Confucianism and Technical Standards

Carl M. Johnson

Confucianism Seminar

December 11, 2006

Confucianism is often characterized as a primarily social philosophy, relating to the interpersonal actions of human beings. It can for this reason be called an ethical philosophy, but it is important not to let this description narrow one's conception of the scope of Confucianism. In the West, it is common to understand life as largely ethically neutral with only occasional moral "events" in which an agent must make an active decision that is either "right" or "wrong." However, for a Confucian there is more to being an ethical person than making occasional momentous moral decisions. Instead as Kupperman says "Naturalness Revisited,"[1] for Confucius being ethical means cultivating one's sense of "style" or aesthetics by grounding one's interactions with others in one's own sense of self as a member of a community. Within the human realm, there are different kinds of norms that overlap in some ways but not others. For example, it is possible for an action to be legal but not moral, such as an extramarital affair. It is also possible for an action to be moral but not legal, such as an act of civil disobedience. There are also interpersonal norms which are neither legal nor moral but which nevertheless are important for human interaction. I shall call this class broadly "standards" in this essay and focus chiefly on how Confucian thinking can help us understand their role in modern computer centric, technocratic society. After setting out examples of standards and Confucian thinking broadly, this essay will delve in depth into how standards are implemented (with POSIX support in Windows NT as a paradigmatic example of the failure of traditional rule-based Western-style standards), how stand-

[1] Kupperman, Joel J. "Naturalness Revisited: Why Philosopher Should Study Confucius" in *Confucius and the Analects, New Essays*, ed. by Bryan Van Norden, Oxford Univ. Press, 2002.

ards are promulgated (with the "Web standards movement" as a paradigmatic example of how Confucian social pressures can lead to standards acceptance after more formal pressures have failed), and how standards are created (with the IETF as a paradigmatic example of pragmatically evolved standards).

Contents:

I. What are standards?

Standards enter into every level of human education and interaction but often invisibly. Spoken language is arguably the first and still greatest standard ever implemented by humans. Its rules are a tangled mess created by a combination of consensus, social norms, political regulation, and human biology, but it allows us to transmit any rational concept (or at least, innumerably many of them) to any other person conversant in the same language. This is a powerful feat. Writing must come next in a catalog of great standards, allowing the power of spoken word to be archived and allowing each new generation access to the knowledge of the distant past.

Of course, not all standards are so lofty, nor should they be. There are standards for food and fashion. There are standards that tell us how to turn on the tap of a faucet, standards for how light switches are shaped, and standards for opening a door. One of the joys of international travel is that these normally invisible standards become visible when one interacts with a culture where all of them seem to have been arbitrarily tweaked. A difference of standards can keep a friend in England from being able to send

a VHS tape to a friend in America, while another set of standards allow the same pair to exchange homemade DVDs but not store bought ones. Some standards are enforced by legal fiat (drive on the correct side of the road), and others are merely social (shake hands when meeting someone). Recently there have been a spate of books telling us the history of humble objects like the pencil or the screw, so that the curious reader can see how these everyday things achieved their current standardized form.[2] Though in some cases these sorts of standards may be merely arbitrary, nevertheless, it is still important for the functioning of society that they exist in some form and preferable that they exist in certain optimized forms rather than other suboptimal ones.

The standards of interest to this essay are primarily those in the computing field. As a simple example, when one is browsing the internet on a Dell computer using Internet Explorer in Windows XP, every part of that experience—from the image and text files on the internet to the network hardware that delivers it to the computer to the hardware inside the computer to the interaction between the software and its operating system—can be considered to fall under some sort of standard. In particular, WinAPI, ASCII, HTTP, JPEG, HTML, Ethernet, IDE, VGA, and EnergyStar are just some of the many standards that are vital to the coordinated activity of the machine. These standards were created and implemented by human designers, engineers, and programmers, and as such though they are technological standards, they still fall under the rubric of social standards as well.

By their nature, all computer programs can be seen as implementing a standard in at least the trivial sense that the program was written in a particular programming language in order to create a program that allows certain specified modes of interaction. In addition to this, most computer programs also have to be able to implement a standard in the sense that they need to be able to open files or transmit other data either

---

[2.] See: *One Good Turn: A Natural History of the Screwdriver and the Screw* by Witold Rybczynski, *The Pencil: A History of Design and Circumstance* by Henry Petroski, etc.

locally or over a network, and these files and their modes of transmission must proceed according to previously specified external standards. Some of these standards arise through the forethought of technical bodies, such as ANSI (American National Standards Institute), IEEE (Institute of Electrical and Electronics Engineers), and ISO (International Organization for Standardization), but many other very important technical standards have arisen largely spontaneously through the joint efforts of programmers who documented a standard only after they had finished creating something to utilize it. The difference between these two methods of standards development will be explored in greater depth later.

