions to correct the security vulnerabilities in software. As explained earlier, there is no clear advantage with either model when it comes to rectifying known security problems. Updates and patches for commercial software are typically available only from the original software vendors. They have an incentive to ensure the reliability and trustworthiness of the updates and patches released, since their credibility will be severely affected if their use results in more problems. Updates and patches for open source solutions come from a greater variety of sources. They may be developed through community effort and distributed through channels such as discussion groups. Such updates and patches may be quickly available, and they are iteratively refined and improved on by the community if they initially do not correct the problem completely, though

they may be subject to less rigorous testing before they are released.

However, for lesser known open source software that is not widely used or supported by commercial companies, the flexibility to allow people other than the original software vendor to contribute solutions to identified vulnerabilities may not necessarily be desirable for business customers. Although updates and patches may come from a broader community, such flexibility reduces the accountability of the original software vendor to the customer for the problems in the software. Community contributors of such "public" updates or patches do not usually bear the same accountability and responsibility as "commercial" updates or patches that are distributed by commercial software providers. Certainty and predictability of business operations are essential to the efficiency of business operations. Flexibility that leads to the absence of accountability by the software vendor and the lack of predictability of vendor support can translate to a serious operational hurdle for businesses.

The flexibility to modify source code in an open source solution also leads to another phenomenon known as "forking". Forking occurs when one developer decides to modify the software source code and takes a path that is divergent from the original software such that any subsequent changes or improvements made to one version of the software will not apply to the other version. Issues of compatibility and continuity will therefore arise and need to be managed when forking occurs. A notable example of forking occurred in the early days of UNIX when different hardware vendors produced different variants of UNIX for each of their platforms, e.g. System V, BSD, AIX, Solaris, HP-UX, etc.

**The main issue to consider is whether there is a need to customize the acquired software to meet specific needs.**

On a smaller scale, customers who make their own modifications to the software will also find that the continued support and maintenance of such changes becomes a more involved process, as the support resources need to be equipped with the knowledge of the prior customization, as well as the skills needed to perform subsequent alterations. In contrast, commercial software solutions tend to have a more well-defined and controlled upgrade and migration path for products. Customization built on such platforms using the published application programming interfaces often will continue to work with upgraded and future versions of the product with little or no changes.

Ultimately, in considering whether flexibility is important in making a software choice, the main issue to consider is whether there is a need to customize the acquired software (whether at the application or operating system level) to meet specific needs. If customization is required, the user needs to obtain additional resources to maintain such non-standardized customization from the original release of the acquired software. If customizations are not done to the acquired software but are instead built in addition to or at a layer above the software, the software model of the acquired software is, in that scenario, not a

## Key Actions

In light of the issues highlighted, in considering options between open source and commercial software choices, the following should be kept in mind:

- Cost considerations should be viewed in totality. While cost is an important issue, it is usually not the sole determining factor for a procurement decision. In some situations, such as in mission critical or public safety systems, or in places where manpower costs are substantially higher than software prices, cost may even be a subsidiary concern.

- In any software deployment, the total required manpower should not be underestimated. Options are available in the market today for suitably skilled and trusted manpower for the support of a software platform to be retained in-house or obtained from an outsource vendor. It is essential that the entire range of manpower required be taken into account in the evaluation and selection of the software product. Such manpower includes resources required for support, maintenance, training, data migration and conversion, integration with legacy systems, enhancement of systems, managing incompatible systems, etc.

- To enable the use of a product securely and reliably, there needs to be a shared responsibility between the customer and the software provider. The software provider has the responsibility to develop the software in accordance with best practices in security, to rigorously stress-test the software and to develop updates and patches rapidly when vulnerabilities are subsequently uncovered. On the part of the customer, suitable and adequate resources should be allocated to ensure the correct installation, deployment and maintenance of the software. Updates and patches, when made available by the software provider, should be applied in a timely manner. A technological product is only as secure and reliable as the extent to which the users have taken the necessary care to properly install and maintain the product. A poorly maintained product offers little security, regardless of the software development model used to create the product.

- If a security review of the source code is required, appropriate expertise should be made available to meaningfully scrutinize the source code of the components to be deployed. It should not be assumed that because the source code has been made publicly available that it has, in fact, been sufficiently reviewed.

