Illinois Public Law and Legal Theory Research Papers Series

Research Paper No. 04-07
March 18, 2004

Nurturing Software: How Societal Institutions Shape the Development of Software

Rajiv C. Shah*
Jay P. Kesan**

*University of Illinois, Institute of Communications Research (ICR)
**Associate Professor, University of Illinois College of Law

Abstract

It is widely recognized that software affects fundamental societal concerns, such as privacy. Software does not just appear, but is produced within a variety of societal institutions. This article analyzes how societal institutions shape the development of software and its resulting implications for society. Specifically, we consider how institutional rules are evident in the different structures, motivations, and influences of four societal institutions. We begin by discussing universities and continue on to firms, consortia, and the open source movement. Once we understand how all of these factors operate, we can offer predictions on the resulting attributes of software. In the final section, we show how these institutional factors explain the variation in the development of web browsers as well as in the incorporation of the societal value of security into software.

Nurturing Software: How Societal Institutions Shape the Development of Software

Software doesn't grow on trees. Instead software is nurtured and cultivated in several different environments from universities to firms. In each environment software is nurtured differently. In turn, this influences the final features of software. This simple observation has important implications not only for the software development, but also for society. After all, as Lessig and others have argued, software is the law of cyberspace and affects fundamental societal issues such as security, privacy, trust, and accessibility [6].

Scholars studying software have emphasized that software can be designed differently. Each of these designs contain biases that favor certain actors or social values [4, 12]. For example, Internet search engines do not provide results in an unbiased and neutral fashion. Instead, they systematically favor certain types of sites over others [5, 7]. Similarly, there is concern that the design of digital rights management software may inadequately consider values of fair use and privacy [1, 10].

We argue that an important source of values in software is the institution in which it is developed. An institution is composed of a group of actors who are subject to a system of rules that structures their activities. These rules concern goals, rights, procedures, social norms, and formal legal rules that attenuate individual preferences. We examine four environments or societal institutions where most software is developed. They are universities, firms, consortia, and the open source movement. While there is a great degree of variation within each of these categories, social theorists recognize these institutions as distinct sources of software.

Understanding the development of software has larger implications than revealing the differing working conditions for software engineers. Increasingly, society seeks to shape

software development to address its concerns. This is evident in governmental policies from funding research into cybersecurity to regulations regarding accessibility features for software. Through these policies, government is attempting to shape the development of software in various societal institutions.

To understand how these institutions nurture software differently we utilized a series of historical case studies that included NCSA Mosaic, Netscape's cookies, the Platform for Internet Content Selection (PICS), and the Apache web server. The criteria for these case studies were that the software must be significantly developed within the societal institution of interest, and involve an interaction with substantial public policy issues. After all, if there was no significant public policy issue, then society has little interest in understanding software development. The data for the case studies was built on documentary sources and interviews.

This article analyzes how societal institutions shape software and its resulting implications for society. Specifically, we consider how institutional rules are evident in the different structures, motivations, and influences of four societal institutions. We begin by discussing universities and continue on to firms, consortia, and the open source movement. Once we understand how all of these factors operate, we can offer predictions on the resulting attributes of software. In the final section, we show how these institutional factors explain the variation in the development of web browsers. A second example considers how institutional factors affect the incorporation of the societal value of security into software. We discuss the current state of security as well as how it could be improved.

Universities

Universities are institutions whose purpose is to expand the frontiers of knowledge. Many significant computing technologies have emerged from universities including the Internet, artificial intelligence, and computer graphics. Society funds universities because private firms will under invest in basic research [9]. Universities now account for about half of all basic research spending within the United States and are also the genesis of many technology firms [8].

Developers in universities are primarily motivated to enhance their reputation in the scientific community and not purely on economic gain [2]. Reputation is derived from being the first to discover or develop innovative findings as measured through peer recognition. Consequently, software developed within universities is institutionally biased towards matters that are regarded as important by a researcher's peers. This leads to a secondary regard for potential economic gain when developing software within a university.

To foster the development of innovative software, universities provide their developers with a measure of autonomy. Universities have recognized that the freedom to pursue self-directed research can lead to new knowledge and the development of innovative software. For example, our research found that two key developments for the World Wide Web ("web") were the result of autonomy within university-style environments. They include the original conception of the web by Tim Berners-Lee at the European particle physics laboratory, Conseil Europeen pour la Recherche Nucleaire (CERN). Similarly, Marc Andreessen initially developed NCSA Mosaic, the first popular web browser, at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. Andreessen has said

that he believes the unstructured work environment at NCSA supported the development of innovative ideas and software.

