Illinois Public Law and Legal Theory Research Papers Series

Research Paper No. 03-18
December 18, 2003

Manipulating the Governance Characteristics of Code

Rajiv C. Shah*
Jay P. Kesan**

*Doctoral Candidate at the Institute of Communications Research, University of Illinois
** Associate Professor, University of Illinois College of Law and the Institute of Government and Public Affairs, University of Illinois

Manipulating the governance characteristics of code

Rajiv C. Shah and Jay P. Kesan

Rajiv C. Shah is a Doctoral Candidate at the Institute of Communications Research, University of Illinois at Urbana-Champaign, 228 Gregory Hall, 8105, Wright Street, Urbana, Illinois, USA. Tel: +1 217 444 2549; E-mail:r-shah4@uiuc.edu
Jay P. Kesan is an Associate Professor at the College of Law and the Institute of Government and Public Affairs, University of Illinois at Urbana-Champaign, 504 East Pennsylvannia Avenue, Champaign, Illinois, USA. Tel: +1 217 333 7887; E-mail: kesan@law.muc.edu

Keywords *Information, Communication processes, Technology led strategy, Regulation, Standards, Modulators*

Abstract *Regulation through "code," i.e. the hardware and software of communication technologies, is growing in importance. Policymakers are addressing societal concerns such as privacy, freedom of speech, and intellectual property protection with code-based solutions. While scholars have noted the role of code, there is little analysis of the various features or characteristics of code that have significance in regulating behavior. This paper examines three universal governance characteristics that policymakers may use to ensure code comports with societal concerns. The characteristics are transparency, defaults, and standards. For each characteristic, the paper discusses the salient regulatory issues for manipulating code. Additionally, the paper provides normative proposals for modifying some characteristics, such as defaults. In sum, our analysis should aid policymakers seeking to manipulate code to ensure that code comports with our societal values and addresses our societal concerns.*

I. Introduction

One of the most significant advancements in telecommunications policy is the recognition that the architecture of information technologies can be used to regulate behavior in a positive manner. Policymakers and regulators are using or advocating architectural solutions to address societal concerns with crime, competition, security, free speech, privacy, the protection of intellectual property, and revitalizing democratic discourse (Katyal, 2001; Lemley and Lessig, 2001; President's Critical Infrastructure Protection Board, 2002; Lessig and Resnick, 1999; Pollack, 2001; Burk and Cohen, 2001; Wilhelm, 2000). However, there is little theoretical or empirical knowledge driving these decisions. In effect, regulators are "shooting from the hip" when employing architectural solutions. This paper examines three governance characteristics of architectural solutions using information technologies. These characteristics are analogous to "knobs and levers" that policymakers can manipulate to regulate behavior.

Our work begins with the recognition that while legal scholars have long theorized and analyzed the effect of law on society, the study of the effects of architecture is noticeably immature. The most significant work in this area is by Katyal (2002), who focuses on how architecture can reduce crime. With the exception of a few scholars, such as Winner (1977) and Sclove (1995), most theorists do not consider how regulators can manipulate architecture.

This paper was presented at the Telecommunications Policy Research Conference, Washington DC, 29 September 2002. This material is based on work supported by the National Science Foundation under Grant No. ITR-0081426. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Instead, their efforts are concentrated on ensuring that the significance of architecture is recognized. As a result, theorists from disciplines including sociology, geography, and architecture provide little insight for regulators seeking to employ architectural solutions to online problems (Foucault, 1979; Lefebvre, 1991; Markus, 1993).

In the realm of information technologies, scholars have recognized the power of architecture or code to regulate behavior (Katsh, 1996; Lessig, 1999; Reidenberg, 1998). The term "code," as we use it, refers to the physical hardware and software of information technologies. In *City of Bits*, Mitchell (1996) described how code can control our behavior. Just as a building can be constructed with walls of brick or glass, code can create analogous walls in cyberspace. The design of code may either facilitate tracking and surveillance, or it may aid in preserving anonymity. In this way, the architecture of code may have a profound influence on behavior.