- Requirements for flexibility in modifying the acquired software should be carefully considered against whether the expertise to exploit such flexibility is available, and if the necessity for flexibility is fundamental or merely incidental. The long term support implications for non-standardized modifications to the software should also be factored into the purchase decision.

relevant consideration.

## Co-Existence of Open Source and Commercial Software

In the discussion above, the arguments that have been commonly raised regarding open source and commercial software solutions are analyzed with a view to distilling their validity from their rhetoric. The analysis demonstrates that both open source and commercial software have their strengths, and they both have a place in the market.

As we have seen from the first part of this discussion, the open source and commercial software development and business models have complemented each other in a number of aspects. Nonetheless, the debate continues with regards to which model is “superior”. Having examined the issues in greater depth in this second part of our discussion, we have aimed to clarify that the issue of which model is “better” lies with the specific circumstances or requirements that a customer may face, rather than in the inherent nature of either model.

In the third part of our discussion, we will delve deeper into the pertinent issues that are facing Asian governments in this area, and we will consider and analyze the effectiveness of some strategies that may be adopted to address concerns of consumers and the industry.

## Part III : Policy Concerns – Piracy, Digital Divide and Domestic Industry Development

In this third part of our discussion, we will examine the policy considerations surrounding the open source and commercial software debate that are of particular concern to Asian Governments and economies. With a deeper understanding of the issues involved, one can better discern what would be suitable strategies and responses to these important concerns.

### Government Concerns

In recent years, Asian Governments have taken an active interest in open source software as a possible solution to emerging concerns within their domestic markets. Such concerns include a desire to address software piracy problems, to cope with digital divide issues, and to develop an indigenous software industry as a bulwark against a "dependence" on foreign sources of software. We will discuss each of these perspectives in greater detail below.

**Both open source and commercial software products are predicated on strong copyright protection.**

#### Software Piracy

The cost of software has often been cited as a primary impetus for software piracy, with critics charging that consumers in developing countries have no choice but to look to cheaper, pirated products to meet their computing needs. Similarly, some have argued that the adoption of open source could potentially lead to a reduction of piracy levels, since open source solutions may be freely copied and distributed legally, typically in a royalty-free manner.

To understand the economics behind the pricing of software, one needs to consider that a software product is unlike other consumer-oriented commodity products. Software is a productivity tool for businesses and households, providing the consumer with different levels of productivity gains. Such gains are possible through the investments made in the research and development of software products. Just as one does not expect the cost of professional services to be pegged at a fixed rate regardless of the experience and qualifications of the professional engaged, the pricing structure of software should not be expected to be pegged at a commodity product level, irrespective of its utility.

In countries where piracy rates are high, the arguments relating to the cost of software and the advocacy for software prices to be lowered do not address the crux of the issue. There remains an underlying necessity for consumer education regarding the need to respect intellectual property rights and not to engage in software piracy or other forms or intellectual property theft. If high pricing were the primary reason for software piracy, there would not be the phenomena of piracy of other already lower-priced commodity products such as music, videos and games. Experience has shown that there is no particular correlation between the price of software and the levels of software

piracy. Some of the most commonly pirated software products, such as anti-virus and other utility software, are among the least expensive. Even software available for download at no charge can be found in retail pirate outlets.

The notion that promoting open source software will eliminate software piracy is also erroneous. Increasingly, computers have been shipped with the option of being pre-installed with open source operating systems and office productivity suites as an alternative to their commercial software equivalents. This does not mean, however, that these computers will not at some point in their life cycle contain pirated software. In fact, some studies[1] indicate that a high percentage of computers shipped with open source solutions may eventually be replaced with pirated software, a pattern that is also seen in computers that are not shipped with open source software.

Both open source and commercial software products are predicated on strong copyright protection. Open source software products, like their commercial counterparts, set forth licensing terms and conditions that dictate how these programs may be utilized, modified, and distributed. Moreover, the increased use of certain open source products does not necessarily mean that consumers or businesses will use these products in lieu of commercial options. On the contrary, it is quite common to see open source and commercial solutions being used in the same environment, often in a complementary fashion. A user's reliance on an open source operating system, for example, does not mean that he will forego the use of commercial applications on that operating system. These applications, in turn, must continue to be protected against piracy.