The scarcity of resources within universities is an important economic influence affecting software development. Universities simply cannot fully fund all their ongoing research. Developers cannot depend upon a large technical support staff, and functions seen as extras or "bells and whistles," such as technical support and documentation, are not fully supported. Consequently, this provides researchers the impetus to seek resources, such as research grants, outside the university. This can also lead developers to focus on developing the standards and building blocks for others to build upon. Berners-Lee used this strategy during the development of the web. He developed the standards and reusable building blocks of software that became the basis for future web browsers and servers.

Firms

In our economic system, it is the private sector that develops most software. In 1999, firms spent six times more than government on basic research, applied research, and development activity [8]. As a result, firms such as IBM and Microsoft have historically developed much of the software widely adopted in society.

The motivation of a firm is straightforward. Firms are driven by profit. To achieve profits, firms must provide goods and services that meet consumer demand. Successful firms listen to their customers, provide them with the services that they need and will need, and provide support when they run into trouble [11]. As a result, economic concerns shape the development of software created by firms. Consequently, values that are assumed to be unprofitable, are not factored into a firm's decision-making process, even if these values are

important to society. For example, consider the cookies technology, which allows web sites to save state and therefore maintain information on their users. When Netscape implemented this technology, we found that Netscape did not spend its resources developing software that would minimize the privacy concerns posed by the cookies technology since this action could be viewed as not being profitable. This explains why early versions contained no cookie management tools or even documentation about cookies. This neglect of unprofitable societal concerns by firms is understandable and is further discussed in the implications section.

Consortia

The production of software is not done entirely by firms or by the government to the exclusion of the other. Often, they cooperate. An important form of cooperation is the consortium. Two prominent consortia for the Internet are the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF). They both develop standards that are useful to all their members by relying on cooperation between competing firms.

The primary influence on the development of standards within a consortium is its members. It is the members who decide what projects to pursue and the appropriate level of resources. The members also affect the development process through their choice of a consortium's structure. This involves decisions on membership composition and membership rights, intellectual property rights, and the procedural rules that govern their work. Additionally, these structures are not fixed but can change over the lifetime of a consortium. For example, the W3C was established with the intent of creating a faster standards process than the IETF. As the W3C has aged, it has added formal procedures that have slowed down its development process.

Consortia may ignore or overlook outside social influences or third parties during the development process. This is important because, at first glance, consortia often appear to be working for the benefit of the public as a whole. But because consortia are accountable only to their members, they do not adequately consider the needs of third parties, such as independent software vendors and end users. This can result in ineffective or technically poor solutions. For example, our research found the W3C developed PICS largely in response to political and legal pressures placed upon its members. As a result, they did not consider the needs of end users, such as parents, or firms that sell filtering software. This occurred because the end users and the commercial filtering firms were not part of the development process. As a result, PICS is of little use to parents and firms selling filtering software.

Open Source Movement

The open source movement is an institution that stands apart from universities, firms, and consortia. It has developed software that rivals commercially available software, including Apache, the Linux operating system, the scripting language PERL, and the popular email server Sendmail. The defining characteristic of the open source movement is an overriding norm that source code should be made available to the public [3]. By keeping the source code publicly available, developers can build upon others' earlier work to create more refined and complex software.

The motivations of the open source movement are varied. There are a small number of paid participants as well as private firms. These entities, such as IBM, have a direct financial motivation in the development of open source software. However, the motivations for the vast majority of unpaid participants vary. They range from being focused on software development

for creative satisfaction, utilitarian enhancement including a rise in reputation, or a political motivation that sees open source as superior to proprietary software.

The open source movement's development process is primarily influenced by its membership of volunteer developers. Because they can only provide limited time and resources, the limits of volunteerism shape the development of software. In contrast to a firm, there is no pressure to force volunteers to work on a particular project in a timely manner. Consequently, it is the volunteers who decide what software will be written and on what time schedule. They also wish to work on interesting tasks. The resulting software is then biased towards the interests of the volunteers, who are sophisticated developers and not ordinary users. This leads to the projects that developers think are interesting or useful, such as desktop environments and text editors. As a result, volunteer members may not necessarily work on software that is in greater demand or more socially beneficial.

The emphasis on personal motivation attenuates external influences on open source software. An international team of volunteer members leads the open source movement. This diverse set of developers is focused on developing what is interesting to them and is not driven entirely by external political or economic concerns. For example, Mozilla, an open source web browser based on Netscape's web browser, contains features to block images from third party web sites and pop-up advertising windows. Moreover, at times, the open source movement can be defiant to conventional economic and political influences. For example, consider the refinement of the Gnutella file-sharing program. The open source movement improved Gnutella largely because its decentralized design was intended to prevent users from being blocked access to the file-sharing network and also to avoid vicarious copyright liability concerns after Napster.

The results of our findings are summarized on institutional features and the influences on the development process are represented in Table 1 and Table 2, respectively. The results shown in the tables are from our larger body of research, only some of which is discussed here.

