

Governance Characteristics of “Code”: The Role of Transparency, Defaults, and Standards

Rajiv C. Shah* & Jay P. Kesan**

2002 Telecommunications Policy Research Conference
September 28-30, 2002
Alexandria, Virginia

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 the social, technical, and legal ramifications of three universal governance characteristics of code that are significant in regulating behavior. The characteristics are crucial in understanding how code operates as well as the various possible regulatory settings for code. These characteristics studied are transparency, defaults, and standards.

These characteristics were identified through a number of case studies, which explored how code regulates by studying code in a variety of historical periods, developed within different types of institutions, and based upon diverse expectancies of user competencies. The case studies include the Finger protocol, Netscape’s cookies technology, and the Platform for Internet Content Selection developed by the World Wide Web Consortium.

The first characteristic analyzed is transparency. Transparency in code allows people to understand how code operates. This allows people to make an informed decision when using code. For regulators, transparency allows them to ensure code properly addresses issues of societal concern. For example, a key complaint of content filtering software is its lack of transparency. Users and policymakers do not know what material is and is not being blocked. We discuss several important regulatory issues with transparency, including its contextual nature and dependence upon implementation.

The second characteristic is the role of defaults. Defaults set the condition for how code operates in the absence of intervention. Defaults are a method regulators can use to ensure people have multiple options when using code. We discuss the importance of defaults, their power, and also why people may not follow default settings. An example of the use of defaults concerns privacy. Should the defaults of code be designed so people have to intervene to protect their privacy or only intervene when giving up personal information?

The third characteristic is standards. We use this term broadly to encompass open standards that allow for interoperability as well as modularity that saves developers from

recreating code. This characteristic has important economic, social, and legal ramifications. The economic ramifications include the encouragement of competition by allowing for a number of different producers. The social ramifications are allowing people and regulators to set or select the regulatory settings for code. For example, the open source web browser Mozilla, is being reconfigured and enhanced by a number of parties, including university researchers, experimenting with enhanced privacy protection; firms such as Netscape that are developing consumer oriented web browsers; and the open source community who are designing alternative web browsers. As a result, people can choose the code that meets their preferences.

* Doctoral Candidate, Institute of Communications Research, University of Illinois at Urbana-Champaign.

Email: r-shah4@uiuc.edu

** Assistant Professor, College of Law and Institute of Government and Public Affairs, University of Illinois at Urbana-Champaign. Email: kesan@uiuc.edu

This material is based upon 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.

I. Introduction

One of the most significant advancements in communications policy is the recognition that the architecture of information technologies can be used to regulate behavior in a positive manner. Policymakers are using or advocating the use of architectural solutions to address societal concerns with crime,¹ competition,² free speech,³ privacy,⁴ the protection of intellectual property,⁵ and revitalizing democratic discourse.⁶ However, there is little theoretical or empirical knowledge driving these decisions. In effect, policymakers are "shooting from the hip" with the use of architectural solutions. This paper examines three characteristics of code for regulators to consider when employing architectural solutions. These characteristics allow policymakers to modify the regulatory settings of code as well as ensuring these settings are easily understandable. The eventual goal of our work is the development of a theoretical framework for how information technologies can regulate behavior.

¹ Neal Kumar Katyal, *Criminal Law in Cyberspace*, 149 U. PA. L. REV. 1003 (2001).

² The open access movement is based upon the principle that the architecture can support competition as well as providing a platform to support innovative applications. Mark A. Lemley & Lawrence Lessig, *The End of End-To-End: Preserving the Architecture of the Internet in the Broadband Era*, 48 UCLA L. REV. 925 (2001).

³ This Article discusses the use of architectural solutions for addressing the problem of minors viewing inappropriate content. A number of commentators have addressed this issue. Lawrence Lessig & Paul Resnick, *Zoning Speech On The Internet: A Legal And Technical Model*, 98 MICH. L. REV. 395 (1999); Jonathan Weinberg, *Rating the Net*, 19 HASTINGS COMM. & ENT. L.J. 453 (1997). See also David E. Sorkin, *Technical and Legal Approaches to Unsolicited Electronic Mail*, 35 U.S.F. L. REV. 325 (2001) (discussing approaches to limit unsolicited bulk email); CASS SUNSTEIN, REPUBLIC.COM 182-89 (2001) (proposing the redesign of web sites to incorporate links of different viewpoints to provide exposure to differing viewpoints).

⁴ An example of an architectural solution for privacy is the Preferences for Privacy Project (P3P). See William McGeeveran, *Programmed Privacy Promises: P3P and Web Privacy Law*, 76 N.Y.U. L. REV. 1813 (arguing for P3P as a solution to privacy problems). See also Malla Pollack, *Opt-In Government: Using the Internet to Empower Choice—Privacy Application*, 50 CATH. U. L. REV. 653 (2001) (proposing the creation of a government search engine that only links to web sites that protect a user's privacy); Shawn H. Helms, *Translating Privacy Values With Technology*, 7 B.U. J. SCI. & TECH. L. 288 (2001) (arguing the government, privacy advocacy groups, and users should support the adoption of privacy enhancing technologies).