The power of code is that it affects not only our communication experience, but also affects fundamental societal values such as freedom of speech. For example, code in the form of filtering software is often used to ensure minors do not access inappropriate material. However, Weinberg (1997) points out that the imprecise nature of filtering software limits the availability of meaningful material concerning AIDS and breast cancer. Shapiro (1999) argues that the current design of the Internet without any type of public space hampers freedom of speech. Without public spaces, it becomes difficult for speakers to reach the ears of the people. Another code related restraint on the freedom of speech concerns the quality of traffic on the Internet and the rush for content producers to utilize high-speed overlay networks such as Akamai. The use of this code results in commercial content that appears much quicker and richer than content that cannot afford to pay for overlay networks. These are all examples of how code affects freedom of speech. Similarly, code affects other values and rights such as privacy, security, trust, innovation, and the protection of intellectual property.

There are several reasons code is of heightened importance as a regulatory mechanism. First, consider its growing pervasiveness. We are increasingly spending more and more time in code-based worlds. Second, code is malleable. The architecture of code is entirely man-made and is not subject to the same physical laws as the built environment. Software designers can shape code in a variety of ways, such as the creation of virtual worlds. The malleability of code allows society to address societal concerns with code (Shah and Kesan, 2003). Already, code-based solutions are being considered for issues such as privacy and free speech. Third, the malleability of code declines over time and our choices become entrenched. This leads to much higher costs in modifying code. For example, the transition to digital television broadcasting has been a costly and complex transition, because of the entrenched nature of the standards and infrastructure for television.

To analyze how code regulates behavior, we chose to use a series of case studies. The case studies go beyond the anecdotal accounts of code in the current literature. Anecdotal accounts are useful to grasp the basic concept of code as a regulatory mechanism. However, they are not capable of providing a systematic understanding of how code regulates. To understand myriad ways code regulates, the case studies took into account the early Internet as well as contemporary examples of code. Moreover, the case studies touched on societal values such as free speech and privacy. The case studies are: the finger command, the cookie specification, and the Platform for Internet Content Selection (PICS).

This paper is organized as follows. Part II provides a short background on the three case studies. Part III discusses three governance characteristics of codetransparency, defaults, and standards. These characteristics have implications for regulators who are seeking to employ code as an architectural method for regulating behavior. This part discusses how regulators can understand and manipulate these characteristics to further the goal of ensuring that code comports with societal interests.

II. Summaries of the case studies

This part provides a short background into the three case studies. The first case study examines the finger protocol, which allows users to find information about other users on the network. The second case study examines cookies developed by Netscape, which allows Web sites to maintain information on their visitors. The third case study examines the PICS, which was developed by the World Wide Web Consortium (W3C). PICS is a standard for labeling Web pages, primarily to prevent minors from accessing inappropriate content.

A. Finger command

The finger command was an early computer network application that allowed people to see who else was using any specific computer system, as well as providing basic information on that person. The finger command was the source of one of the Internet's first flame wars (Hafner and Lyon, 1996). Although this debate over online privacy occurred over 20 years ago, it is still very relevant today. This debate began when privacy bits were added to the finger command at Carnegie Mellon University (CMU). The privacy bits allowed a user to turn off information about their behavior. This information at issue was whether the user is currently logged on, when the user had logged off, whether there was any mail in her mailbox, when the user has last

read her mail, and if there is mail, the most recent sender. The privacy bits were an option that allowed people to decide if they want this information revealed. Moreover, the privacy bits had a default setting to prevent this information from being released. Thus, to enable others to find out when you last logged on, a person had to proactively turn their privacy bit "on" to reveal this information. At CMU, the other information revealed in the finger command such as your office location and office number was left to the discretion of the user. Thus with the addition of the privacy bits, a user could now ensure that no information about them was revealed if someone "fingered" them. The addition of these privacy bits was controversial and debated both inside and outside of CMU.