Table 1. Institutional Features

Institution (Case Study)	Institutional Focus	Developers' Motivations	Structural Features	Perceived End User
University (NCSA Mosaic)	Create and disseminate knowledge	Reputation	Scarcity of resources	Public
Firm (Cookies)	Profit	Financial compensation	Adequate resources	Customers
Consortia (PICS)	Cooperatively develop products	Varies, developers may be employees or volunteers	Many variations in structure, such as membership composition and rights	Varies, they may restrict to their members or may make public
Open Source Movement (Apache)	Create software with publicly available source code	Varies, developers may be employees or volunteers	Decentralized development process	Public

Table 2. Influences on the Developmental Process for Software

	Social & Political Influences	Economic Influences
University	Developers are provided autonomy and strive for peer recognition.	Striving for grants and creating "building blocks"
Firm	Influential, if they are likely to lead to higher costs	Strong emphasis on anticipating consumer demand
Consortia	Members are the primary influence, but this may lead to overlooking third parties.	
Open Source Movement	Members are the primary influence. The "limits of volunteerism" and the needs of developers serve as powerful influences, while economic and political influences are slight.	

Implications

Once we understand the institutional features and influences that shape software, we can explain certain attributes of software. A schematic of this process is shown in Figure 1. The attributes affected could include technical issues, such as the use of open standards, or business issues, such as providing technical support, or social characteristics such as the concern for

privacy. We use the term shape instead of cause because all of these institutional factors do not cause specific features in a simple one-to-one fashion. Instead, software development is more complex and nuanced. Our analysis seeks to indicate salient institutional factors that can influence or shape the development of certain attributes. Our findings of how institutional features and influences affect the technical and social attributes of software is shown in Table 3. We illustrate these implications with two examples. First, we briefly consider the difference in the development of web browsers. Second, we discuss how institutions differ over incorporating societal concerns into software by considering security.

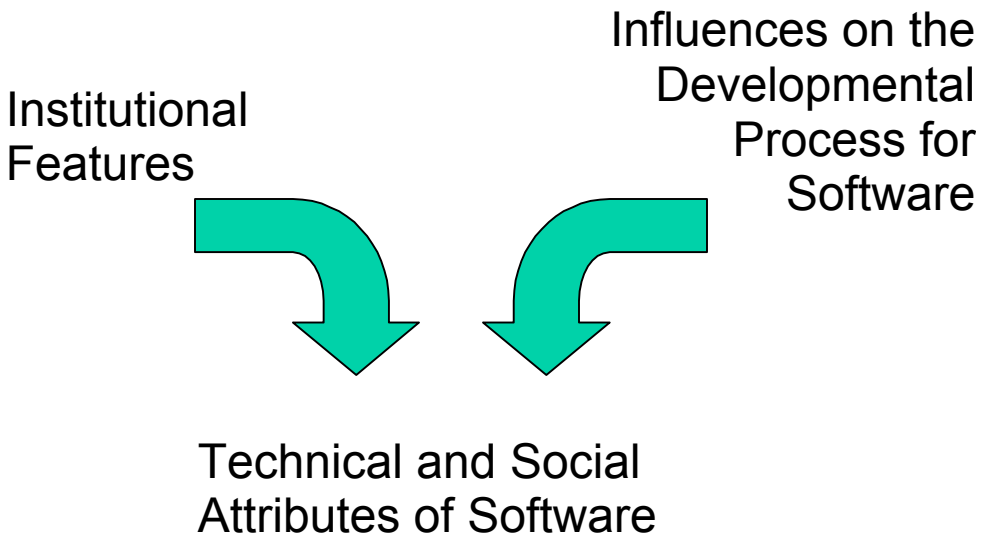

Figure 1. Schematic Representation of Software Development

Table 3. Technical and Social Attributes of Software

	IP Protection	Open Standards	Quality of Software	User-friendly	Marketing & Technical Support	Incorporation of Societal values
University	Not typically	Favors open standards	Not emphasized	Not emphasized	Limited due to scarce resources	Autonomy results in software with a wide variety of values
Firm	Typically, to restrict access to customers	Varies, it is a strategic decision	Generally high quality	Emphasized	Emphasized to create, develop, and retain customers	Only profitable values are incorporated
Consortia	Varies, but W3C and IETF make their standards freely available	Varies	Varies, depending upon input from members and the public	Varies	Varies from little marketing to those that promote and certify standards	Varies, but it may address societal issues in ways that benefit the members and not the public
Open Source Movement	Used to ensure availability of source code	Favors open standards	Varies, but can produce high quality software	Not emphasized, and often software is for sophisticated developers	Not emphasized	Variety of influences, but limited by the members' concern