Piracy can also manifest itself with open source software in a different form. Just as the lack of respect of intellectual property lead to commercial software products being pirated through illegal copying, similar mindsets may also lead to the source code of open source software to be pirated by unscrupulous developers – by incorporating open source code within the code of proprietary solutions in a manner contrary to the governing open source license, and potentially passing off copied code as their own. Ignoring the terms of an open source license is also piracy.

Software piracy is therefore not a problem that will be eradicated through the increased use of open source software. While open source solutions may provide cost benefits to consumers in certain instances (as discussed earlier in the first part of this series of articles), open source is not by definition the most appropriate or the cheapest option for consumers in every instance. At its root, Governments, especially in developing economies, would benefit from bringing about a change in mindset and attitudes towards piracy, and encouraging the recognition of the value of intellectual property and the need to protect intellectual property as an asset essential to a country's information economy. Without a

**Software piracy is therefore not a problem that will be eradicated through the increased use of open source software.**

[1] See "Gartner: Piracy driving Linux PC Shipments: at http://www.infoworld.com/article/04/09/29/HNlinuxpiracy_1.html and http://www.linuxbusinessweek.com/story/46582.htm.

fundamental appreciation of the importance of intellectual property to a nation's economic growth, the mere promotion and adoption of open source solutions may not, in and of itself, lower piracy levels in a particular country, nor necessarily create an environment that is conducive to the growth of a domestic software industry.

**An effective strategy to address digital divide issues should ensure that all aspects of preparing the general public for the information age are addressed.**

Indeed, research[2] has shown that countries with the most robust local software industry also have the lowest piracy rates. High piracy rates inhibit the development of a domestic software industry to create solutions to meet local software needs[3]. We will further discuss the issue of local software industry development in a subsequent section, after first looking at the issues surrounding digital divide.

## Digital Divide

The need to bridge the widening digital divide in some developing countries has led to the introduction of initiatives to make low cost personal computers available to the general public. Such low cost personal computers are often offered with the option of being installed with open source software so as to reduce the initial cost of owning the computing technology. Increasingly, commercial software vendors have also made available low-cost and localized-functionality versions of their software in support of such low-cost personal computer initiatives.

In considering a strategy to address the digital divide, a Government's agenda should go beyond the mere provision of computers to low income families. Of greater importance is the need to have in place a comprehensive program that will equip such targeted families with the necessary information technology and software literacy skills so as to empower and enable these individuals to make use of the technology. At the same time, infrastructural needs also have to be addressed, for instance, to ensure that there is adequate and appropriate content and applications that are relevant to the general public (e.g. suitable e-government services), and to have available reliable connectivity for the public to access such content.

Lessons and best practices drawn from case studies have illustrated that the challenge of the digital divide requires a strategic and multilayered response. As concluded by a report[4] that examined digital divide issues in a number of developing regions, including Africa, South America, and India, providing access to technology proved critical, but the need for access went far beyond mere physical access to a computer or network connection. The study concluded that the benefits of such computers and connections would be lost if users lacked the knowledge to operate those technologies effectively. Access should be considered more broadly in the context of integrating technology into people's lives, according to this study[5].

[2] See study by PricewaterhouseCoopers, "Contributions of the Packaged Software Industry to the Global Economy" (page 14) at http://global.bsa.org/usa/globallib/econ/pwc1999.pdf.
[3] Conclusion found at section 5.3 at page 14 of the PricewaterhouseCoopers' study.

[4] See "Spanning the Digital Divide: Understanding and Tackling the Issues" at http://www.bridges.org/spanning/index.html.
[5] Conclusion found at section 4.3 on "Drawing out Lessons and Best Practices" at http://www.bridges.org/spanning/chpt4.html#_Toc515100166.

In keeping with these observations, an effective strategy to address digital divide issues should ensure that all aspects of preparing the general public for the information age are addressed. The strategy should not, as a first step, be focused solely on the provision of low-cost computers to the homes, but rather, should take a holistic and multi-prong approach covering skills, connectivity and content to ensure that such computers would be useful to those who have access to them.

## Domestic Software Industry Development

The information technology and software industry is regularly viewed as a key strategic sector for countries to develop and cultivate. With the wide availability and adoption of software products from multinational companies, some Governments have viewed the success of foreign developers relative to local competitors as an indication of an unlevel playing field. Consequently, Governments sometimes feel the need to create policies that purport to level the playing field. In this vein, some Governments are considering procurement preference policies in favor of local companies. Other Governments have welcomed the availability of source code from open source software solutions as a way to jumpstart their domestic industry. Their desire is for local players to make use of the available source code to develop and build their own software solutions. In regions with strong histories and cultures of communal settings, a spirit of sharing and openness may be seen as the preferable approach to promoting a local software industry.