B. Cookies technology

Cookies are one of several technologies Netscape developed in order to position its Web servers for e-commerce. The cookies technology was the most innovative feature and one that would forever alter the Web. In early Web browsers, the Internet was a stateless place. It had little regard for who you were, what product you are asking for, or how many purchases you have made. It had no memory. The lack of statelessness on the Web made commerce difficult. It was difficult for a Web site to remember multiple purchases or allow for an automatic one-click shopping feature that remembers your personal information. According to Lessig (quoted in Schwartz, 2001), "before cookies, the Web was essentially private. After cookies, the Web becomes a space capable of extraordinary monitoring".

Cookies were incorporated in the earliest version of Netscape's Web browser released in 1994. However, it was not until early 1996 that the public became aware of cookies. The *Financial Times* broke the story on 12 February 1996 with an article on cookies and privacy (Jackson, 1996). The article immediately drew attention to cookies and resulted in a great deal of uproar about the use of cookies. Over the next few years, cookies became one of the top Internet privacy issues.

Netscape's cookies led the Internet Engineering Task Force (IETF) to work on a standard for state management on the Internet. The IETF, as the *de facto* Internet standards body, sought to ensure there was a complete technical specification on state management. Eventually, they decided to use the Netscape's cookies specification as the basis for the IETF's standard. However, the standards process soon ran into problems, because Netscape's implementation of cookies was fraught with privacy and security problems (Kristol, 2001). Eventually, the IETF developed a standard (Kristol and Montulli, 2000). However, this standard has not been fully adopted by the leading developers of Web browsers such as Netscape and Internet Explorer.

C. PICS

The history of the PICS begins with proposed legislation to regulate indecent speech on the Internet by Senator Exon in the summer of 1994. Senator Exon reintroduced his legislation in February 1995. This would eventually become the Communications Decency Act (CDA). The law made it unlawful to transmit indecent material over the Internet to minors. In response, in June 1995, the W3C began setting up a meeting to discuss technical solutions for the regulation of Internet content. In August 1995, the W3C held a members meeting with two goals in mind. The first was to create a viewpoint-independent content labeling system. This would allow content to be labeled in many different ways. This labeling system, which went beyond movie ratings of content, was more general and encompassed other classification schemes such as the Library of Congress cataloging scheme. The second goal was to allow individuals to selectively access or block certain content.

The result was PICS (Resnick and Miller, 1996). This is a set of standards for labeling material on the Internet. The labels can be placed on material by self-labeling or by third parties. Generally, this system can be analogized to the V-chip, where programs are rated and the chip is able to block certain types of programs. The PICS concept goes beyond a simple rating system such as the movie rating vocabulary of PG, R, and X. The PICS specification allows customized ratings types as well as allowing any party to rate material. Thus, a Web page could be rated with many different types of labels for different purposes or different organizations. PICS does not dictate or establish a rating vocabulary nor does it identify who should pay attention to which labels. Instead, people are able to decide which rating systems, if any, they wish to conform to. Moreover, people can also rely on third party "labeling bureaus" for sites that are not labeled or merely to seek a third party rating. For example, an organization may set up a labeling bureau to identify and rate "hate" sites.

III. Manipulable governance characteristics

This section focuses on the three governance characteristics of code. Based on our case studies, we found these characteristics to have a significant regulatory influence in our case studies. We believe these characteristics are universal and have implications for regulators seeking to employ code. For each characteristic, we provide an overview, discuss public policy implications, and recommend how regulators should modify these characteristics. The first characteristic we discuss is transparency. This characteristic concerns whether users can understand how code operates. The other two characteristics are focused on the settings for code. The

second characteristic concerns the setting of defaults. The third characteristic that we consider is standards.

A. Transparency

The first governance characteristic is transparency. Transparency in code allows people to understand how code operates. They can make an informed decision about whether to use code and how best to employ code. For example, transparency is an important issue for technologies that affect privacy. This was evident in our case study on cookies. In the early versions of Netscape's browsers, users were provided no information or control over the cookies technology (Millett *et al.*, 2001). The resulting opaqueness of cookies did not allow people to understand how this technology operates. Because of the lack of information, users had to look elsewhere to understand this technology. In the case of cookies, this led to a great deal of misinformation over cookies. Cookies became the scapegoat for a variety of privacy and security issues.