II. Basics of Confucianism

This essay conforms to the philosophical reconstruction school of interpretation. That is, this essay is more concerned with what Confucius would say if he were aware of developments today than what he did say during his own lifetime. That is not to say that I discount his own understanding in any way, since understanding what he said then is an important part of understanding what he would say now. The point is that in cases of dispute about his actual meaning, this essay is more concerned with what would have been better for him to say rather than what is the most likely interpretation possible.

Central to even a basic understanding of Confucianism is the understanding of 禮, 義, and 仁. 禮 is usually translated as "ritual" or "ritual propriety." 禮 are those basic everyday actions whose repetition creates social order. Hall and Ames describe them as, "patterns of behavior initiated and transmitted in order to refine and enhance life in a community."[3] In Confucius' time, one bowed in certain circumstances, and today we leave tips at restaurants. Both are interactions that have become set social patterns through the repeated spontaneous expression of goodwill among human beings. Through the refinement of 禮 by repetition, we are able to express 義.

[3.] Hall, David L. and Roger T. Ames. *Thinking Through Confucius*. State University of New York Press, Albany, pp. 89, 1987.

義 is usually translated as "righteousness" or "appropriateness" but can also be translated as "meaning" or "significance." 義 constitutes what is proper to do in a given circumstance, but unlike some Western notions of morality, 義 is not a fixed thing existing independently of circumstances, but a personal criteria for what is appropriate that arises from circumstances. This is not to say that all standards are relative. An individual must cultivate him or herself in order to realize 義 in a given situation. It is just that 義 cannot be prejudged. Confucius describes himself in *Analects* 18.8 by saying, "I am different from [previous sages] in that I do not have presuppositions as to what may or may not be done."[4]

What keeps Confucianism from running off the rails into lawlessness is its grounding in 仁, translated as "humaneness," "benevolence," or "authoritative humanity." 仁 is a positive concern for other human beings that becomes a part of one's essential character through 禮. As Confucius says in *Analects* 12.1, "Through self-discipline and observing [禮] one becomes [仁] in one's conduct." To give a concrete example, saying "thank you" after a commercial transaction is just an arbitrary social ritual, but through practicing it, it is possible to cultivate one's sense of thankfulness until its genuineness can be felt by the listener. Doing this makes life concretely better for both the speaker and the listener, if only slightly. 仁 is an attempt to take these other centered customs and use them to make a genuine concern for others an integral part of the self.

The goal of the Confucian is to become 君子, a "gentleman" or "exemplary person." Such a person embodies compassionate authority due to a lifetime of cultivation, and as such is worthy and able to create and promulgate new social norms among the people.

[4.] Unless otherwise noted, all *Analects* quotes from: Ames, Roger T. and Henry Rosemont, Jr. *The Analects of Confucius: A Philosophical Translation*. Ballantine Books, New York, 1998.

III. Defense of the application of Confucianism to technical standards

The reader is entitled to here question the connection of all of this to the implementation of technical standards. What relationship do following rituals, creating a sense of the appropriate, being compassionate, and becoming an outstanding person have to programming a computer or writing a technical document? The rituals that Confucius emphasized were those centered around political and family life. How can they relate to these very different activities?

In part, a full defense of the relevance of Confucianism to technical standards stands or falls on the merits of the rest of this essay. However, before evaluating it, I think it can be shown that the concerns of Confucianism are not entirely alien to the technical domain.

First of all, as has already been emphasized, Confucianism is not merely a moral philosophy in the Western sense of attempting to legislate correct behavior during occasional moral events. Instead, the practice of 禮 with 義 is intended to inform one's whole life in the pursuit of 仁. For this reason, any human endeavor can be done in a Confucian manner. Technical fields also frequently take on a massive collaborative scope. Under those conditions, the Confucian teachings regarding the kind of cultivation necessary to lead are particularly instructive.

Moreover, in the area of technical standards Confucianism is especially relevant, because technical standards are always, on final analysis, designed for the benefit of humans. Even when a technical standard has a large number of arbitrary requirements that are imposed by the limitations of the hardware on which they will be implemented, nevertheless the technical standard can still be said to exist ultimately in order to fulfill some perceived human need. For this reason it is imperative that the standards be created with an eye to humaneness, so that the need is met without subjecting the standard implementor, creator, or user to unnecessary stress. While programmers may sometimes take a perverse pride in tackling obfuscated code contests,[5] under ordinary

circumstances a technical standard will be employed by other human beings in order to achieve a practical result. Accordingly, it is morally important to design in such a way to make the lives of those people less frustrating and more rewarding. If Confucianism is correct, then the best way to achieve harmony between those who create standards and those who implement them is to utilize 仁 in design by performing 禮 with 義.