⁵ Dan L. Burk & Julie E. Cohen, *Fair Use Infrastructure for Rights Management Systems*, 15 HARV. J.L. & TECH 41 (2001) (providing an example of an architectural solution to allow fair use in digital based intellectual property). The media industry has been very vocal in supporting architectural solutions to protection their intellectual property. Amy Harmon, *Hearings on Digital Movies and Privacy*, N.Y. TIMES, Mar. 2, 2002, available at <http://www.nytimes.com/2002/03/01/technology/01DIGI.html>.

⁶ See ANTHONY G. WILHELM, DEMOCRACY IN THE DIGITAL AGE 44-47 (2000); Cathy Bryan et al., *Electronic Democracy and the Civic Networking Movement in Context*, in CYBERDEMOCRACY 1 (Roza Tsagarousianou et al. eds., 1998).

Our work begins with the recognition that while scholars have long theorized and analyzed the effect of law upon society, the study of architecture is noticeably immature. A variety of disciplines have recognized the role of the architecture, consisting of man-made physical structures, on society. Social, political, and cultural theorists have written about the role of architecture and space upon the production of social space and technologies.⁷ Criminologists have theorized and studied how the architecture can be shaped to prevent crime.⁸ Architectural theorists have analyzed how architecture regulates behavior.⁹ Design theorists have struggled with how to make architecture intuitive and easily comprehensible to people.¹⁰ Finally, a more practical school of thought focuses on using architecture to encourage consumer spending.¹¹ As a result, there is a wide variety of theoretical approaches for understanding how architecture regulates behavior. However, none of these approaches appears to be easily extended to code.

In the realm of information technologies, architecture has a more profound influence than in the built environment. The term "code," as we use it, refers to the physical hardware and software of information technologies. In *City of Bits*, William Mitchell wrote how code can control our behavior.¹² 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

⁷ HENRI LEFEBVRE, PRODUCTION OF SPACE (1991); MICHEL FOUCAULT, DISCIPLINE AND PUNISHMENT (1979).

⁸ RAY C. JEFFERY, CRIME PREVENTION THROUGH ENVIRONMENTAL DESIGN (1971); OSCAR NEWMAN, DEFENSIBLE SPACE (1972); RONALD CLARK, SITUATIONAL CRIME PREVENTION (1997); Neal Kumar Katyal, *Architecture as Crime Control*, YALE L.J. (2002).

⁹ THOMAS MARKUS, BUILDINGS & POWER: FREEDOM AND CONTROL IN THE ORIGIN OF MODERN BUILDING TYPES (1993).

¹⁰ DONALD A. NORMAN, THE DESIGN OF EVERYDAY THINGS 1998.

¹¹ SUSAN DAVIS, SPECTACULAR NATURE: CORPORATE CULTURE AND THE SEA WORLD EXPERIENCE (1997); DOUGLAS RUSHKOFF, COERCION: WHY WE LISTEN TO WHAT THEY SAY (1999).

¹² WILLIAM MITCHELL, CITY OF BITS (1996).

surveillance, or it may aid in preserving anonymity. Mitchell's observations on the power of code to regulate behavior have been formalized by a number of scholars.¹³

Code not only affects the communication experience but also can affect fundamental societal values such as freedom of speech in several ways. For example, code in the form of filtering software is often used to ensure minors do not access inappropriate material. The imprecise nature of filtering software results in limiting the availability of meaningful material concerning AIDS and breast cancer.¹⁴ Others believe that the 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.¹⁵ 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 which cannot afford to pay for overlay networks. These are all examples of how code affects freedom of speech. Code also 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 manmade and is not subject to 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. Already, code-based solutions are being

¹³ LAWRENCE LESSIG, CODE AND OTHER LAWS OF CYBERSPACE (1999); Joel R. Reidenberg, *Lex Informatica: The Formulation of Information Policy Rules Through Technology*, 76 TEX. L. REV. 553 (1998); M. Ethan Katsh, *Software Worlds and the First Amendment: Virtual Doorkeepers in Cyberspace*, 1996 U. CHI. LEGAL F. 335.

¹⁴ Jonathon Weinberg, *Rating the Net*, 19 HASTINGS COMM. & ENT. L.J. 453 (1997).

considered for issues such as privacy and free speech. Third, society needs to shape code early before it becomes entrenched, and hence, more difficult to shape. 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 the myriad of ways code regulates, the case studies took into account the early Internet as well as contemporary examples of code. Moreover, the case studies touched upon social 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 the three governance characteristics of code, which are transparency, defaults, and standards. These characteristics have implications for regulators who are seeking to employ code as an architectural method for regulating behavior. Transparency is concerned with whether users and regulators can understand and verify how code operates. Defaults and standards are important characteristics on the settings of code. Regulators may influence these characteristics and change code's settings to ensure it complies with societal interests.

¹⁵ ANDREW SHAPIRO, *THE CONTROL REVOLUTION: HOW THE INTERNET IS PUTTING INDIVIDUALS IN CHARGE AND CHANGING THE WORLD WE KNOW* (1999).