Thus far, our research has revealed several regulatory aspects of transparency. First, transparency varies by audience. What is transparent for one group of people may be opaque to another group. An example of this is the reveal codes command in WordPerfect that shows how a document is formatted. This allows a person to quickly identify and rectify problems with formatting in a transparent manner. The complexities of formatting are made transparent for the average person. However, this command does not truly reveal how the formatting of a document is encased within the saved binary file. This command does not provide any transparency for a developer seeking to understand the structure of WordPerfect files.

An extreme level of transparency can be found in open source code or code placed into the public domain. This transparency allows anyone to inspect the code. It is then possible to see what the source code is capable of accomplishing and allows one to identify potential problems. In fact, the transparency of the open source movement's code is a major reason for the high quality of its code. Moreover, the transparency cultivates a level of trust in the code among users, because they can fully examine the code.

Regulators in evaluating code must consider this variation in transparency. At an individual level, regulators are concerned with whether an average person can make an informed decision in employing code. For example, to ensure that people are adequately informed about cookies, a regulator may need the expertise of those practiced in the design of user interfaces for evaluating an effective cookie management tool. At a higher societal level, regulators are concerned about whether code properly addresses issues of societal concern. This may involve access to documentation on the development process or access to the source code. For example, regulators may seek to identify the rules for excluding sites in filtering software. Understanding the rules allows for transparency in filtering software[1].

Finally, transparency is dependent on the implementation of code. For code to be transparent, it must be simple for a person to understand and use the code. Making it simple depends on the developers of code. PICS is an example of code that is transparent because of its implementation. The implementation in Microsoft's Internet Explorer is intuitive and allows a person to quickly understand how PICS operates and what the possible limitations might be. In contrast, the cookies technology was implemented in an opaque manner. In the earliest versions of Netscape's browsers, cookies were implemented in a manner that users had no idea that cookies even existed. Even after this, Netscape provided little or no documentation on cookies. Even after cookies became widely recognized as a privacy issue, the implementation was far from transparent. It took several years before Microsoft's and Netscape's browsers provided users with information on cookies and control over cookies (Millett *et al.*, 2001).

B. Defaults

The second salient governance characteristic of code is defaults. A default for code is a preselected option adopted by the code when no alternative is specified by the user. Defaults are important from a public policy perspective because they provide a user with choices. For regulators, a default accords for multiple regulatory settings that can be chosen by the user. Thus by changing the default or adding a default setting, a regulator is changing the settings within the code.

For a default to exist there must be an alternative for a person to select. If there is no alternative, then it is not a default setting, it is a fixed setting. This was evident in our case study on cookies. It was not until the later versions that Netscape allowed users to manage the cookies technology. However, in all of these versions, the default setting was to accept cookies. One of the controversial aspects of the IETF's formal standard on cookies was that it required that the default for cookies be set to refuse unverifiable or third party cookies (Kristol and Montulli, 2000). These types of cookies have been criticized because users have no expectation that a third party Web site may be placing a cookie on their computer. Despite the IETF's standard, the latest versions of Web browsers by Netscape and Microsoft still accept third party cookies by default to satisfy the advertising management companies that rely on third party cookies.

Defaults are an essential part of code. A typical program has tens (and up to hundreds) of default values that are set by the developers. For example, there are many default settings in each of our case studies. However, they do not all

have public policy consequences. Moreover, society can easily insist that a developer create a default to give people choices, because code is malleable. For example, regulators could insist defaults be designed so people have to intervene to protect their privacy or they could insist that defaults be set to protect personal information.

The importance and relevance of defaults can be seen in the debate over the finger command at CMU. Much of the controversy over the privacy bits was focused on the default settings. The defaults could have been set to protect privacy or to reveal information. Some contested the setting of the defaults to favor individual privacy. The ensuing debate was about whether the code should favor individual rights or community rights. Each side wished to have the defaults set to support their values. The underlying premise in these arguments was that the default setting is very important. Therefore, people should have to undertake some action to forsake this default value. The finger case study illustrates the power of default settings.