While the desire to promote and develop a domestic software industry is understandable, Governments need to identify and consider what inherent background and advantages each particular country has to build on to create niche opportunities, and have a clear perspective on what policies will actually result in the creation of such an industry. In particular, Governments need to carefully examine whether promoting one specific software development model over another will necessarily bring about the end benefits desired.

The software industry can be broadly segmented into off-the-shelf products, customizable products, custom-built products and embedded software products. The software solutions that most consumers use on a daily basis are off-the-shelf products. Larger organizations also use customizable products that are tailored to meet their business operations. For very specialized requirements, a software product can also be custom-built for a customer. Embedded software solutions are typically created in conjunction with hardware innovation.

In light of the different software industry segments, Governments promoting and steering the growth of the domestic software industry should have a broad understanding of the strengths of their industry and the appropriate segment to cultivate. Off-the-shelf products typically generate revenue from the software, and companies venturing into this segment need a good understanding of the mechanics of different software licensing models. In particular, the General Public License (GPL), upon which the majority of open source software is based, has an important caveat on the commercial exploitation of new software products that are adapted or evolved from programming

code that was previously subject to the GPL. There is a need to ensure that companies understand the different business and licensing models and their constraints, as well as issues relating to indemnity, warranty and liability related to the use of software. The same considerations apply to embedded software.

Where Governments advocate that companies pursue open source with the desire to develop the domestic software services industry, there are indeed some benefits that may be derived for providers such as system integrators. System integrators can build on existing code from the open source community to customize solutions for customers in the form of customizable products and custom-built products. However, in practice, software companies in the service-oriented business are capable of providing support or other professional services to the market based on both the open source and commercial software approach. There is little reason for Governments to ask such companies to focus their business to only one software model, so long as the skills and resources are available to the company.

Another important consideration with the greater availability and use of open source software is that the domestic software industry needs to be much more vigilant about tracking the incorporation of any external sources of licensed code within their own open source and commercial software products. There is value in exercising prudence in the use of such code and to conduct regular audits to ensure that a product that incorporates external source code is used in a manner that is consistent with any applicable license, whether open source or commercial[6], thus minimizing potential liabilities.

Governments ultimately need to understand the upstream and downstream effects of their choices in formulating policies that aim to cultivate the local software industry and to help their domestic companies move up the software development value chain to become leaders and players in their chosen fields. For instance, while it is common for Government funding to be provided to companies that kick-start open source development efforts, companies adopting open source models must eventually have a revenue stream to be able to become financially sustainable. The Government would not have achieved its objective of developing its domestic software industry if companies developing open source products are unable to sustain themselves commercially without continuous Government funding of open source projects. Governments should identify and build on their country's background and inherent competitive advantages. In essence, there needs to be a holistic underlying economic strategy behind the push to develop a domestic software industry.

**Governments ultimately need to understand the upstream and downstream effects of their choices in formulating policies that aim to cultivate the local software industry.**

## Competition Development

The presence of competition in a market has a

[6] See International Herald Tribune article at http://www.iht.com/articles/2004/12/28/business/code.html for a further discussion on how the use of open source software may be detected and dealt with, and how liability may potentially arise in some circumstances.

direct impact on the efficiency of the companies operating within the market, and in the long term, on the benefits that consumers may receive from the players in the market. For example, it has been the consistent experience of countries that have liberalized their telecommunications market that the incumbent telecommunications operator quickly becomes more cost-efficient in the face of more competition, and consumers experience significant cost savings while obtaining better service quality[7]. In the software market, the availability of open source alternatives competing with commercial solutions have also made commercial software vendors more responsive to consumer needs, delivering better and more innovative products and services. As a public policy goal, a healthy competitive environment is desirable, as it brings about greater market efficiency and more choices for consumers. Consequently, companies in economies that are not subject to the discipline of market forces may find that there is less incentive to maintain cost efficiency, and thus these companies risk becoming complacent.