II. Summaries of the Case Studies

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

A. Finger

The finger command was an early computer network application that allowed people to see whom else was using any specific computer system as well as providing basic information on that person. It provided three basic pieces of information.¹⁶ The first piece of information listed who all were using the specific computer network. Second, the finger command could be used to determine if a specific person was using the computer network. Finally, the finger command could retrieve information about a particular person, such as their telephone number. For example, in response to the command `finger atstarr@unix.amherst.edu` a computer running the finger program would respond with the following information:¹⁷

```
Login name: atstarr           In real life: Andrew Starr
Office: Kansas City         Home phone:
Last login Mon Nov  8 13:22 on ttyre from sdn-ar-001mokcit
Plan:
To come so far one must be brave.
```

```
ATStarr@Amherst.Edu
http://www.amherst.edu/~atstarr/menu.html
```

¹⁶ D. Zimmerman, *The Finger User Information Protocol*, RFC 1288, Dec. 1991, available at <http://www.ietf.org/rfc/rfc1288.txt>.

¹⁷ Andrew Starr, *Finger*, available at <http://www.emailman.com/finger/> (last visited Aug. 20, 2002).

The information that the finger command returns is customizable by the site. Here the finger command returns a login name, real life name, office, and home phone. The finger command also reveals the last time the person has used the computer system and the actual computer they last used. The information below the Plan line (Plan:) is totally customizable by the user. Here the person is providing another email address and a web site address.

The finger command was the source of one of the Internet's first flame wars.¹⁸ Although this debate over online privacy occurred over twenty 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).

A short time ago, the CMU Finger program was endowed with the ability to reveal when a user last logged in and when that user last read his/her mail with our RDMAIL program. To respect the privacy of the individual I arranged for two user profile bits to be added to our existing profile facility (which determines whether a user automatically sees a bulletin board, or gets a message when mail arrives etc.) The two new bits determine whether Finger may reveal the date/time a user last logged in and the date/time that the MAIL.MSG file was last changed. The default setting for the profile bits inhibits Finger from revealing this information. (Email by Ivor Durham)

To recapitulate, Ivor Durham added some privacy bits to allow a user to turn off information about their behavior. This information was (1) whether the user is currently logged on, (2) when the user had logged off, (3) whether there was any mail in the mailbox, (4) when the user has last read mail, and (5) 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 resulting controversy pitted the rights of the individual against that of the community. For example, Ivor Dunham (the person who that ensured the privacy bits were added) discussed this conflict between the rights of the individual and the community. He said, "the social implications here are not that the decision violates any 'right' of a close knit community, but the rights of the individual. We opted for siding with the individual. Simple as that, I think. Individual".

Other people also valued the rights of individuals over communities:

I realize that we are a "friendly", "cooperative", ..., community, and I expect that most people won't mind this information being released. But you must recognize that the information can act [to] coerce people into a particular lifestyle. A major attraction, to me, of netmail is that I read it when I want to -- not when the sender wants me to. Last week I chose not to read my mail for 4 days. As soon as it becomes public that I did that, however, there can/will be both external and internal pressure to read it everyday. Now, maybe that's good too, the goodness/badness is not the issue. The point is that simply because we are a friendly community does not give everyone the right to know certain things about me -- and only I can determine what the things I want known are. Thus, even though information about when I last logged in may seem trivial, its not up to us to decide whether that is something that a particular individual wants known. Again, I think Ivor made exactly the right choices.

A few people felt that in this case the community's right was of more importance than the individual's right:

¹⁸ KATIE HAFNER & MATTHEW LYON, WHERE THE WIZARDS STAY UP LATE: THE ORIGINS OF THE INTERNET 216 (1996).

In the 5 years that I have been at CMU, I have watched a decline of direct person-to-person talking and an increase of computer-based conferencing of all kinds. I have seen people send computer mail to someone ten feet away to avoid having to get out of a chair and actually use his vocal cords. I have seen an increasing number of design discussions take place entirely within the confines of the computer. And I have watched the "sense of community" that is so valuable to research institutions become weaker and weaker" [. . .] Personal privacy in the face of computer systems and data bases is a very sticky problem, but it is not the problem that I was trying to address with my complaints about closedness. I was worried, and still am worried for that matter, about the decay of our lab as a place in which to do research and creative thinking. I certainly don't want some credit bureau to know when I last logged out of the machine, but I certainly do want my co-workers to know when I did. The question of "who is asking" is to me very important.

This discussion over privacy rights in 1979 resembles many contemporary concerns over privacy. In our future work, we will carefully analyze this controversy. However, for this paper we choose to focus on how changes in code affected privacy.

B. Cookies

Cookies are part of several technologies Netscape developed in order to position its web servers for commerce. The cookies technology was the most innovative feature and one that would forever alter the web. According to Lessig, "before cookies, the Web was essentially private. After cookies, the Web becomes a space capable of extraordinary monitoring."¹⁹ In early web browsers, the Internet was a stateless place. A stateless web is analogous to a vending machine. It has little regard for who you were, what product you are asking for, or how many purchases you have made. It has no memory. The lack of statelessness on the web makes commerce difficult. For example, without a state mechanism, buying goods is analogous to using a vending machine. You could not buy more than one product at a time and there would

be no automatic one-click automated shopping feature that remembers your personal information.