Finally, Confucius as a highly practical philosopher was concerned with integrating musical instruments, horse riding equipment, archery tools, farming systems, and the other arts of his time organically into human life, not as ends in themselves but as means to human flourishing. In these more technologically saturated times, it is no less important for our more modern tools and systems to be closely examined so that we too can find the optimal means of using our tools to improve our lives. A famous anecdote from the creation of the Apple Macintosh recounts how Steve Jobs motivated his engineers to improve the start up time for the machine:

> One of the things that bothered Steve Jobs the most was the time that it took to boot when the Mac was first powered on. It could take a couple of minutes, or even more, to test memory, initialize the operating system, and load the Finder. One afternoon, Steve came up with an original way to motivate us to make it faster. …
>
> "You know, I've been thinking about it. How many people are going to be using the Macintosh? A million? No, more than that. In a few years, I bet five million people will be booting up their Macintoshes at least once a day.
>
> "Well, let's say you can shave 10 seconds off of the boot time. Multiply that by five million users and thats 50 million seconds, every single day. Over a year, that's probably dozens of lifetimes. So if you make it boot ten seconds faster, you've saved a dozen lives. That's really worth it, don't you think?"[6]

Obviously, not all products have the same impact as the original Macintosh, and even in such a case, effects are distributed rather than concentrated, such that no lives are actually "saved." Nevertheless, the impact of technological standards on our society is cumulatively great and hence worthy of careful study.

[5.] See: *The International Obfuscated C Code Contest*. `http://www.ioccc.org`.

[6.] Hertzfeld, Andy. "Saving Lives." *Folklore.org*. `http://folklore.org/StoryView.py ?project=Macintosh &story=Saving_Lives.txt`.

IV. Implementing a standard

A. Two models of development

Speaking broadly, there are two extremes in the method by which software is created. The first extreme is known as the "waterfall model." In this method, a group first meticulously maps out project requirements, then thoroughly specifies the design of finished product, and only then commences actually implementing the design. After the implementation is finished, it is evaluated for bugs then released. Each step of the process must be completed before moving down to the next level, and no step is repeated or returned to. This gives the impression of the project as a series of waterfalls that cascade downwards. The second extreme is known as the "iterative model." In this method, design requirements are first sketched out very broadly but implementation begins as soon as possible. Once a little progress has been made, the implementers then check for bugs and stop to consider what they have done so far and how they should proceed given what they have learned from the implementing process. They refine their goals slightly, sketch out a new design, and then return to implementing it. This process ideally continues cyclically until both the client and the implementers are satisfied that there are no longer any requirements worth adding to the project and those requirements previously proposed have all been implemented satisfactorily.

The waterfall model tends to be more popular for government contractors[7] and mature industries. In those situations, it is of primary concern that the design process first come up with a reasonable projection of the costs that the implementation will incur. Once these costs are understood, then funds can be appropriated as needed or the project cancelled if it will be too expensive. A complete list of requirements needs to be made, so that when the project finishes it will be absolutely explicit whether the implementation group has lived up to their contractual obligations. Similarly, forethought is needed for how the requirements will be implemented, so that a time table can be draw up for

7. "Parametric Cost Estimating Handbook." *Cost Estimating Web Site JSC*. `http://www1.jsc.nasa.gov/bu2/PCEHHTML/pceh.htm`.

each stage of the process, and no unexpected difficulties can arise late in the process that require a time consuming redo of some aspect of the project.

While at first blush, the waterfall model may seem to be the most logical way to tackle a problem—namely, define a problem, break it into steps, and follow them out—in practice it runs into numerous difficulties. One difficulty is that in many situations, it is not clear what problems need to be solved and what problems can be left unsolved for now due to their complexity. A more fundamental issue is that for a good computer scientist, defining how the system should work in great enough detail is equivalent to actually programming it. For example, if a client were to say, "I want a database for my warehouse," that information is not deep enough for the waterfall model to make an accurate projection of the complexity of the project. On the other hand, once the client has made a list of every possible object that the database might been called on to catalog, what properties each of these objects can have, how these properties are interrelated, how the entry forms for the various objects should appear, and how access to the database should be restricted, the actual implementation process is comparatively trivial. The difficulty is that, as a non-specialist, the average client is no position to make such a thorough list of requirements, thus their initial specification if followed to the letter through will be an inadequate solution to the problem. As Fred Brooks famously argued in his manifesto on software design, "No Silver Bullet," "I believe the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems."[8] In other words, the central problem is the creation of useful designs, rather than their implementation. Accordingly, Brooks goes on to endorse iterative refining of requirements and rapid prototyping.

[8.] Brooks, Frederick P. Jr. "No Silver Bullet: Essence and Accidents of Software Engineering." Proceedings of the IFIP Tenth World Computing Conference, pp. 1069-1076, 1986.

In the *Analects* 15.31, Confucius is quoted as saying, "Once, lost in my thoughts [思], I went a whole day without eating and a whole night without sleeping. I got nothing out of it, and would have been better off devoting the time to learning [學]." Waterfall methodology falls into the same trap. They spend much time reflecting [思] on the future of the implementation. However, the only way to find out about certain aspects of the design should be implemented is to try implementing them and see what happens [學] then go back and change the system to match what they have learned. For computing, many bugs will remain hidden until actual testing brings blind spots in the design out into the open. If the software is complex enough, no one human mind can follow all of the twists and turns of the logic as it is executed, so preplanning is ineffective. Waterfall processes were invented to deal with the construction of physical systems in which the cost of going back after making a mistake is very high, so everything must be planned out in detail in advance. However, for computing, the cost of going back to an earlier stage is low, since actual code is immaterial and only the time wasted matters. For this reason, it is better to start testing faster, so that more bugs are revealed earlier in the development process.