This leads policymakers to ask, why do people follow default settings that are not in their best interest. The reason for this is two-fold. First, people do not change defaults because they are uninformed. A default setting is essentially useless if a person does not know about the possibility of changing the option or the ramifications of each choice. One of the motivations for the heated debate over the finger command was to raise awareness among users of the significance of the default settings.

The second reason people do not change defaults is their lack of technical sophistication. If people cannot figure out how to change a default, they cannot change the default. This can be remedied by changing the code. For example, to change the cookie settings in early Web browsers, users were forced to navigate through complex menus. It was not until the most recent browsers that an unsophisticated user could readily change their default settings for cookies. A second solution is to provide guidance, education, or documentation on how to change the default settings.

Determining the optimal setting of default values has been studied by behavioral economics (Sunstein, 2002). Much of this analysis has focused on defaults associated with law and social policy, specifically contracts, but can also be extended to how code governs. The starting point is the Coase theorem, which holds that a default rule does not matter if there are no transaction costs. This is because the parties will bargain to a common result that is efficient. Under this analysis, regulators do not need to be concerned with defaults in code, assuming there are no transaction costs. However, we know that there are transaction costs. This leads to the conclusion that regulators should set the default to minimize transaction costs.

Besides transaction costs, there are three other effects unrecognized by the Coase theorem that regulators need to consider. First, defaults often initially favor one party over another. This results in an endowment effect where people who initially were allocated the default value it more than if the default had been set to favor the other party. Consequently, the parties will not bargain to the same result, with different initial default allocations. This can be seen in the finger controversy at CMU. Undoubtedly, a different discussion and result would have occurred had the defaults been set to reveal information. Second, the initial allocation of defaults has another effect besides the endowment effect. Defaults have a legitimating effect, because they carry information about what most people are expected to do. This is often the case with code. People often assume that the default settings are ordinary or sensible practices. For example, the default setting of allowing cookies in Web browsers has resulted in people considering cookies to be an ordinary part of using the Web. Third, regulators can set defaults to operate as "penalty defaults" to ensure disclosure between the parties (Ayres and Gertner, 1989). Penalty defaults can require the parties to reveal information to each other or third parties. For example, a penalty default could be used to ensure that consumers are provided adequate information on all the potential uses of their medical information. All three of these effects are useful in determining whether a regulator should intervene to change a default as well as the optimal default values.

C. Standards

The third important regulatory characteristic for code is standards. Standards are considered to be a quantifiable metric used by a group of people for common interchange (Cargill, 1989). In relation to code, standards can be considered as the specification, schematic, or blue print for the parts of code that must interoperate or interconnect with other code. For example, in order to transmit e-mail between different computers running different software, there is a need for standards that specify the format for the transmission of e-mail messages. A related code characteristic is modularity, which can be considered as a higher form of a standard. Modularity breaks down a large piece of code into smaller pieces or modules.

Open standards are of interest to us. Open standards can promote competition and consumer choice by providing for more than one vendor for any product. Furthermore, consumers can be confident that the solution they purchase will be compatible with products from other vendors. Similarly, modularity provides for flexibility, competition, and choice by allowing multiple vendors to develop parts of code. From a regulatory standpoint, this flexibility allows consumers to choose the appropriate code for their task and is preferable to requiring certain code. For example, the open

standard for telephone interconnection allows consumers to buy many different types of phones with an assurance that they will operate with each other.

Open standards are typically in the form of a written specification. The specification notes the requirements to meet the standard and allows for interoperability. Open standards are defined by three elements (Crocker, 1993). First, the standard is publicly available to everyone at a minimal cost. Second, no entity controls the standard or the standard is licensed on "reasonable and nondiscriminatory terms". Third, the development process in creating the standard involves public participation. Examples of open standards on the Internet include the transmission protocols such as FTP, or HTML, which serves as the language for Web pages, and the image format known as JPEG.

Often the development of the specification occurs within Standard Developing Organizations or consortia. Within these groups, the shapes of standards are decided. This can be an important process, because standards can be biased and favor certain values or parties. For example, Netscape's cookies standard consisted of an informally written, page and a half description. However, the final standard written by the IETF, an open standards organization for the Internet, was over 15 pages. This was because the Internet community wanted a more precise specification and one that considered security and privacy issues.

