Digisprudence: the design of legitimate code

Diver, Laurence

Published in:
Law, Innovation and Technology

DOI:
10.31228/osf.io/nechu
10.1080/17579961.2021.1977217

Publication date:
2021

License:
CC BY-NC

Document Version:
Accepted author manuscript

Accepted for publication in 13(2) *Law, Innovation & Technology* (forthcoming)

Digisprudence: the design of legitimate code

Laurence Diver

laurence.diver@vub.be ▪ @laurence_diver

Abstract: This article introduces *digisprudence*, a theory about the legitimacy of software that both conceptualises regulative code's potential illegitimacies and suggests concrete ways to ameliorate them. First it develops the notion of *computational legalism* – code's ruleishness, opacity, immediacy, immutability, pervasiveness, and private production – before sketching how it is that code regulates, according to design theory and the philosophy of technology. These ideas are synthesised into a framework of *digisprudential affordances*, which are translations of legitimacy requirements, derived from legal philosophy, into the conceptual language of design. The ex ante focus on code's production is pivotal, in turn suggesting a guiding 'constitutional' role for design processes. The article includes a case study on blockchain applications and concludes by setting out some avenues for future work.

Keywords: code as law, normativity, legitimacy, computational legalism, affordance, design, blockchain, techno-regulation

Funding statement

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 788734).

COHUBICOL project (Counting as a Human Being in the Era of Computational Law), Law, Science, Technology & Society research group (LSTS), Vrije Universiteit Brussel, Belgium ▪ cohubicol.com

1. Introduction

The interplay between legal theory and design is a crucially important but under-theorised area. In this contribution I set out the idea of *digisprudence*, a descriptive and normative lens through which to consider the legitimacy of digital systems that govern behaviour. Why the name 'digisprudence'? In short, it aims to be to the design of software-based rules as *legisprudence* is to the design of legal rules. Where the latter seeks to justify the imposition by the state of freedom-limiting rules through the inclusion of formal features that legitimate their design,[1] digisprudence seeks to do the same for code-based normative orders, framing the legitimacy of a digital system in terms of the *affordances* described in section 3.3 below. The legisprudential principles respect the autonomy of the citizen in the face of a potentially overbearing state; by analogy the digisprudential affordances require that, in its structuring of our social and economic interactions, the material design of code also reflect that fundamental respect for autonomy, and indeed legal protection. The analysis thus aims to synthesise the practical issue of how the material design of code regulates with a legal-theoretical view of what constitutes legitimate regulation. It views software code[2] as an a-legal normative order that regulates human behaviour more effectively and directly than law does, can, or should, which for that essential reason ought therefore to be legitimated. If we expect behavioural governance in the form of laws to be legitimate, then we should expect the same from code that has, or aims to have, a similar or even stronger effect. This is the case even – and indeed especially – in the absence of positive laws that seek to guide code's design. As we shall see, lawmakers cannot possibly keep abreast of the normativities imposed by the vast, overlapping, and intersecting assemblages of code that structure and govern our lives, and so a basic, 'constitutional' level of design legitimacy is required, to ensure that such systems provide citizens with transparency, intelligibility, choice, oversight, and contestability.[3] The need for these design features is underlined by code's unique ability to regulate and, crucially, to *constitute* the terms of our behaviour. That ability flows from its characteristics of 'ruleishness', opacity, immediacy, immutability, pervasiveness and, perhaps most importantly, its production by private enterprise for commercial gain. These characteristics have

---

1 See L. Wintgens, *Legisprudence: Practical Reason in Legislation* (Surrey: Routledge, 2012) p. 297.

2 I refer to 'software' and 'code' synonymously in the rest of this article.

3 These are the digisprudential affordances, discussed below in section 3.

the potential to exhibit what I call *computational legalism*, an extreme form of unreflective rule-following, explored further in section 2.1 below. The implications of computational legalism are troubling, particularly in societies where we expect that rules regulating behaviour have some connection to democratic processes and ex post judicial oversight. The central claim, then, is that the standards that make legal norms formally legitimate ought *mutatis mutandis* to be applicable to other normative orders that enable and constrain behaviour, regardless of the positive content of those rules (although form will, of course, constrain content). Legal philosophy already provides tools for combatting unreflective legalism in the legislative realm; my contention is that these might be adopted and adapted for application within the realm of design, to achieve code that as far as possible avoids the characteristics of computational legalism.