The main problem of the iterative model is the question, how can you start to build something when you are not sure what you are making? Certainly, this leads to deep problems with scheduling. If you do not know what you are doing, it is difficult to know when you will be finished. However, the search for something that we do not know for certain exists is not unique to software creation. The Confucian pursuit of 義 is similar in certain respects. As mentioned, 義 can be translated as appropriateness, righteousness, significance, etc. Hall and Ames explain that it "has normative force without… constituting a norm."[9] Put simply, 義 is that which is cultivated through 禮, ritual action. For an iterative design process, the ritual action must be the process of evaluating the

[9.] Hall and Ames. pp. 102.

progress of the implementation as it is created, bearing in mind *Analects* 15.30, "Having gone astray, to fail to get right back on track is to stray indeed." That is, mistakes in the pursuit of 義 are acceptable, but one must learn from those mistakes in order to make use of them. Or to quote *Analects* 15.18, "Having a sense of appropriate conduct (義) as one's basic disposition (質), developing it in observing ritual (禮), expressing it with modesty, and consummating it in making good on one's word (信): this then is an exemplary person (君子)." These qualities are also the makings of an exemplary hacker, in the positive sense of that term. By working iteratively toward the goal of quality as the culmination of good practice, the hacker makes good on his or her word by fulfilling the expectations of the client, delivering no less than what was promised.

B. POSIX support in Windows NT as a paradigmatic failure of antagonistic implementation of rules based standards

While we have seen the utility of the iterative model in implementing a particular program, we have not explored its role in the creation of cross application standards. Unix is a family of operating systems that evolved over several decades on a number of different kinds of computing hardware by many distinct groups of programmers and corporations. Originally developed at Bell Labs in the early 1970s, it has been rewritten from scratch numerous times since then, each time achieving a little more polish and bringing along the positives of previous implementations while removing negatives and adding new innovations. Neal Stephenson considers it to be the "Oral Tradition" of computer science. As with a collection of folk tales, Unix continually gets better in the retelling:

> After this kind of thing has happened several hundred or thousand times, the hacker understands why Unix is the way it is, and agrees that it wouldn't be the same any other way. It is this sort of acculturation that gives Unix hackers their confidence in the system, and the attitude of calm, unshakable, annoying superiority… Windows 95 and MacOS are products, contrived by engineers in the service of specific companies. Unix, by contrast, is not so much a product as it is a painstakingly compiled oral history of the hacker subculture. It is our Gilgamesh epic.
>
> What made old epics like Gilgamesh so powerful and so long-lived was that they were living bodies of narrative that many people knew by heart, and told over and over again—making their own personal embellishments whenever it struck their fancy. The bad embellishments were

> shouted down, the good ones picked up by others, polished, improved, and, over time, incorporated into the story. Likewise, Unix is known, loved, and understood by so many hackers that it can be re-created from scratch whenever someone needs it.[10]

That Unix has been re-implemented so many times leads to a complicated question of identity. Namely, it is difficult to say what "is" or "is not" Unix in an absolute sense. Obviously, the legal trademarks that derive from the original Bell Labs operating system are still in existence and constitute one meaningful sense of what is a Unix. (Typically, this is called UNIX in all capital letters to distinguish the trademark from the concept.) However, a more useful meaning of Unix is a system capable of running programs originally designed for some other operating system that is also considered to be a Unix. The IEEE codified the POSIX (Portable Operating System Interface for uniX) standard for just such a purpose. Along with the documents that define what elements are necessary to meet the POSIX standard, the Posix Conformance Test Suite stands as the ultimate judge of whether an operating system is allowed to advertise itself as POSIX compliant or not, and thus to be considered a Unix in one sense. Roger Martin of NIST (National Institute of Standards and Technology) is quoted in the trade journal *Government Computer News* as saying, "If you run the test suite and pass everything, you're in business."[11] Thus, the judgment of the status of an operating environment as POSIX compatible or non-POSIX compatible can be judged programmatically.

In the early 1990s, the government had a rule in place that for certain tasks they would only purchase operating systems that were POSIX compliant. Microsoft naturally was upset by the rule, since their Windows NT operating system is the only modern operating system code base in wide use today that is not in some sense a Unix derivative. (There were more at the time however, including the OS of the Macintosh of that era.) Accordingly, they started a crash program to allow Windows NT to pass the Posix

---

[10.] Stephenson, Neal. *In the Beginning was the Command Line*. `http://www.cryptonomicon.com/beginning.html`.

[11.] McCarthy, Shawn P. "NT is Posix-compliant, GSBCA decides; ruling raises questions." *Government Computer News*. `http://www.gcn.com/print/14_23/31557-1.html`.