Web Browsers. A cursory examination of web browsers shows how institutional factors affect software development. Web browsers created within universities, such as NCSA Mosaic, tend to focus on innovative features. In the case of NCSA Mosaic, this browser was a significant advance over other web browsers because of its ease of use and the capability to view images in web pages. While firms focus on features that contribute to profits. In the case of the Netscape web browser, this led to an emphasis on improving security, commerce, and performance, and resulted in new technologies such as cookies, continuous document streaming, and Secure Sockets Layer. In contrast, consortia focus on their members needs. The W3C's web browser, Amaya, is not designed to compete with commercial web browsers, but to test new technologies produced by the W3C and their members. Finally, the open source movement has developed a wide variety of web browsers including Mozilla and Konqueror. Their accomplishment has not been innovation, but instead continual incremental refining and the inclusion of features that

commercial web browsers were hesitant to incorporate, such as pop-up ad blocking and cookie management tools.

Societal Concerns. Institutions differ in their incorporation of societal concerns into software. Examples of societal concerns include features such as privacy, protecting minors from inappropriate content, and protecting intellectual property rights. If these concerns are not adequately expressed in software, society may use actions such as government regulation to force the development of software. As an illustration of the different inclinations of institutions consider the incorporation of security.

Universities have traditionally conducted research into a variety of issues including cybersecurity issues. Security is but one of many sub-fields of computer science. Recently, considerable interest has been shown by government and firms in security research. This is being accomplished by increasing funding for research as well as increased demand for graduates well versed in cybersecurity.

Firms generally do not produce software that supports unprofitable, but socially beneficial, values. This is because firms seek to meet the needs of their customers and not society in general. Historically, we see a decreased emphasis on security in many software products produced by firms. This was because firms believed that consumers were seeking speed, convenience, and new features over security. These attitudes can be explained by consumers not fully understanding the costs of security and not having to fully bear the costs of poor security. If consumers become more knowledgeable about security issues and choose to collectively bear greater responsibility for security, this could prompt firms to incorporate security.

Consortia differ in their willingness to develop standards that address societal values. In the case of security, there are several consortia that are actively developing new security standards. However, the final standard that is adopted is only approved by the consortia's members and necessarily excludes certain third parties who may be affected by the standards. As a result, the standards developed by consortia may be ineffective or not widely adopted. This is what happened with PICS.

Finally, the open source movement is subject to a variety of influences and can incorporate a wide range of social values including security. The development of secure software is fundamentally affected by whether it is a motivating concern for open source developers. If developers care about security then it will be reflected in the final software. Another factor that results in improved security is the widespread dissemination of source code, which allows for peer review of the software. Thus far, there is a wide variation in security among open source projects, with some projects being heavily scrutinized and improved upon while others languish with security flaws.

Conclusion

The nurturing or the development of software is not universally uniform. Instead, software engineers labor within institutions. Our research found that societal institutions have differing structures, motivations, and influences that affect software development. This led us to identify and explain salient institutional factors for the development process. Because of these varying institutional factors, software with the same technical functionality may have very different features, characteristics, and values. This was illustrated by showing how universities, firms, consortia, and the open source movement each have shaped the development of web

browsers in different ways. We also showed how institutional factors can explain the current state of security for software as well as how it can be improved in the future. While institutional factors cannot explain all features of software, they provide a starting point for understanding how software develops. Our hope is that this analysis not only describes the nurturing of software, it also may inform how society could intervene to develop better software.

References

1. Camp, L. DRM doesn't really mean copyright, *IEEE Internet Computing 7*, 3 (2003), 59-65.

2. Dasgupta, P. and David, P. A. Toward a new economics of science. *Research Policy 23* (1994), 487-521.

3. DiBona, C., Ockman, S., & Stone M. *Open sources: Voices from the open source revolution*. O'Reilly, Cambridge, 1999.

4. Friedman, B. and Nissenbaum, H. Bias in computer systems. *ACM Transactions on Information Systems 14,* 3 (1996), 330-347.

5. Introna, L. and Nissenbaum, H. Defining the web: The politics of search engines. *IEEE Computer 33*, 1 (2000), 54-62.

6. Lessig, L. *Code and other laws of cyberspace.* Basic Books, New York 1999.

7. Mowshowitz, A. and Kawaguchi, A. Bias on the web. *Communications of the ACM 45*, 9 (2002), 56-60.

8. National Science Board, *Science and engineering indicators - 2002*. National Science Foundation, Arlington, Va. 2002.

9. Nelson, R. The simple economics of basic scientific research. *Journal of Political Economy 67*, 3 (1959), 297-306.

10. Samuelson, P. Encoding the law into digital libraries. *Communications of the ACM 41*, 4 (1998), 13-18.

11. Shapiro, C. L. and Varian H. R. *Information rules*. Harvard University Press, Cambridge, 1998.

12. Winner, L. Do artifacts have politics? *Daedalus 110*, 4 (Winter 1980), 121-136.