Cookies solve the statelessness problem by storing data on the users computer. Lou Montulli and John Giannandrea developed the cookies technology or more formally titled, Persistent Client State HTTP Cookies.²⁰ Programmers used the term cookies to refer to a small data object passed between cooperating programs. Similarly, Netscape would use cookies to pass information between a user's computer and the web site they were visiting. The first use of cookies was by Netscape to determine if visitors to Netscape's web site were repeat visitors or first time users.²¹

Cookies were incorporated in the earliest version of Netscape's web browser released in 1994. Cookies were not made public to users in several ways. Netscape turned the feature on by default without notifying or asking the consent of users.²² Second, there was no notification mechanism to alert people when cookies were being placed on their computer. Users did not know that information about them was being saved. Third, the cookies technology was not transparent. Examining a cookies file provides no information about what is stored in the cookie file. Fourth, there was no documentation available that explained what cookies were and their privacy implications.²³

¹⁹ John Schwartz, *Giving the Web a Memory Cost Its Users Privacy*, N.Y. TIMES, Sep. 04, 2001, available at <http://www.nytimes.com/2001/09/04/technology/04Cook.html>.

²⁰ *Id.* See also Netscape, *Persistent Client State HTTP Cookies*, at http://home.netscape.com/newsref/std/cookie_spec.html (last visited Sep. 19, 2001) (the original Netscape specification on cookies); Persistent Client State in a Hypertext Protocol Based Client-Server System, U.S. Patent No. 5,774,670 (issued June 30, 1998) (filed on Oct. 6, 1995), SIMON ST. LAURENT, COOKIES (1998) (providing an excellent overview of cookies and how to use them).

²¹ Netscape browsers default home page is Netscape's web site. This meant every user would visit Netscape's web site at least once. See Alex S. Vieux, *The Once and Future Kings*, RED HERRING, Nov. 1, 1995.

²² Lynette I. Millett et al., *Cookies and Web Browser Design: Toward Realizing Informed Consent Online*, CHI 2001 Proceedings, at 46 (2000) (conducting an analysis of cookie management tools in web browsers).

²³ In fact, technically proficient users in 1995 called for Netscape to document the cookies feature. For example, Marc Hedlund listed the following problems with Netscape's implementation of cookies, "1. Why doesn't the word "cookie" appear in the Netscape Online Handbook?, 2. Why isn't the cookie specification URL given in any

Netscape incorporated cookies into its web browsers released in 1994. It was not until early 1996 that the public became aware of cookies. The Financial Times broke the story on February 12, 1996 with an article on cookies and privacy.²⁴ 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 to Internet Engineering Task Force (IETF) to begin 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.²⁵ The IETF found that Netscape's implementation of cookies was fraught with privacy and security problems.

The most serious problem was third party cookies. The intent of Netscape's cookies specification was to only allow cookies to be written and read by the web site the person was visiting. For example, if the New York Times placed a cookie on a computer, Amazon.com could not read or modify the New York Times cookie. This provided security and privacy by only allowing sites access to information they authored. However, Netscape's cookies specification allowed third party components of a web page to place their own cookies. This created a loophole by which third parties could read and write cookies. This security and privacy defect was the outgrowth of the rapid development and incorporation of the cookies technology.

README or implementation notes file?, . . . [3] How are users supposed to know what information is being kept about them, or for how long?" Marc Hedlund, *State Wars, part XI (was: Revised Charter)*, HTTP-WG MAILING LIST, Nov. 1, 1995, available at <http://www.ics.uci.edu/pub/ietf/http/hypermail/1995q4/0161.html>.

²⁴ Tim Jackson, *This Bug in Your PC is a Smart Cookie*, FIN. TIMES, Feb. 12, 1996. The next day a similar story appeared in the United States. See Lee Gomes, *Web 'Cookies' May Be Spying on You*, SAN JOSE MERCURY NEWS, Feb. 13, 1996.

²⁵ David M. Kristol, *HTTP Cookies: Standards, Privacy, and Politics*, ACM TRANSACTIONS INTERNET TECH., Nov. 2001, at 10.

This loophole has led to a new breed of businesses, the online advertising management companies.²⁶

Third party cookies can be used by online advertising companies to create detailed records on a person's web browsing habits. Many sites contract out their banner advertising to advertising management companies. These companies find advertisers for web sites and ensure that their banners appear on the web site. For example, DoubleClick sells advertising space on sites such as ESPN and the New York Times. DoubleClick is also responsible for placing the banner advertising on their client's web site. Through the loophole of third party cookies, DoubleClick uses its advertising banners on an ESPN web page to place a cookie when a person visits ESPN. DoubleClick can then read and write to that same cookie when the same person visits the New York Times web site.²⁷ This allows DoubleClick to aggregate the information about a person's web surfing from its client web sites. They can then create a detailed profile of a person's surfing habits. This has obvious and serious privacy implications.²⁸

The IETF's cookies standard is critical of third party cookies allowed by Netscape's cookies specification. The standard states that third party cookies must not be allowed. It does allow an exception if the program wants to give the user different options. However, the baseline default must be set to off.²⁹ It also requires that the user be able to disable cookies, determine when a stateful session is in progress, and be able to save cookies depending upon the cookies domain. This last one is especially significant, because it allows users to manage what sites can and can't place cookies.