Conformance Test Suite. One internet user who claims to have worked with Microsoft in 1992 and paraphrases the project manager for the first meeting of POSIX implementation group as saying:

> "Before we proceed with this presentation, there's one thing I (Microsoft) want to make clear. The POSIX subsystem is a check box. We're only doing it to fulfill the requirement to have POSIX so we can get government contracts."[12]

In other words, Microsoft had no interest in implementing POSIX as a means of becoming compatible with the vast catalog of previous Unix systems or inheriting the best design attributes of previous implementations. They merely needed to be able to say "POSIX compatible" in order to win some valuable contracts. The program succeeded, but outside parties noted numerous questions surrounding the legitimacy of this pronouncement. In particular, for numerous tests, the Microsoft implementation was only partially complete. It was enough to pass the test, but not enough for practical use. From the previously mentioned trade journal article, "'It's not fair to say the OS had 'a true pass,' Martin said. 'I think the correct wording would be, "This test did not fail."'" In fact, the Microsoft implementation essentially ended the legitimacy of POSIX as means of distinguish Unix systems from non-Unix systems. Ironically, one of the best known implementations of Unix today, the GNU/Linux operating system, has not been put through POSIX testing, in part for this reason. Now that POSIX no longer has the meaning it once had, an important standard has lost its role in the computing ecosystem. How was the possible?

As Peimin Ni points out in "'Rules' with Exceptions", it is not enough to follow an ethical rule programmatically.[13] In this case, we can see that it is not enough to follow rules programatically even if the rules in question pertain to a program! The Golden Rule for example is a very commonly employed ethical rule. However, in the example

---

[12.] Username "Yagu." "I call bull hockey!" *Slashdot.org*. `http://slashdot.org/comments.pl ?sid=165237& cid=13787008`.

[13.] Ni, Peimin. *"Rules" With Exceptions: A Discussion On Confucian Ethics*.

of the masochist, it becomes clear that "doing unto others" involves more than what one personally wishes would be done to oneself. Similarly, a judge is not wrong to send a criminal to jail though that judge may wish to be set free were the judge on trial for committing the same crime. To counter these two examples, we can of course construct a revised version of the Golden Rule called GR–2 that says, "Do unto others as you would have them do unto you, unless you are acting as a masochist or a judge." However, is it really practical to try craft enough exceptions to deal with every possible exception to the Golden Rule in such an ad hoc manner? As in the example of the waterfall model, it is difficult to anticipate certain kinks in the rules before they are run into in actual experience. 思 (reflection) is not the best means for finding examples that contradict one's assumptions, because one is by definition inclined to accept beliefs that conform to one's biases. Furthermore, even if one undertakes an iterative process in order to find a superior version of the Golden Rule, how does one judge whether the version of the rule under consideration is superior or inferior to the one already in use? This supposes the preexistence of a higher ethical rule by which to judge the suitability of the proposed rule. However, if a higher rule were already concretely possessed, there would be no need to use the lower rule at all.

On top of this, it is fair to say that the original Golden Rule has a clear aesthetic superiority to our GR–2, because of its simplicity. By the time we iteratively hit on a variant of the Golden Rule that can practically deal with any situation in experience, GR–*n*, our new rule will no longer have the directness or clarity that made the Golden Rule appealing in the first place. In codifying POSIX, the goal was to programmatically define "Unix-ness" in the same way that the goal of the Golden Rule was to codify right-eousness programatically. In the given examples, it may seem clear that Linux possesses Unix-ness but Windows NT does not, and that a judge convicting a criminal is righteous but a masochist torturing others is not, but when we are pushed for an explanation as to

why we believe these things, all we can say is that these judgments comport more closely with our intuitions. The difficulty is that we have been treating these guidelines for being more like a Unix and being a better person antagonistically. To treat the Golden Rule antagonistically is to search for loopholes, that is, to search for a way to say, "Gotcha!" so that we can safely ignore the way that it highlights our hearts preexisting injunctions. The masochists or judges who find themselves unable to comply with the Golden Rule should strive to change their own natures, so that when they treat others gently or in accordance with the law, they themselves would also desire the reciprocation of that treatment. In this way, the Golden Rule has use as a moral standard for cooperatively cultivating the self, rather than as a rule to get out of. In the same way, POSIX has its positive use as a way of evaluating an operating system so that it can be brought into closer compatibility with previous Unix systems. The point of passing the test suite should not be to establish a system as nominally a Unix, but the point should be to make a system a *better* Unix. If Microsoft had treated the test suite as a 禮 (in other words, used it in the pursuit of the 義 of Unix-ness), it is possible that Windows NT could have become another legitimate Unix descendent set apart by different stylistic innovations. Instead Windows NT is an entirely unique operating environment—and thus Microsoft has been burdened throughout its history with the task of reimplementing from scratch certain features that are taken for granted in the Unix realm, particularly in the field of computer security.