In this article I begin by discussing the need for digisprudence, and why reliance on positive law to achieve 'compliance by design' is, by itself, insufficient. I then briefly sketch the elements of the theory's dialectical structure, which runs along the following lines. First, I describe from the perspectives of both design and legal theory how code is an a-legal normative order: how its designed materiality regulates and constitutes behaviour, and thereby how it is apt to reflect legalism, leading *a fortiori* to computational legalism. By way of example, I discuss blockchain applications as a technology whose characteristics have the potential to embody particularly strongly the characteristics of computational legalism. Next, I move on to the normative element of the theory, considering how legal-theoretical frameworks intended to mitigate or oppose legalism in legal domain – particularly Fuller's *internal morality of law* and Wintgens' *legisprudence* – might be used to combat legalism in the computational domain. This analysis provides the scaffolding for a bridge between legal theory and design practice, namely the affordance of legitimacy in code.

Ultimately, this kind of analysis requires a focus on the *production* of code, rather than just its operation. If lawyers are properly to grapple with the realities of how code regulates, we must embrace an analytical shift that takes into account not just its effects but also the practical realities of its production. This means we ought to consider the processes and tools that make up the 'legislature' where code is 'enacted', including, for example, software development methodologies and the integrated

development environments (IDEs) where the text of code is actually written. They are the point at which 'constitutional' protections can be built into the very fabric of the code.

Digisprudence is thus a contribution to the nascent 'design turn' in legal scholarship, according to which the material design of digital artefacts can (and should) be critiqued from the perspective of legal philosophy.[4] This might be achieved by translating notions of (legal) legitimacy into the language of design theory, while at the same time maintaining sensitivity to the unique technical and temporal aspects of the code development process. Although a full exposition of the theory will involve additional work, the article aims to set out its main contours and thereafter to identify avenues for that future research.

1.1 The need for 'digisprudence'

Why is this kind of analysis necessary? Goldoni expresses the problem well:

> Given that code is not exactly like law, it is difficult in the realm of code to adopt a kind of rule of law (or 'rule of code') approach. Yet, we have also seen that when a particular code is 'enacted', it may be too late to remedy the violation of certain rights. This is why the accent should be put on the moment of production, rather than on the moment of distribution.[5]

Code is like law in that it governs behaviour, but is also different from it in crucial ways. Like legislation, it is created to achieve some purposive end; it is 'enacted'. In operation, code is capable of violating rights, but its ontological characteristics are resistant to the constraints of legality and the rule of law.

---

[4] Other emerging work hints at such a 'turn', although principally in relation to privacy. See W. Hartzog, *Privacy's Blueprint: The Battle to Control the Design of New Technologies* (Cambridge, Mass: Harvard University Press, 2018); S. Gürses and J. van Hoboken, 'Privacy after the Agile Turn' in E Selinger, J Polonetsky and O Tene (eds.), *The Cambridge Handbook of Consumer Privacy* (1st edn., Cambridge University Press, 2018) See also P. Nemitz, 'Constitutional Democracy and Technology in the Age of Artificial Intelligence' (2018) 376 *Philosophical Transactions of the Royal Society A* 20180089, arguing for a design perspective on the effects that artificial intelligence is having on constitutional democracy.

[5] M. Goldoni, 'The Politics of Code as Law: Toward Input Reasons' in J Reichel and AS Lind (eds.), *Freedom of Expression, the Internet and Democracy* (Leiden: Brill, 2015) p. 128.

Lastly, we can observe that there are essentially two stages at which code can be assessed: ex ante at the point of production, or ex post at the point of operation.