**As a public policy goal, a healthy competitive environment is desirable, as it brings about greater market efficiency and more choices for consumers.**

It is commonly accepted that companies should be allowed to freely compete with each other to develop the best and most innovative product or service, through their own chosen method that allows them to best deliver their product or service. At the same time, the Government's role in molding the competitive landscape should be limited to only the most necessary of circumstances, lest the competitive spirit of the industry be quelled. Often, the industry (propelled by consumer demands) is more knowledgeable than Government policy makers regarding the direction that the market should take. Further, experience in many countries has demonstrated that heavy regulation may stifle the domestic industry rather than cultivate growth. For instance, within the United States, in the age of converging computing, telecommunication and media industries, the existence of a largely unregulated Internet and information technology industry has resulted in explosive growth and innovation over the past few decades, particularly when compared to the regulated telecommunication and broadcast industries.

The issue of software choice often emerges in discussions relating to competition in the software industry. It should be noted that, a consumer's inertia to change from one software solution to another can be a factor in determining whether such a change takes place. The inertia may arise from the cost of change, and not necessarily from the inability to change (or the lack of choice). At the same time, the lapse by a software provider to continually innovate will result in competitors replacing the incumbent with better products to meet the growing needs of consumers. The intervention by Government policy to pick winners or to constrain any player or industry segment goes against the principles of fair competition and free choice. Such actions will can discourage and harm the industry and inhibit

[7] See, for instance, the Closing Statement made at the OECD Global Conference on Telecommunications Policy for the Digital Economy (www.oecd.org/dataoecd/52/35/1810903.pdf) and the paper "Liberalizing Telecommunications: The Asian Experience" authored by members of the World Bank's Research Development Group (http://rru.worldbank.org/Documents/PapersLinks/1441.pdf).

the benefits that may otherwise arise out of competitive market forces.

### Sovereignty

For Governments seeking to implement an industrial policy aimed at growing a domestic software industry, a desire for sovereignty and the avoidance of a dependence on foreign technology are often motivating factors.

In this regard, it is worthwhile to consider the history of industrial development. There was a time when Governments felt the need to build their own steel industries to meet domestic needs and to address sovereignty concerns. Today, little thought is given as to where steel is actually produced, as long as steel is widely and cheaply available. By the same token, there was a time when each country felt the need to have a national airline. However, after the last financial crisis and economic downturn, there was a significant consolidation in the airline industry, and several carriers, including national carriers, ceased operations.

To draw a possible parallel for the software industry, one can look to the aircraft manufacturing industry. The manufacturing of aircrafts, like the manufacturing of software, is a specialized skill that can be developed. For many years, Boeing was the primary provider of aircrafts to the world, until Airbus was formed and eventually grew to become a formidable competitor. Today, airlines around the world continue to purchase their aircrafts primarily from these two companies, although there are a number of smaller aircraft manufacturers. There are clearly dominant players in this market, but there is keen competition between the companies. Looking at the players in the market, the competitive landscape of the aircraft industry is significantly more limited than that of the software industry. However, notwithstanding the importance of air travel to a country's economy, aircrafts continue to be acquired from Boeing or Airbus rather than nations endeavoring to create their own domestic aircraft industry, so as to avoid a dependence of the national airline on a foreign supplier of aircrafts.

Though there are differences in the aircraft and software industries, it is still worth pondering the reasons why countries feel compelled to build a domestic software industry with the primary objective of avoiding a dependence on foreign sources of technology, yet they do not feel similarly compelled in other areas. In better understanding some of these reasons, countries may perhaps be in a position to better capitalize on their inherent advantages and identify ways to work with other international players (rather than isolate the domestic players) in a collaborative manner for mutual benefit in the long term.

## Government Response

The above discussion illustrates that the policy concerns facing Governments in this area are multifaceted and complex. The important lesson that emerges is that there is no panacea to address all the challenges faced. Understanding some of the complexities involved may place us in a better position to consider the effectiveness of some strategies that are being considered by various Governments around Asia.

# Part IV : Looking Ahead – Technology Neutrality, Interoperability and Standards

In this fourth part of our discussion, we will evaluate issues relating to software procurement and technology standards currently being contemplated by a number of Governments. For decision-makers looking ahead and formulating policies for the future, there is a need to recognize the underlying importance and fundamental nature of software innovation to the success of the technology industry, especially given that it is through software that other computing technology becomes useful and productive.