The thoughtful reader will here be able to provide numerous other examples of system in which attempting to use a standard antagonistically has resulted in less than exemplary outcomes. To give a brief example, in the school system students are given grades in order to assess their learning, but soon begin to try to get the highest grades with the least amount of effort. This leads to a less desirable outcome for all parties involved. As Confucius says in *Analects* 4.12, "To act with an eye to personal profit will

incur a lot of resentment." The teachers are faced with antagonistic students, the students must compete for grades as proof of excellence to third parties, but those third parties find that they cannot trust the grades as a sign of anything other than the ability of the students to pass tests.

As advice to all standards implementors, I exhort them always to use programmatic standards as a spur to excellence in implementation rather than as a minimum threshold to cross. Use standards as a means of learning (學) how to make an implementation better, not as a simple test to pass. Obviously as in the case of Microsoft's desire for government contracts or the student's desire for a job, there may be temptations to treat the standard antagonistically, but where possible, implementors should resist these temptations, since giving in ultimately ruins the standard for everyone, much as POSIX conformance no longer implies Unix-ness and good grades no longer imply mastery of the subject matter. On a wider scope, however, a large share of the responsibility to ensure that standards are used positively rather than antagonistically ultimately rests on the shoulders of those who design and promulgate the standard, as we shall see in the next sections.

V. Promulgating a standard

HTML (HyperText Markup Language) was originally created by Tim Berners-Lee as a custom variant of an existing standard, SGML (Standard Generalized Markup Language), for use with the world's first web server and browser, which he also authored. In the mid- to late 1990s as the internet exploded in popularity, browser makers such as Mosaic, Netscape, and Microsoft began adding proprietary features to their implementations of the language so that their web browser would be able to display more compelling web pages than rival browsers. Around the same time, Berners-Lee and others worked to standardize what was in danger of becoming a proprietary format by

forming the World Wide Web Consortium (W3C). The Web Standards Project explains the message that they began to deliver in 1998:

> If Netscape and Microsoft persisted in building ever–more incompatible browsers, the cost of development would continue to skyrocket, tens of millions would find themselves locked out, and the Web would fragment into a tower of digital Babel. In fact, we said, it had already begun to do so. …
>
> Our message did not go down easy. As competitors, Netscape and Microsoft were disquieted by the notion that they should support the same, open technologies. As powerful companies, they were also quite naturally uncomfortable with the idea that anyone—including the people who actually built the Web—had the right to tell them what they should or should not do.
>
> Between 1998 and today, we persevered, and ultimately the browser makers listened. In fact, engineers at both companies agreed with us, and privately delighted in our efforts to persuade the managers in charge to let Engineering do the right thing.[14]

How was this possible? How could the web standards movement coerce browser makers to cooperate when the W3C has no legally binding authority over anyone? *Analects* 2.3 suggest a means by which disparate elements can be coaxed into cooperation:

> Lead the people with administrative injunctions and keep them orderly with penal law, and they will avoid punishments but will be without a sense of shame. Lead them with excellence and keep them orderly through observing [禮] and they will develop a sense of shame, and moreover, will order themselves.

If the W3C had taken an antagonistic stance toward the browser makers and tried to regulate the behavior of others through injunctions and rules, the result would have been that they ignored or resisted the W3C's recommendations, or at best, they gave the appearance of cooperation while secretly undermining the process through the exploitation of loopholes. Instead the W3C invited interested parties to participate in the standards creation process, and as a result, versions 5 and 6 of Microsoft's Internet Explorer were among the most standards compliant browsers at the time of their release.[15] Engineers working with the W3C were convinced that cooperating was, in the Web Standards Project's words, to "do the right thing." This meant they felt a sense of shame about

---

[14.] "History of the Web Standards Project." *The Web Standards Project*. `http://www.webstandards.org/about/history/`.

[15.] Unfortunately, after releasing Internet Explorer 6, Microsoft essentially dropped out of browser development from 2001 until 2005, leading it to fall behind its competitors in standards adherence.

doing otherwise, and as a result of this feeling, they pressured their companies into working together, even though they had legitimate competitive reasons to fragment the market in order to gain greater share. This sense of "shame" is to be distinguished from a mere sense of "guilt." To explain this, it is worth quoting *Analects* 12.13, "In hearing cases, I am the same as anyone. What we must strive to do is to rid the court of cases altogether." Confucius felt that interpreting rules to tell if they had been breached was a job anyone could do. A more demanding job is to engineer the situation so that people are no longer inclined to break the rules. Note that Microsoft was under investigation by the U.S. Justice Department during part of the late 1990s. It is likely that the W3C could have pressed the government to include standards conformance as a punitive measure of the eventual settlement. However, this was not pursued, and I would argue had the W3C done so, it would have been counterproductive as it would have caused Microsoft to approach web standards antagonistically, as they did the POSIX standard. The key to the success of the W3C was to create shame among engineers for failing to do the right thing, not trying to pronounce their guilt when they went astray. If the W3C had set up tribunals, they could have easily "convicted" Microsoft or Netscape of various counts of improper HTML rendering. However, such a system would impart only a sense of guilt after the fact, not a sense of shame. Guilt is the dread of being caught. Shame is the disappointment in oneself for not being better. Guilt avoids the punishment but ignores the reason that the punishment was created. Shame chides a person from the inside to strive for excellence in the future.