When commercial enterprises create code, they create a-legal normative orders that are apt to replace or augment positive law as a primary source of behavioural regulation. Crucially, the private contexts within which this code is created are not subject to the legitimising procedural or formal standards of rule-making we might expect to find in constitutional democracies or the *Rechtstaat*. In the move from public to private rule-making, the effects of such code on behaviour therefore risk being illegitimate, whether or not this is intended.[6]

Even though code is not law *per se*, it can be useful to ask similar questions about the two normative orders, given they each govern human behaviour. In this respect, I take regulation to straddle two of the definitions provided by Black, namely (i) the 'promulgation of rules by government accompanied by mechanisms for monitoring and enforcement' (i.e. positive law and its institutional framework), and (ii) 'all mechanisms of social control or influence affecting all aspects of behaviour from whatever source, whether they are intentional or not'.[7] Code may not be law *per se*, exemplifying the second definition rather than the first, but it is precisely because of the ways in which it is *un*like law that this kind of analysis is made necessary: code can control behaviour more directly than can 'true' law, but simultaneously it lacks the latter's mechanisms of ex ante legitimation and ex post remediation, i.e. its 'legality'. In seeking solutions to this problem, it makes sense to look at the standards that other fields have considered in relation to the legitimacy of behaviour-governing rules – namely certain domains of legal philosophy.[8] In any event, code does not just regulate behaviour, but indeed constitutes

---

6 E. Bayamlıoğlu and R. Leenes, 'The "Rule of Law" Implications of Data-Driven Decision-Making: A Techno-Regulatory Perspective' [2018] *Law, Innovation and Technology* 1, p. 12.

7 J. Black, 'Critical Reflections on Regulation' (2002) 27 *Australian Journal of Legal Philosophy* 1, p. 11.

8 B.-J. Koops, 'Criteria for Normative Technology: The Acceptability of "Code as Law" in Light of Democratic and Constitutional Values' in R Brownsword and K Yeung (eds.), *Regulating Technologies: Legal Futures, Regulatory Frames and Technological Fixes* (Oxford; Portland, Or: Hart, 2008) p. 162. Asscher's earlier observation of this was insightful; see L. Asscher, '"Code" as Law: Using Fuller to Assess Code Rules' in E Dommering and L Asscher (eds.), *Coding Regulation: Essays on the Normative Role of Information Technology* (The Hague: TMC Asser Press, 2006) p. 86.

it, creating new conditions of possibility that frame our actions from the outset.[9] While it might seem odd to think of new possibilities as *regulating*, it is precisely in the defining of those possibilities and the network effects they inaugurate that the regulative dimension can again come in, as it were by the back door, to 'enforce compliance beyond anything that a written law can achieve.'[10]

Those who define those possibilities in code, and whose work is veiled by the private context of its production, ought to wield that power legitimately.[11] Generally speaking, rules that shape our behaviour ought to meet pre-existing standards of accountability, transparency, and contestability[12] (although, as we shall see, these standards require unpacking in the code context). If notionally sovereign state legislatures are bound by constitutions so that they cannot arbitrarily create rules in contravention of that ideal, then neither should this be possible for private enterprises in their creation of normative code that shapes behaviour in myriad and potentially arbitrary ways.[13] Whether or not the producer of the code *claims* authority to regulate is a moot point; in a sense, by creating code that structures its user's behaviour or mediates her experience of the world, its creator is staking a claim to at least some part of her autonomy. What matters in the end is how the code *in fact* regulates and constitutes behaviour, and in turn the exercise by the designer of power over citizens' behaviour once they have entered its sphere of influence. Of course, the citizen may never actually engage with the code, and so the analogy with law would to that extent appear limited, at least with respect to generality of application.[14] This will vary from code to code, however, since the general imposition of technological normativity varies considerably, due for example to network effects or the infrastructural nature of a software system. (It

---

[9] M. Hildebrandt, 'Legal and Technological Normativity: More (and Less) than Twin Sisters' (2008) 12 *Techné: Research in Philosophy and Technology* 169.

[10] *Ibid.*, p. 178.

[11] K. Yeung, 'Can We Employ Design-Based Regulation While Avoiding Brave New World?' (2011) 3 *Law, Innovation & Technology* 1.