²⁶ Schwartz, *supra* note 19.

²⁷ Kristol, *supra* note 25 at 30.

²⁸ Michael Gowan, *How It Works: Cookies*, PC WORLD, Feb. 22, 2000, available at <http://www.pcworld.com/hereshow/article/0,aid,15352,00.asp>.

²⁹ David Kristol & Lou Montulli, *HTTP State Management Mechanism*, RFC 2965, Oct. 2000, available at <ftp://ftp.isi.edu/in-notes/rfc2965.txt>. A central premise of the standard is that informed consent should guide the design of systems using cookies.

Despite the IETF's standard, Netscape's and Microsoft's browsers have allowed third party cookies. They never fully followed the standard. The latest version of web browsers still accept third party cookies by default to satisfy the advertising management companies. However, they have improved their cookie management tools. This allows people to manage what sites can and can't place cookies.

C. Platform for Internet Content Selection

The history of the Platform for Internet Content Selection (PICS) begins with proposed legislation to regulate indecent speech on the Internet by Senator Exon in the summer of 1994.³⁰ Senator Exon reintroduced his legislation in February 1995. This would eventually become the Communications Decency Act (CDA).³¹ On June 14, 1995, the Senate approved an amendment (the CDA) to the United States Telecommunications Competition and Deregulation Act of 1995 that would make it unlawful to transmit indecent material over the Internet to minors. This proposed legislation was followed by the now infamous Time cover story on cyberporn³² This combination of media and political pressure threw the upstart Internet companies into action.

In June of 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 went beyond movie ratings of content but was more general to encompass other classification schemes such as

³⁰ 140 CONG. REC. S. 9745 (1994) (including the amendment to S. 1882 by Senator Exon) *available at* http://www.eff.org/Censorship/Internet_censorship_bills/exon_s1822.amend.

³¹ Communications Decency Act of 1995, S. 314, 104th Cong. (1995).

³² Philip Elmer-Dewitt, *On a Screen Near You: Cyberporn*, TIME, July 3, 1995, *available at* <http://www.time.com/time/magazine/archive/1995/950703/950703.cover.html>.

the Library of Congress cataloging scheme. The second goal was to allow individuals to selectively access or block certain content.

The result was the Platform for Internet Content Selection (PICS). This is a set of standards for labeling material on the Internet.³³ 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 MPAA movie rating vocabulary of PG, R, and X. The PICS system 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 who should pay attention to which labels. Instead, people will be 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 rely on a third party rating. For example an organization may set up a labeling bureaus to assist people that wish to have “hate” sites blocked.³⁴

The PICS labels are incorporated into the web pages or into a header by the web server.³⁵ This allows the web browser to detect the label and then based upon the users preferences, a web browser will either display or not display the web page. An example of a label incorporated in a web page is: <META http-equiv="PICS-Label" content='(PICS 1.0 "http://www.classify.org/safesurf/" 1 r (SS~~000 4 SS~~001 5 SS~~004 2 SS~~007 2 SS~~008 3))>

³³ For all the PICS technical specifications, see <http://www.w3c.org/PICS/>.

³⁴ Paul Resnick, *Filtering Information on the Internet*, SCI. AM., Mar. 1997, at 106.

³⁵ The method include using a an extra header in the HTTP header stream that precedes the contents of documents that are sent to web browsers, run a label bureau at a specific location on your server, or embed labels in HTML documents using a META tag.

This label uses the SafeSurf rating system. As a result, it would be translated that this web page contains a Rating of Suitable for Older Teens, Profanity with a Caution level of 5, Nudity with a caution level of 2, Intolerance with a level of 2, and Drug Use with caution level of 3.³⁶ The label becomes clearer when you realize that the Safesurf rating system of SS~~000 4 means that a web site has a rating of suitable for older teens and that SS~~001 refers to profanity with a caution. The style of the label is influenced by the LISP, an early computer language. However, if Resnick, a co-designer of PICS, had to redesign PICS all over again, he would also slightly alter the design of the PICS labels. He would make them less like LISP and more like C or HTML. This would make them more understandable by a wide segment of the technical community.³⁷

While the PICS team was focusing on protecting children from Internet pornography, they knew that the PICS specification was capable of much more. PICS was developed as a generic method for providing information about web pages for two reasons. The first is the need for various types of labels to describe pages that are inappropriate for children by such varied criteria as violence, nudity, sex, and racism. The second was the impetus of the designers to “over-engineer” PICS as long as its performance did not suffer. Thus led to the PICS specification as a general “metadata” system.³⁸ Metadata is the term for information that provides information about data. For example, the rating of web pages would produce metadata. The metadata would the rating, such as PG, of the web page. However, metadata, and thus PICS, could also be extended to provide information about Copyright, Payment Information,

³⁶ Ray Soular and Wendy Simpson, *The SafeSurf Internet Rating Standard*, available at <http://www.safesurf.com/index.html> (Dec. 1995).

³⁷ Interview with Paul Resnick, Designer for PICS, in Bloomington, Ill. (Aug. 10, 1999).

³⁸ *Id.*

Data Authenticity, Reviews, Nominations, SOAPs, “Meta-Information”, and URCs.³⁹ The PICS team knew and proposed these possible applications.