The choice of an open standard is not taken lightly by firms (Shapiro and Varian, 1999). Moreover, even when using an open standard, the economic pressures on firms are so pervasive that they will tend to incorporate proprietary features into their products that are based on open standards. Firms hope to raise switching costs for users, and thereby maintain their share of customers. For example, Cisco is adding proprietary features to its open standards-based routers. These new features can be used only with other Cisco routers. Their goal is to keep customers from switching, by persuading them to use their proprietary features.

The characteristic of modularity is analogous to standards. Modularity breaks down a large piece of code into smaller pieces or modules (Baldwin and Clark, 1999). With modularity, it is possible to replace a module with a different module and the program as a whole would still operate as before. This saves developers time and effort from having to recreate the entire code. Moreover, this style of design allows for considerable flexibility. For example, a developer unhappy with a certain module need only replace that module. This is much simpler than modifying the entire code.

The public policy ramifications for standards are that they permit varied settings, thus allowing consumers to choose a form of private governance. In using open standards, there may be a certain set of standardized code that allows for interoperability. But beyond the basic standard for interconnection, there are many technical and non-technical possibilities for developers. This can result in a variety of products for different tastes all of which compete and perform the same basic function. For example, consider the variation available in telephones all of which use the same common standard. So from a regulatory standpoint, an open standard can allow for certain settings in the code, while encouraging developers to build a diverse portfolio of products.

The public policy ramifications for modularity are similar. Modularity allows developers to modify parts of a code, without have to recreate the entire code. This allows developers to build new applications rapidly. An example of this is with the open source Web browser, Mozilla. It has been designed using modularity. This allows developers to reconfigure and enhance Mozilla for different purposes. The parties include: university researchers experimenting with enhanced privacy protection; firms, such as Netscape that are developing consumer-oriented Web browsers; and the open source community which is designing alternative Web browsers. The result is that modularity allows developers to modify a piece of code to match their needs and values. For regulators, this means code can be used in many ways, because it is much simpler to modify the code's settings and functionality.

IV. Conclusion

This article seeks to provide a better understanding of how code regulates society. To this end, we have analyzed three important universal governance characteristics of code – transparency, defaults, and standards. Our analysis allows regulators to understand how code operates and how to change the settings of code. Transparency is important for understanding how code operates. We argue that in using transparency, regulators must consider both the implementation of transparency and that transparency varies by audience. The characteristics of defaults and standards also affect the settings of code. This ability to set how code operates leads us to argue that defaults are an essential and important characteristic available to regulators. We also discuss issues related to changing default values as well as determining the optimal default value. Finally, we argue that standards and modularity are characteristics that allow people to choose a form of private governance. By supporting these characteristics, regulators can provide users with considerable flexibility on how they are regulated by code. In sum, this analysis is a first step towards manipulating code. We recognize that there are many more governance characteristics that are knobs to manipulate code, such as the flexibility or adaptability of code, obligatory passage points in code, and the characteristic of robustness that ensures code fails gracefully. Our analysis permits policymakers and regulators to begin employing code as an alternative regulatory mechanism to address societal concerns. ■

Note

1 Benjamin Edelman is seeking a declaratory judgment that will allow him to decrypt and publish portions of N2H2's list of blocked sites. By viewing the list, the public can determine what content N2H2 blocks. Edelman argues that this information is important, because it allows the public to evaluate N2H2's effectiveness in blocking content.

References

Ayres, I. and Gertner, R. (1989), "Filling the gaps in incomplete contracts: an economic theory of default rules", *Yale Law Journal*, Vol. 99, pp. 97-130.

Baldwin, C.Y. and Clark, K.B. (1999), *Design Rules: The Power of Modularity*, MIT Press, Cambridge, MA.

Burk, D.L. and Cohen, J.E. (2001), "Fair use infrastructure for rights management systems", *Harvard Journal of Law and Technology*, Vol. 15, pp. 41-83.