[12] See for example J. Waldron, 'Can There Be a Democratic Jurisprudence?' (2009) 58 *Emory Law Journal* 675.

[13] See Hildebrandt, *supra* n. 9, p. 173 *et seq*. See also Goldoni, *supra* n. 5, p. 119. This understanding is similar to the concept of 'governance' in the regulatory literature – see C. Reed and A. Murray, *Rethinking the Jurisprudence of Cyberspace* (Cheltenham, UK: Edward Elgar Publishing, 2018) p. 140.

[14] This of course is Fuller's first principle of legality. See L.L. Fuller, *The Morality of Law* (Yale University Press, 1977) pp. 46–49.

would be difficult, for example, to argue that the rules defined in the Internet Protocol do not have general – if not completely universal – applicability, at whatever level of abstraction is relevant to a given individual.[15] This pervasive aspect of computational legalism is considered in more detail below.)

Despite Lessig's influential work on 'code as law' generating a sizeable literature, the design of code artefacts *per se* is a topic that has received relatively little attention in the legal world, especially with regard to the normative standards to which code designers, acting as 'quasi-legislators', ought to be held. Insofar as this topic has been considered, the emphasis has tended to be on the legitimacy of the *decision* to use code, as opposed to some other mechanism, to enforce a norm, and not on the materiality of the code itself (its normativity *per se*) or the mechanisms of its production. The designed materiality of artefacts is where the behaviour-shaping action takes place, and so it should not escape our analysis – a failure to do so will seriously compromise our understanding of the regulative landscape. The point of departure then is the focus on the form of the design 'rule' itself, and the question of whether that form is legitimate, viewed separately from either any higher-level decision-making that precedes its creation, or any substantive legal compliance requirements motivating that creation.

1.1.1 Digisprudence vs. 'compliance by design'

One could argue that it is properly the task of legislatures to develop rules aimed at designers, and that any failure of the latter to follow them ought to be dealt with according to orthodox judicial processes. The compliance of code with relevant substantive laws is of course very important, but on its own it overlooks (i) the *sui generis* nature of code and its unique ability directly to constitute and shape behaviour (i.e. computational legalism), and (ii) how the translation from textual norms to code-based norms invariably involves some level of modification of the rule.[16] The reality envisioned by legal text is not reflected in the reality constructed by code, partly because law itself is (and in many cases should

---

[15] A casual user may have less awareness of IP's strictures than a network administrator, but they are nevertheless both bound by them.

[16] L. Diver, 'Law as a User: Design, Affordance, and the Technological Mediation of Norms' (2018) 15 *SCRIPTed* 4; Goldoni, *supra* n. 5, p. 129; M. Hildebrandt and B.-J. Koops, 'The Challenges of Ambient Law and Legal Protection in the Profiling Era' (2010) 73 *The Modern Law Review* 428, p. 452 *et seq*.

be) vague or underdetermined,[17] and partly because the two ways of representing meaning (text and software code) are categorically different, both because language is underdetermined where code is precise[18] and because words require translation into behaviour whereas code is simultaneously 'words and actions'.[19]

The problems involved in creating representations of natural language rules in code are well known, but more relevant for the present argument are what van den Berg and Leenes call 'techno-effects', or the unintended or unexpected constellations of normativity that are continually brought into being by digital artefacts.[20] This notion of techno-effects points to the aggregate normativity of an artefact, whatever it may in fact be, observed separately from the intent of its designer. This contrasts with 'techno-*regulation*', where the focus is on the use of technology as a tool to effect *legal* norms, rather than on the altogether more pervasive techno-effects.[21] The difference is important, given that '[t]he "regulatory" potential of technologies – in the broadest sense – is tremendous, and daunting, indeed.'[22] The notion of techno-effects points to the difficulty in discerning the intention of the designer, as well as where the line lies between what they ostensibly intend their code to do and the additional,

---

17 Hildebrandt, *supra* n. 9, p. 177; C. Reed, 'How to Make Bad Law: Lessons from Cyberspace' (2010) 73 *The Modern Law Review* 903, p. 904; T. Endicott, 'Law Is Necessarily Vague' (2001) 7 *Legal Theory* 379.