By November 1995, the PICS technical subcommittee released the PICS specifications for public review. This was followed by several presentations at leading conferences on the Internet and the World Wide Web. By February 1996, Microsystems put the first PICS ratings server on the Internet.⁴⁰ By March, a number of companies including Netscape and Microsoft had publicly committed to using PICS in their browsers.⁴¹ And by December 1996, the W3C made PICS an official "recommendation", the highest recognition a standard can receive by the W3C.⁴²

For PICS to be successful using self-labeling, it would be necessary to gain a critical mass of labeled web sites. Such a critical mass would allow people to turn on their PICS filter and use the web. Without a critical mass, users would have to allow unlabeled content through their PICS filter. And if there was plenty of unlabeled content, users would have little reason to turn their filters on, since the filtering features in browsers would be ineffective. After all, if less than 1% of web pages are labeled, then what use is a web filter?

There were several steps taken to ensure content was widely labeled. First, there was the attempt to persuade search engines to ignore non-labeled web pages. Robert Davis, the president of Lycos stated that he “threw a gauntlet to other search engines . . . saying that

³⁹ These possible applications were discussed on a presentation on PICS at an Oct. 30, 1995, conference titled, The First International Conference on Electronic Commerce at the University of Texas, *see* http://www.w3.org/Talks/9510_UTeCommerc/.

⁴⁰ W3C, *Microsoft Teams With Recreational Software Advisory Council To Pioneer Parental Control Over Internet Access*, available at <http://www.w3.org/PICS/960228/Microsoft.html> (Feb. 28, 1996).

⁴¹ W3C, *PICS Picks Up Steam*, available at <http://www.w3.org/PICS/960314/steam.html> (Mar. 14, 1996).

⁴² W3C, *W3C Issues PICS as a Recommendation*, available at <http://www.w3.org/Press/PICS-REC-PR.html> (Dec. 3, 1996).

collectively we should require a rating before we index pages."⁴³ In effect, this would leave unrated sites as "invisible" to search engines. Both Yahoo! and Excite supported this strategy. In fact, many web search engines seriously considered limiting their results to pages that were labeled with PICS labels. Jim Miller recalls that the technical people at the search engines could easily implement such filtering, however there was never any communication with people "at the right level" to put this into use. For example, "Alta Vista had implemented part of it [PICS filtering] and given us some of the results."⁴⁴ However, none of the search engines ever limited their results to PICS based pages. Jim Miller surmises that this was because search engines did not know how to make money off such filtering nor would they make any friends with such filtering.⁴⁵ Today, these features exist in a number of search engines that allow for "family filtering" of results.⁴⁶

Second, there was talk about setting the default in web browsers to filter web pages. In another scare to unrated sites, Ken Wasch, the president of the Software Publishers Association announced that the next version of Microsoft's Internet Explorer would have a default set to view only rated or labeled web pages. However, Microsoft quickly denied that, and stated that the default would not change.⁴⁷ Microsoft has continued to keep the default to off for their filtering features.

Third, there was the attempt have web sites label their content. One important category that failed to rate their content was news sites. A special rating was created for news sites, in order to encourage them to label their content. This label could allow parents to ensure that their

⁴³ Declan McCullagh, *At the Censorware Summit*, NETLY NEWS, July 16, 1997, available at <http://www.onmagazine.com/on-mag/reviews/article/0,9985,12297,00.html>.

⁴⁴ Interview with James Miller, Designer for PICS, in Bloomington, Ill. (Aug. 13, 1999).

⁴⁵ *Id.*

⁴⁶ For example, Google offers its SafeSearch feature. See Google, *SafeSearch Filtering*, available at <http://www.google.com/help/customize.html#safe> (last visited Aug. 20, 2002).

children would not be exposed to the gruesome violence of some new stories such as terrorist actions. Moreover, it could allow users to block access to non-journalistic web sites easily. However, for such a system to work, it would be necessary to decide what sites were sufficiently journalistic to label themselves as a news site. Moreover, most news sites were fundamentally concerned with free speech and therefore opposed to content labeling. The proposed N rating flopped. Thus most news sites still do not carry PICS labels.⁴⁸

These approaches failed for a number of reasons. First and foremost, after the CDA was struck down, the tide turned against PICS.⁴⁹ PICS went from a solution to the problem. People realized it could be more insidious and affect free speech much more than the CDA.⁵⁰ As a result there was little support for widespread use of PICS. In the end, most sites refused to rate their sites with PICS compliant labels. Today, the number of web sites rated with PICS compliant labels is at best merely 0.4% of all web sites.⁵¹

The use of PICS for third party ratings never became viable. A system of third party labeling bureaus never emerged because of the lack of economic incentives and the necessary software tools. A system of third party bureaus was seen by Resnick as the most realistic scenario through which PICS would become useful. However, the existing filtering software

⁴⁷ McCullagh, *supra* note 43.

⁴⁸ Tim Clark and Courtney Macavinta, *RSAC Shelves News Rating*, CNET NEWS.COM, Sep. 10, 1997, available at <http://www.news.com/News/Item/0,4,14139,00.html>.