## Software Procurement Preferences

Factors regarding the desirability of each software model may lead a Government to consider adopting policies and strategies that are aimed at providing an advantage to one software model over another. Such policies may include a procurement preference policy or other national preference policy, e.g., a preference policy for funding research and development that adopts any one software model. The ostensible intent of such policies is perhaps to shift behavior and attitudes towards a particular software model, with a view that it will bring about advantages such as lower costs and wider vendor choice. There may also be other motivations, such as a desire to develop an indigenous software industry and domestic products so as to be free of a dependence on foreign providers.

It should be recognized from our earlier discussion that open source and commercial software each has its strengths, and that it is not the case that either software model is inherently superior to the other. Rather than attempting to force-fit customer requirements through artificial constraints, the better approach is for the consumer to have the flexibility to choose the best product or solution to meet his specific needs. To ensure that this flexibility of choice remains available to the consumer and endures through rapid technological changes, the most effective and sustainable way is to have free competition in the market between different software providers, international and domestic.

Governments should therefore be prudent in considering any policy that creates a preference in the marketplace of one software model over another. In practice, such a preference policy interferes with free competition in the market without necessarily bringing about the benefits that may be expected, e.g. cost savings, avoidance of vendor dependence[1] or advantages for the domestic software industry[2]. From the consumer perspective, having a preference policy artificially limits the choice of software that can best meet a customer's needs in a cost-effective manner. Preference policies prevent software providers from competing on equal terms. Solutions will not be selected based on whether the product has the best functionality or value for

[1] The dependence on a single vendor is not unique to either software model, or avoided by switching to the other model. Switching from one open source implementation to another open source variant involves similar types of costs and time as switching between different commercial software, or from commercial software to open source software. Users face the same deterrent effect in switching, due to the inertia of the change itself, and not the software platform being adopted.
[2] A preference for domestic software companies over international software can lead the domestic companies to become complacent and less incentivised to innovate, and eventually not be sufficiently prepared to be internationally competitive.

## Benefits of a Freely Competitive Market

By not having a preference policy, free competition in the market is maintained. By allowing market forces to freely operate, the level playing field that results will benefit the industry and consumers in the long term. For instance:

- software developers (both domestic and international) are motivated to compete at delivering maximum value, rather than be tempted to or rely on or fear preferential treatment on the basis of the software development model that they choose to adopt;
- software developers can freely innovate without being preemptively constrained in the way they can commercialize or license their innovations in the future;
- domestic software vendors can operate within a domestic marketplace that is realistic, allowing them to better prepare themselves for competition in the global marketplace rather than grow complacent in a sheltered marketplace;
- customers can have the freedom to choose the software option that gives them the best value that they can afford; and
- end users can trust that the products that they are required to use have been chosen based on utility of products and not based on politics.

the customer, but on factors such as the software model or the origin of software that may not have any intrinsic implication on the quality, value or utility of the software. It is counter-productive for a governmental policy to explicitly stifle competition if the government's end goal is to enhance access to the best technology by promoting competition and choice.

**Public administrations should fully preserve their ability to choose their software solutions, like any other product, based on the merits.**

Instead, public administrations should fully preserve their ability to choose their software solutions, like any other product, based on the merits in terms of functionality, performance, interoperability, security, value and cost of ownership in relation to other software solutions available in the market. An organization procuring software should state in clear and objective terms the functionality and requirements that it needs fulfilled, and allow all vendors, including both open source and commercial software vendors, to submit their proposals to the organization for consideration. The specifications should contain criteria such as the functionality, security requirements and performance characteristics that the user needs, rather than stipulate the name of specific

products or how the software should be developed or licensed. Each instance of procurement should be evaluated on its own, taking into total consideration the specific needs, requirements and environment where the software is to be used.

Fair and open competition, not Government-mandated preferences, should determine which products earn the confidence of consumers, including Government entities. Preference policies will stifle innovation that is essential to ensuring the growth of the software industry.

## Interoperability and Technology Standards

The need to promote interoperability is often cited as a motivation behind efforts to promote a particular software development model. Rather than having a preference policy, taking a long term perspective, a more effective approach to achieving interoperability is to develop a good understanding of technology standards, and have a suitable strategy to adopt interoperable standards. This will better achieve the desired policy objectives.