Cargill, C. (1989), *Information Technology Standardization: Theory, Process, and Organization*, Digital Press, Burlington, MA.

Crocker, D. (1993), "Making standards the IETF way", *Standardview*, Vol. 1, pp. 48-56.

Foucault, M. (1979), *Discipline and Punishment*, Vintage Books, New York, NY.

Hafner, K. and Lyon, M. (1996), *Where Wizards Stay up Late: The Origins of the Internet*, Touchstone, New York, NY.

Jackson, T. (1996), "This bug in your PC is a smart cookie", *Financial Times*, 12 February.

Katsh, M.E. (1996), "Software worlds and the first amendment: virtual doorkeepers in cyberspace", *The University of Chicago Legal Forum*, Vol. 1996, pp. 335-60.

Katyal, N.K. (2001), "Criminal saw in cyberspace", *University of Pennsylvania Law Review*, Vol. 149, pp. 1003-114.

Katyal, N.K. (2002), "Architecture as crime control", *Yale Law Journal*, Vol. 111.

Kristol, D. and Montulli, L. (2000), "HTTP state management mechanism", RFC 2965, available at: www.ietf.org/rfc/rfc2965.txt

Kristol, D.M. (2001), "HTTP cookies: standards, privacy and politics", *ACM Transactions on Internet Technology*, Vol. 1, pp. 151-98.

Lefebvre, H. (1991), *The Production of Space*, Blackwell Publishers, Oxford.

Lemley, M. and Lessig, L. (2001), "The end of end-to-end: preserving the architecture of the Internet in the Broadband era", *UCLA Law Review*, Vol. 48, pp. 925-72.

Lessig, L. (1999), *Code and Other Laws of Cyberspace*, Basic Books, New York, NY.

Lessig, L. and Resnick, P. (1999), "Zoning speech on the Internet: a legal and technical model", *Michigan Law Review*, Vol. 98, pp. 395-431.

Markus, T.A. (1993), *Buildings and Power: Freedom and Control in the Origin of Modern Building Types*, Routledge, London.

Millett, L., Friedman, B. and Felten, E. (2001), "Cookies and Web browser design: toward realizing informed consent online", in *Conference on Human Factors in Computing Systems*, Association for Computing Machinery, pp. 46-52.

Mitchell, W.J. (1996), *City of Bits: Space, Place, and the Infobahn*, MIT Press, Cambridge, MA.

Pollack, M. (2001), "Opt-in government: using the Internet to empower choice-privacy application", *Catholic University Law Review*, Vol. 50, pp. 653.

President's Critical Infrastructure Protection Board (2002), "The national strategy to secure cyberspace", available at: www.whitehouse.gov/pcipb/.

Reidenberg, J.R. (1998), "Lex Informatica: the formulation of information policy rules through technology", *Texas Law Review*, Vol. 76, pp. 553-93.

Resnick, P. and Miller, J. (1996), "PICS: Internet access controls without censorship", *Communications of the ACM*, Vol. 39, pp. 87-93.

Schwartz, J. (2001), "Giving the Web a memory costs its users privacy", in *New York Times*, 4 September.

Sclove, R. E. (1995), *Democracy and Technology*, Guilford Press, New York, NY.

Shah, R.C. and Kesan, J.P. (2003), "Incorporating societal concerns into communication technologies", *IEEE Technology and Society Magazine*, Fall.

Shapiro, A. (1999), *The Control Revolution: How the Internet Is Putting Individuals in Charge and Changing the World We Know*, Century Foundation, New York, NY.

Shapiro, C. and Varian, H.R. (1999), *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press, Boston, MA.

Sunstein, C.R. (2002), "Switching the default rule", *New York University Law Review*, Vol. 77, pp. 106-34.

Weinberg, J. (1997), "Rating the Net", *Hastings Communications and Entertainment Law Journal*, Vol. 19, pp. 453-82.

Wilhelm, A. (2000), *Democracy in the Digital Age*, Routledge, New York, NY.

Winner, L. (1977), *Autonomous Technology*, MIT Press, Cambridge, MA.