⁴⁹ *Reno v. American Civil Liberties Union*, 521 U.S. 844, 117 S.Ct. 2329, 138 L.Ed.2d 874 (1997).

⁵⁰ Simson Garfinkel, *Good Clean PICS*, HOTWIRED, May 1997, available at <http://www.hotwired.com/packet/garfinkel/97/05/index2a.html> (last visited May 19, 1999); Lawrence Lessig, *The Tyranny in the Infrastructure: The CDA Was Bad - but PICS May Be Worse*, WIRED, July 1997, available at http://www.wired.com/wired/5.07/cyber_rights.html; American Civil Liberties Union, *Fahrenheit 451.2: Is Cyberspace Burning? How Rating and Blocking Proposals May Torch Free Speech on the Internet*, available at <http://www.aclu.org/issues/cyber/burning.html> (last visited Sep. 13, 2001).

⁵¹ There are about 120,000 web sites that have adopted PICS. However, the adoption of PICS is lagging behind the growth of the Internet. At last count there are over 30,775,624 web sites. See Wendy McAuliffe, *Home Office Web Site Adopts Adult Rating*, ZDNET, May 4, 2001, available at <http://news.zdnet.co.uk/story/0,,s2086022,00.html> (noting the lack of progress of PICS labels); Netcraft, *August 2001 - Web Server Survey*, available at <http://www.netcraft.com/Survey/Reports/0108/> (counting the number of web servers on the Internet).

companies did not see any commercial scenario for operating public label bureaus. The existing filtering companies incorporated the PICS specifications into their own products, but never committed to running public labeling bureaus.

In tandem with the lack of a business model for public labeling bureaus was the lack of support from software vendors. The server software for creating label bureaus was only developed for a few servers. Most notably, Netscape and Microsoft did not have this feature. The W3C's web page indicates the only commercial server software was IBM's Internet Connection Server. In sum, once the CDA was found unconstitutional by the Supreme Court, the development of software for PICS was essentially stopped. The consequent lack of support from commercial filtering firms, the W3C's members, and other children's groups led to the abandonment of PICS.

III. Governance Characteristics

This section focuses on the three governance characteristics of code. We chose to discuss these three characteristics because we found they played an important role in our case studies. We are aware that other theoretical approaches may consider other characteristics. However, we believe these characteristics are universal and have implications for regulators seeking to employ code. 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 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. The resulting opaqueness of cookies did not allow people to understand how the technology operates. As a result of the lack of information, users had to look elsewhere to understand the 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 exploratory research has revealed two important regulatory issues with transparency.

First, transparency varies by audience. What is transparent for one group of people may be opaque to another group. This is most evident in complex code, such as the music industry's Digital Rights Management (DRM) schemes to limit piracy. An example of such a scheme is the Secure Music Digital Initiative (SDMI). SDMI has complex rules for what music can be copied, under what circumstances, and to what devices. For example, only a limited amount of copies can be made. Or content can be transferred between a computer and an mp3 player, but not transferred onto a CD for a CD player. As a result, transparency exists only for more sophisticated users that understand the rationale behind the DRM's schemes. But for everyday users, such code-based schemes are opaque.

The contextual nature of transparency means that code must be differently designed for different audiences. For most audiences, this requires developers to hide the complexities of the code away from the audience. For example, WordPerfect contains a command that reveals the

formatting codes for a document. This command shows how a document is formatted. This allows a person to quickly identify and rectify problems with formatting in a transparent manner. In this case, the complexities of the formatting are made transparent for the average person.

An extreme level of transparency can be found in the development of the open source web browser Mozilla. Mozilla is one of many code-based projects by the open source movement. An important characteristic of the open source movement's code is that the source code is publicly available. The source code is the human readable part of the software instructions. Because the source code is open to inspection, it is easy to see what the source code is capable of accomplishing as well as 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 allows users a level of trust in the code, because they can fully examine the code.

For a regulator, this variation in transparency must be considered in analyzing code. Micro-level transparency issues are concerned with whether the average person can make an informed decision about whether to use code and how best to employ 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 the design of a cookie management tool. At a higher level, a macro-level transparency issue can concern whether code properly addresses issues of societal concern. This type of transparency is analogous to the transparency we require of government legislation. For example, a macro-level transparency issue could involve an assessment of filtering software by examining the rules for excluding sites in filtering software.⁵² The more transparent the filtering software, the simpler this assessment would be for a regulator.

⁵² 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. See Ross Kerber, *ACLU Sues Firm Over Filtering Software*, B. GLOBE, July 26, 2002, available at

Second, transparency is dependent upon 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 such that users had no idea that cookies existed. Even after this, Netscape provided little or no documentation on cookies. Even once cookies became widely recognized as a privacy issue, the implementation was far from transparency. It took several years before Microsoft's and Netscape's browsers provided users with information on cookies and control over cookies.

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 allows 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 of 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

http://www.globe.com/dailyglobe2/207/business/ACLU_sues_firm_over_its_filtering_software+.shtml; Benjamin Edelman, *Edelman v. N2H2, Inc. - Case Summary & Documents*, available at <http://cyber.law.harvard.edu/people/edelman/edelman-v-n2h2/> (last modified Jul. 30, 2002).

on cookies. It would not be until 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. 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 use third party cookies.