Since 2001, the struggle for web standards has revolved less around browser manufacturers than web page designers. (Though there is still much room for improved standards compliance among web browsers.) The article which signaled this turn was "To Hell With Bad Browsers" on A List Apart, a site that describe itself as "For people who make websites."[16] The premise of the article was that since the web browsers used by the majority of the online public then supported basic standards, it was imperative

for web designers to design their web pages to conform to those same standards, so that… So that what exactly? A careful reading will show that the exact benefits of writing one's web page according to standards rather than just hacking together a web page that works are never explicitly stated in the article. It is just assumed to be self-evident that any web designer would prefer, all things being equal, to design a standards compliant web page. And it is true. Of course, there are ancillary advantages to using standards, such as that a standards compliant web page can easily be transformed to use in different environments, can be relied on to render consistently in future browser upgrades, and often are smaller (and thus faster to download) than their minimally designed counterparts. However, many of these advantages can be obtained without making the extra effort needed in order to achieve 100% standards compliance. Indeed 100% compliance presents an almost unreasonable burden on the designer. If the page renders quickly and correctly for the end user, why should the developer care that the code behind it is imperfect in a technical sense? The bugs are hidden from the casual web surfer, and only the most dedicated of fellow designers will ever look behind the page to its source code in order to see if there are any "faults." For instance, technically, when linking to a page containing an ampersand (`&`) in the URL, one should not write a link to that URL directly. Instead one should replace the ampersand with `&` and use that as one's link. However, all browsers will automatically realize that a link with an improperly positioned `&` in it should be rendered as though it were `&` and will direct the user appropriately, as though the designer had included an `&`. Thus, for whose benefit is the developer working when he or she ensures that all links have the ampersands properly encoded? I contend that the primary reason for using standards is unstated, because it does not need to be stated. Any web designer who is aware of the requirements for

[16.] Zeldman, Jeffrey. "To Hell With Bad Browsers." *A List Apart*. `http://alistapart.com/articles/tohell/`.

a standards compliant page will naturally see such pages as superior, even if he or she resists taking the effort to create them and calls them needlessly fastidious.

Notice also the method by which A List Apart announced the importance of using web standards in the article: they showed their commitment to their own message by redesigning their web page to be fully compliant with web standards. In addition, sites like CSSZenGarden.com were created to excite web designers by showing numerous examples of the kind of web pages that can be achieved with standards based layout. In this way, the web standards movement followed *Analects* 12.18, "If as a leader you were without any improper desires, ordinary people would not steal, no matter the amount to be gained."[17] A Confucian leader sets an example through virtuous conduct that naturally inspires others to attain the same level of accomplishment. The web standards movement succeeded in large part through a process of showing examples of the beauty of designing with web standards. In the computer industry, using your own products internally is sometimes jokingly called, "eating your own dog food." The saying is jocular, but the meaning is clear: if a software developer will not use his own products, neither will users find it acceptable. A Confucian leader recognizes this, and never gives others "dog food" she would find unworthy of herself. That is what is meant by 恕, which is sometimes called the negative Golden Rule. It means never to do to others what you would not have done to yourself. If "To Hell" had been a harangue for web standards delivered on a then typical non-standards compliant web page, it would have been the height of hypocrisy. Other web designers would have turned away from it in disgust. Instead, A List Apart lead by demonstrating the beauty of excellence and explaining how they had created their new, standards compliant page.

As a result, just being aware of the existence of web standards causes the development of a feeling that can only be called shame at non-compliance. Knowing the rigorous

---

[17.] Unlike most *Analects* quotations, this is my own translation.

beauty of a standards compliant page, one cannot make a page that fails the W3C's validation tests without feeling a twinge of disgrace. Speaking as an amateur web designer myself and from my experience talking to friends who are professionals, just knowing about the existence of tools for validating the standards compliance of my web page caused me strive to go back and tweak my code to conform to the standard, and for no other reason than that I could know that I developed my page properly and be without a feeling of shame for any invisible technical faults it would otherwise have had.[18] The W3C's validation tools are used as a 禮 to help approach 義, rather than as a rule to try to skirt around. In this manner, an esteem for web standards has spread across the web, and in complete disregard to their practical necessity for the end user, new web browsers like Firefox trumpet their superior standards compliance. All of this rests on the hard work of individuals like Jeffery Zeldman, Dave Shea, and Eric Meyer who have tirelessly championed web standards to developers, and in doing so created a situation directly reflecting the aforementioned *Analects* 2.3.

VI. Creating a standard

In the last section it was shown that in order to ensure the adoption of standards, encouraging an inward sense of shame through the self-evident excellence of one's standard is superior to merely trying to legislatively declare who is or is not in compliance with the standard, because an overly legislative approach leads people to avoid punishment but ignore the source of the rule that guides the punishment. Next, we will investigate the process by which such standards are created.