**Technology standards play an important role in fostering healthy competition in hardware and software solutions.**

Technology standards play an important role in fostering healthy competition in hardware and software solutions. They facilitate interoperability to provide a customer with the ability and flexibility to choose from a range of innovative software products to meet his need. Where customized products are acquired, it is similarly important for such products to be designed to be interoperable with other existing solutions or future additions so that they do not quickly become obsolete. Standards are particularly important for the public sector due to the need for better communication between government and citizens, and between government agencies (intra and inter-governmental). Standards also allow archival and legacy system problems to be better addressed by providing continuity and minimize the risk of fragmentation of the market into technological solutions that cannot work together.

Technology standards are typically documented in written specifications that enable developers of software, hardware and services to make and distribute products or components that work with one another within a given context. This interoperability can take the form of information exchange (e.g. protocols or file formats), task performance (application programming interfaces – APIs) and other functions that allow systems and people to collaborate effectively. Whole products are generally not designated as standards, but instead, the interfaces and functions of a product, e.g. the way it reads and write data, or the steps it takes to perform certain operations, are what may be designated as standards. Based on the standards, different suppliers can develop their own implementations of a standard, thus giving consumers a choice.

Technology standards are important to the industry as they typically solve problems that cut across the industry and are beyond the ability of a single vendor to address. Different vendors work together to create standards that can solve the

problem. For example, if each digital camera equipment vendor adopts its own data format for storage, a software vendor creating image editing software will need to cater for each variant of the data format of the hardware. The standardization of formats in such a scenario not only makes it easier for the software vendor in having to deal with multiple formats, but also for hardware vendors to introduce new standards-compliant equipment without the need to separately work with the software vendor to ensure that the new equipment is supported by the software. Similarly, alternative software solutions can be introduced without the new software vendor having to work with individual hardware vendors. Standards-compliant domestic products also find themselves more readily accepted by the global market in areas where established standards are already in use. Good standards are neutral and serve the needs of both small and large companies, as well as foreign and domestic companies.

### Voluntary Approach

Voluntary processes have proven to be the most effective means of fueling innovation through standards. The marketplace, responding to customer demands, is in the best position to determine the appropriate timing for the development and promotion of a standard. It allows suppliers to quickly respond to industry and customer needs by developing standards that most effectively address the interoperability issues.

Government-mandated standards in the technology industry can potentially have unintended consequences. Such mandated standards may unnecessarily freeze the development of new technologies and constrain the market's ability to reap the benefits of quickly evolving technologies. Inappropriately mandated standards may also disadvantage certain players competing in the market rather than create a level playing field. Premature adoption of standards that are not yet ready may create obstacles and hindrances to the market acceptance and penetration of the standards, and may preclude a multi-faceted competitive environment from being created. It is recognized however that there are limited situations where standards may need to be mandated in the public interest, e.g. with respect to technology standards as they relate to public health and safety issues (e.g. aviation, medical equipment and cellular emission).

The success of a standard is measured by whether it ultimately solves the problem for which it is intended[3]. A standard may be developed and evolved through a variety of dynamic processes that are voluntary and responsive to market demands. The method of development is not the critical factor that determines a standard's success.

## Open Standards

There are different forms of standards under the umbrella of technology standards, e.g. de facto standards, de jure standards, product standards, proprietary standards, etc. "Open standards" are one type of technology standards that have garnered interest in relation to achieving widespread interoperability. Governments can play an important role in advancing open

[3] For instance, a standard for document viewing may not be ideal in the scenario where document editing is required. However, the success of the document viewing standard should be measured by how well the standard allows documents to be viewed and not how well it allows editing.

## Characteristics of Open Standards

While there is no universally accepted definition of that term, all open standards have the following common characteristics:

- Open standards are published without restriction (e.g. potential implementers are not restricted from accessing the standard) in electronic or tangible form, and in sufficient detail to enable a complete understanding of the standard's scope and purpose;
- Open standards are publicly available without cost or for a reasonable fee for adoption and implementation by any interested party;
- Where there are any patent rights necessary to implement open standards, such rights are made available by those developing the specification to all implementers on reasonable and non-discriminatory (RAND) terms, either with or without the payment of a reasonable royalty fee; and
- Open standards are regularly developed, maintained, approved or ratified by consensus, in a market driven standards-setting organization that is open to all interested and qualified participants. Standards can also develop by consensus in the marketplace.

standards. Government policies that support the implementation or adoption of open standards where open standards exist and are broadly supported by the industry will improve interoperability and benefit Governments and consumers on the whole. On the other hand, Governments should avoid policies that inadvertently discourage the development and adoption of broad-based standards, either by mandating standards or reducing the economic incentive for the industry to participate in the standards process. Encouraging domestic software products to be developed in accordance with internationally accepted open standards can also make such products more readily accepted in the global marketplace.