An example of the importance and relevance of defaults can be seen in the debate over the finger command at Carnegie Mellon University (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. The setting of the defaults to favor privacy was contested by some. 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 of 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. Defaults are powerful, because generally most people don't change the default settings. We believe one of the reasons for the failure of PICS to become widespread was that it did not operate by default in web browsers and search engines. As a result, web sites had little incentive to label their web sites. The power of the default setting means the initial setting is of considerable importance. One of the participants in the finger controversy told us that "for a long time, probably predating this incident, my attitude has been that you have to set things up in a way, for a complex technical

system, to protect users when they are naive and allow them to change things when they are less naive.”

It is not well understood why people follow the default value. Two general reasons for this are that people are lazy or people are not sophisticated enough to change the setting. From a regulatory perspective, it is acceptable that people don't change defaults because they are lazy. However, people should still be aware and informed of the presence of the defaults. A default setting is essentially useless if a person doesn't know about or is not sophisticated enough to change the option. One of the motivations for the heated debate over the finger command was to raise awareness of the default settings. The second reason people don't change defaults is their lack of technical sophistication. If people can't figure out how to change a default, they can't 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 the default settings for cookies. A second solution is to provide guidance, education, or documentation on how to change the default settings.

Default values have a number of different purposes. The setting and switching of default values has been studied by behavioral economics.⁵³ 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. Let us start with 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, there are three circumstances when

⁵³ The following discussion was inspired by a recent article by Cass Sunstein on the role of defaults in employment law. Cass R. Sunstein, *Switching the Default Rule*, 77 N.Y.U. L. REV. 106 (2002).

the Coase theorem is inapplicable to defaults in code. 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 had the default been set to favor the other party. Consequently, the parties will not bargain to the same result if given differing initial starting defaults.⁵⁴ 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, Sunstein has argued the initial allocation of the default has another effect besides the endowment effect.⁵⁵ 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 create “penalty defaults” to ensure disclosure between the parties.⁵⁶ 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 consumers were provided adequate information on all the potential uses of their medical information. In sum, these three circumstances help explain why defaults in code are important and how they can be used by regulators.

Finally, regulators should understand that defaults are an essential part of code. A typically 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 a developer create a default to give people choices, because code is not fixed but mutable. For example, regulators could

⁵⁴ *Id.*

⁵⁵ *Id.*

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.

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.⁵⁷ In relationship 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 email between different computers running different software, there is a need for standards that specify the format for the transmission of email messages. A related code characteristic is modularity, which can be considered a higher form of a standard. Modularity breaks down a large piece of code into smaller pieces or modules.

Of interest to us are open standards. 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.

⁵⁶ Ian Ayres & Robert Gertner, *Filling the Gaps in Incomplete Contracts*, 99 YALE L.J. 97 (1989).

⁵⁷ CARL CARGILL, *INFORMATION TECHNOLOGY STANDARDIZATION: THEORY, PROCESS, AND ORGANIZATION* (1989).

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.⁵⁸ One, the standard is publicly available to everyone at a minimal cost. Second, no entity controls the standard or that 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. In 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 Internet Engineering Task Force, an open standards organization for the Internet, was over fifteen 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.⁵⁹ 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 based on open standards. They are hoping to raise switching costs for users and therefore maintain a higher share of customers. For example, Cisco is adding proprietary features to its open standards-based routers. These new features can

⁵⁸ David Crocker, *Making Standards the IETF Way*, STANDARDVIEW, Jan. 1993, at 1, available at <http://www.brandenburg/ietf/ietf-stds.html>

⁵⁹ CARL L. SHAPIRO & HAL R. VARIAN, *INFORMATION RULES* (1998)

be used only with other Cisco routers. Their goal is to keep customers from switching, by persuading them to use the proprietary features.

The characteristic of modularity is analogous to standards. Modularity breaks down a large piece of code into smaller pieces or modules.⁶⁰ 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 from recreating the entire code. Moreover, this style of design allows for considerable flexibility. For example, a developer unhappy with a certain module could replace only that module. This is much simpler than modifying the entire code.

The public policy ramifications for standards, whether open standards or modularity, are that they allow for varied settings, thus allowing people 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 possibilities for developers. This can result in a variety of products for different tastes that all compete and can perform the same basic function. For example, consider the variation available in telephones that all use the same common standard. So from a regulatory standpoint, an open standard can allow for certain settings for code, while also 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 recreating 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 by 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 who are 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 sought to provide a better understanding of how code regulates society. To this end, we have analyzed three important universal characteristics of code that are important for regulators. These characteristics allow regulators to understand how code operates and how to change the settings of code. These characteristics included transparency, which is important for understanding how code operates. The characteristics of defaults and standards are important for a regulator concerned with a code's setting. Our next goal will be to address characteristics that become relevant when code fails. For regulators, it is important not only to understand how code operates normally, but also what happens when code fails. This will allow policy makers and regulators to begin to employ code as an alternative regulatory mechanism to address societal concerns.

⁶⁰ CARLISS Y. BALDWIN & KIM B. CLARK, DESIGN RULES: THE POWER OF MODULARITY (1999)