In a famous passage, *Analects* 13.3, Confucius was asked by a disciple what his first order of business would be if he were to govern a state. He replied, 正名, meaning roughly "make right the names," "insure that names are used properly," or "rectify

[18.] On request, I can supply the transcript of an instant messenger conversation predating my education in Confucianism between me and a friend in which we both agree to try to use standards on our web pages out of a sense of shame at doing otherwise.

the names." His disciple was somewhat incredulous and asked, "Would you be as impractical as that?" Confucius strongly rebuked his disciple and explained that proper nomenclature is the basis of language and that language is central to taking care of things. The role of standards organizations can be seen primarily as a kind of making right of names. The standards of the W3C, for example, define what a "web page" is. Without some individuals working to keep names right, it would not be possible for users of technology to depend on the interchangeability and interoperability of technologies. I might say that I will deliver your computer a "web page," but send you a document formatted completely incomprehensibly! However, at the same time, it is not simply possible for definitions of names to be imposed by external fiat. Confucius himself was well known for his love of the *Book of Songs* (詩經), a collection of poetry that had been passed down and refined through the centuries which tradition holds he compiled and edited. In *Analects* 17.9, he instructs his students to study the poems, because from them, one can "be aware of many things, the names of birds and beasts, grass and trees."[19] These names were not Confucius' creation but the creation of previous generations. However, through awareness of these names, one is more able to understand the natural world, hence their importance.

The Internet Engineering Task Force (IETF) is an organization with a great deal of respect for "names" that preserve tradition. The organization has risen to prominence with the triumph of its TCP/IP standard and other protocols, and today serves a crucial role in the creation of the bedrock standards for data transmission over the internet, but it has not become an organization that thinks itself capable of lording authority over others. As they describe themselves on their "The Tao of IETF" page:

> In many ways, the IETF runs on the beliefs of its members. One of the "founding beliefs" is embodied in an early quote about the IETF from David Clark: "We reject kings, presidents and voting. We believe in rough consensus and running code". Another early quote that has become a

[19.] Unlike most *Analects* quotations, this is my own translation.

> commonly-held belief in the IETF comes from Jon Postel: "Be conservative in what you send and liberal in what you accept".[20]

The IETF was originally begun as a small group of government employed researchers who met merely to discuss how to make the internet better. From this origin, it is natural that their preference has always been for new standards that integrate compatibly with old ones, rather than radical breaks with the past. This recognition of the past, even in a fast moving field like computers, is one of the IETF's many Confucian characteristics. Though it is one of the premier standards making bodies for the internet, it has no formal membership or enforcement powers. It is worthwhile to quote from their mission statement:

> Standard: As used here, the term describes a specification of a protocol, system behaviour or procedure that has a unique identifier, and where the IETF has agreed that "if you want to do this thing, this is the description of how to do it". It does not imply any attempt by the IETF to mandate its use, or any attempt to police its usage - only that "if you say that you are doing this according to this standard, do it this way". The benefit of a standard to the Internet is in interoperability - that multiple products implementing a standard are able to work together in order to deliver valuable functions to the Internet's users.[21]

It is remarkable how Confucian the language of this text is, given that it was written by those at least not consciously trying to emulate such language. The concern of the IETF is purely pragmatic and focuses ultimately on delivering value to Internet users (an instance of 仁 behavior) by non-coercively describing what a standard (a "name") means. Indeed, their commitment to non-coercion is so great that the standards they publish are known as "RFCs," Requests for Comments. These standards are made to promote excellence by iteratively receiving feedback from involved parties on working code, rather than by designing the standard in the ivory tower of 思 without 學.[22]

[20.] Hoffman, P. and S. Harris. "The Tao of the IETF." *IETF Tools Pages*. `http://tools.ietf.org/html/fyi17`.

[21.] Alvestrand, H. "BCP 95." *IETF Tools Pages*. `http://tools.ietf.org/html/bcp95`.

[22.] Bradner, S. "RFC 2026." *IETF Tools Pages*. `http://tools.ietf.org/html/rfc2026`.

VII. Conclusion

Throughout this document, we have emphasized the ways in which a Confucian understanding can help in the creation of technical standards for our computerized society. By focusing on learning rather than reflection, development can proceed more smoothly. By focusing on using standards as a guideline for improving oneself rather than a rule in which to look for loopholes, the outcome of development will be more appropriate. By leading through shame as a spur to excellence rather than guilt as means of avoiding liability, standards can be organically accepted by their intended implementors. By fixing the names with a pragmatic emphasis on adapting tradition, standards will be more robust and excellent. In each of these cases, an awareness of the teachings of Confucianism may not be explicitly necessary on the part of all the participants, but where such an awareness does exist, it will lead consistently to better implementation, better leadership, and better standards, whereas without such an understanding these results may come about only haphazardly if at all.