**"Open standards" are one type of technology standards that have garnered interest in relation to achieving widespread interoperability.**

### Open Source Distinguished from Open Standards

It should be noted, however, that open standards are not synonymous with open source software, and do not exist only by virtue of open source software. While an open standard is a technical specification (i.e. a written description), open source software is software that may be used to implement an open standard in a particular product or service. Whether a standard qualifies as "open" has nothing to do with the development and licensing model of the software used to implement that standard.

Open standards are neutral with regards to the software model – it is equally feasible

for an open standard to be implemented in open source or commercial software. Software developers writing code individually choose and decide how they code the internals of the software. Thus, either open source or commercial software may well contain elements that are not based on open standards. Once defined, open standards are available to any software developer, and they do not require open source software, or any other form of software, for their adoption or use.

Some open source projects are closely associated with particular open standards and some standards have chosen to release their reference implementation under open source licenses. However, the mere availability of the source code is neither necessary nor sufficient to make something a standard, much less an open standard.

Software industry players, regardless of the software licensing or development model adopted, recognize the need for interoperability and are already working together to define such standards. A mature and balanced understanding of the purpose and internationally-accepted practice of standards setting is essential for a dynamic marketplace and technology industry. A healthy IT ecosystem based on voluntary standards has proven best to help customers achieve their desired goals of interoperability, flexibility and accessibility. The role of Governments should be to encourage and facilitate such standards initiatives and adoption, and raise the general awareness and incentive for domestic software industry to participate and collaborate with international players in this regard.

## Driving Software Innovation

Effective adoption of standards at the lower basic functional level will bring about greater competition and innovation at the higher application levels. Standards provide the technical mechanisms to create greater portability, scalability, stability and compatibility. Vigorous competition among different but interoperable technological products will allow customers to exercise free choice among innovative products to select the solution that best serves their needs. The undue preference for one particular product, platform or software licensing or development model not based on objective criteria such as open standards inevitably disturbs the competitive forces that can bring about the best results for consumers.

**Effective adoption of standards at the lower basic functional level will bring about greater competition and innovation at the higher application levels.**

The rapid advancement of computing technology in recent years has prompted the software industry to rise to the occasion and create better solutions, bringing about greater benefits to the community of consumers. Software must continually innovate and improve so as to remain relevant. Market based competition is ultimately the critical driving force in fostering greater software innovation that is relevant for users and helping to develop a vibrant software and technology industry.

The Business Software Alliance (www.bsa.org) is the foremost organization dedicated to promoting a safe and legal digital world. BSA is the voice of the world's commercial software industry and its hardware partners before governments and in the international marketplace. Its members represent one of the fastest growing industries in the world. BSA programs foster technology innovation through education and policy initiatives that promote copyright protection, cyber security, trade and e-commerce. BSA members include Adobe, Apple, Autodesk, Avid, Bentley Systems, Borland, Cadence Design Systems, Cisco Systems, CNC Software/Mastercam, Dell, Entrust, HP, IBM, Intel, Internet Security Systems, McAfee, Microsoft, Minitab, PTC, RSA Security, SAP, SolidWorks, Sybase, Symantec, Synopsys, The MathWorks, Trend Micro and UGS.

*Translations of this document is available on request.*

*Prepared by:*
Mr. **GOH Seow Hiong** (shgoh@bsa.org), Director of Software Policy for Asia of BSA, September 2005. 


Business Software Alliance
1150 18th Street, NW, Suite 700
Washington, DC 20036
United States of America
Tel: +1.202.872.5500
Fax: +1.202.872.5501
www.bsa.org/usa/policy

BSA Europe / Middle East / Africa
79 Knightsbridge
London, SWIX 7RB
England, United Kingdom
Tel; +44 (0) 20.7245.0304
Fax: +44 (0) 20.7245.0310
www.bsa.org/eupolicy

BSA Asia
300 Beach Road
# 25-08 The Concourse
Singapore 199555
Tel: +65.6292.2072
Fax: +65.6292.6369
www.bsa.org/asia-eng/policy