18 Golumbia contrasts human language and programming languages thus: 'Programming languages, as Derrida knew, are codes: they have one and one only correct interpretation (or, at the absolute limit, a determinate number of discrete interpretations). Human language practice almost never has a single correct interpretation [… t]he use of the term [programming] *language* to describe them is a deliberate metaphor, one that is meant to help us interact with machines […]'. See D. Golumbia, *The Cultural Logic of Computation* (Cambridge, Mass: Harvard University Press, 2009) p. 19.

19 B. Latour, 'Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts' in WE Bijker and J Law (eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (MIT Press, 1992) n. 1. See also N.K. Hayles, 'Print Is Flat, Code Is Deep: The Importance of Media-Specific Analysis' (2004) 25 *Poetics Today* 67. This is not to deny the performative nature of speech acts within the law's form of life.

20 B. van den Berg and R.E. Leenes, 'Abort, Retry, Fail: Scoping Techno-Regulation and Other Techno-Effects' in M Hildebrandt and J Gaakeer (eds.), *Human Law and Computer Law: Comparative Perspectives* (Dordrecht: Springer Netherlands, 2013).

21 On techno-regulation, see for example C. Gavaghan, 'Lex Machina: Techno-Regulatory Mechanisms and Rules by Design' (2017) 15 *Otago L. Rev.* 123; R. Brownsword, 'Lost in Translation: Legality, Regulatory Margins, and Technological Management' (2011) 26 *Berkeley Technology Law Journal* 1321.

22 van den Berg and Leenes, *supra* n. 20, p. 83.

unintended normativity it nevertheless imposes: '[t]he affected individual cannot discern which part of the normativity (as could be inferred from the output) is intentional and which part is merely spin-off in the form [of] unforeseen or secondary effects'.[23]

As we will see later in the discussion of computational legalism, those architectural norms tend by nature toward fixed configurations of behaviour-shaping normativity, which are applied with unqualified force in every case where the necessary computational conditions exist, regardless of any other relevant consideration. A narrow focus on substantive 'compliance by design' is thus unsatisfactory, or at least insufficient, because it elides the very active role that designers play in the creation of such contingent normative 'reality' in and through the code that they produce.[24] To rely on substantive compliance would require an explosion of statutory rules to cover all the normative configurations that code facilitates – plainly an unworkable idea.

Instead of focusing on substantive compliance, then, the idea is to focus on notions of 'by design' that operate at a deeper level, providing universal design-based protection beyond compliance with a necessarily limited set of substantive legal rules.[25] What matters then is not just compliance with the substantive law, but how to implement within the fabric of the code the kinds of constitutional safeguard that institutional law is expected to provide, regardless of its posited content.[26]

---

[23] Bayamlıoğlu and Leenes, *supra* n. 6, p. 12. They refer to this phenomenon as 'normative opaqueness'.

[24] Diver, *supra* n. 16.

[25] Similarly, Hildebrandt's notion of 'legal protection by design' is built around both substantive compliance and more foundational abilities of the individual to resist and contest the code. See M. Hildebrandt, *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* (London: Edward Elgar Publishing, 2015) p. 218. Nemitz points to a similar goal in his reference to 'the principles of democracy, the rule of law and human rights by design' – see Nemitz, *supra* n. 4 *passim*.

[26] Koops, *supra* n. 8.

2. Code as an a-legal normative order

To further clarify the location at which digisprudence is located, we can visualise the normative relationships in the digital sphere like this:

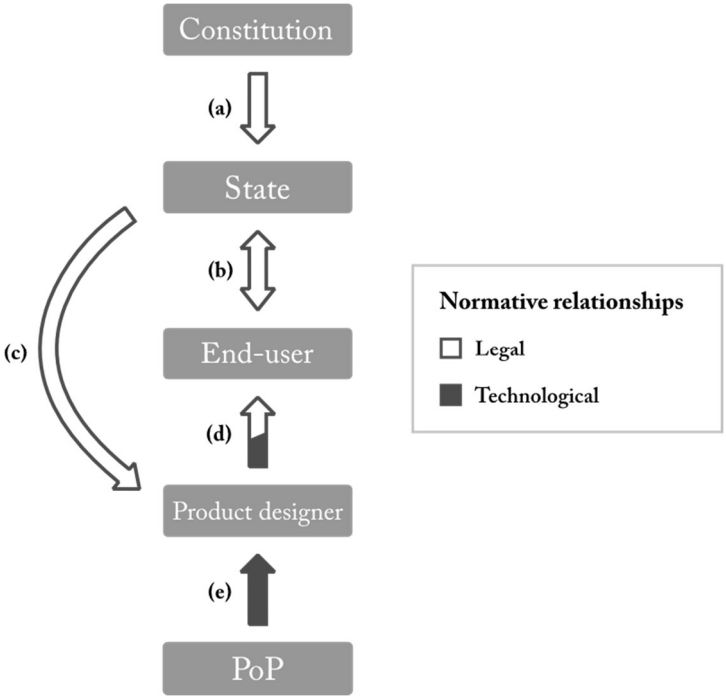

Figure 1. Relationships of legal and technological normativity

Relationship (b) represents the compact between the citizen and the state (the latter being bound by a constitution under relationship (a)), where the democratic process results in legal norms promulgated both through that relationship and relationship (c). The latter represents the traditional understanding of 'compliance by design', where state rules lay down requirements for code that designers must follow. In relationship (d), the product designer regulates via legal and/or technological normativity. The legal normativity in this relationship flows from public-order legislative norms on the one hand, or private-order contractual norms on the other. These are operationalised by (i) legal effect, which operates regardless of the code's design and is enforced in the orthodox way by the courts, (ii) the implementation of those norms in and through that design, or (iii) a mixture of the two. Scenarios (ii) and (iii) involve

code complementing law in what Leenes calls 'state-endorsed techno-regulation',[27] discussed above. Examples of this might include encryption used to implement data protection requirements, or a firewall preventing an employee from accessing social media in accordance with an employment contract. Relationship (e) concerns the 'programmer of the programmer', a concept I will elaborate on in section 3.3 below.

Crucially, while code can complement legal norms, it can also supplant them altogether.[28] Separately from these architectural implementations of public or private *legal* norms, code can implement normativity that is *purely* technological, which is to say rules that constitute and regulate behaviour outside any legal requirement, whether public or private, to do so. Here, norms are created, intentionally or otherwise, that shape user behaviour.[29]

Whether or not these assemblages of code rules aim explicitly to instrumentalise legal norms, their normativity exists separately from the legal system's normativity and its corpus of rules. But it is precisely this separateness that necessitates the present analysis. With legal norms, the legal effect of the data protection statute or the employment contract applies regardless of either instrument's implementation in or through code (subject, of course, to enforcement by a court, if that becomes necessary). But a corollary arises from this: it is precisely in the separateness of the two mechanisms of regulation that the *architectural* force of code, which implements some form of normativity, is able to supplant the normative infrastructure of law. Text-based laws are created in a world of legal-institutional speech acts[30] whose mode of existence relies on the delay, multi-interpretability, and ex post contestability of text as a medium.[31] So whereas text-based legal norms are in a sense passive, lying in wait to be interpreted, complied with, and perhaps enforced or declared null by a court, code simply goes ahead and enforces some configuration of regulative force that might bear no relation whatsoever

---

[27] R. Leenes, 'Framing Techno-Regulation: An Exploration of State and Non-State Regulation by Technology' (2011) 5 *Legisprudence* 143, p. 160

[28] Leenes terms this 'non-state techno-regulation' (*Ibid.*). See also Asscher, *supra* n. 8, pp. 67, 69.

[29] Goldoni, *supra* n. 5, p. 118.

[30] See generally N. MacCormick, *Institutions of Law: An Essay in Legal Theory* (New York: Oxford University Press, 2007) chs. 1–2.

[31] Hildebrandt, *supra* n. 25, p. 143 *et seq*.

to either the requirements of substantive law or any other external requirement. This becomes more problematic when we take into account the ontological characteristics of code that constitute its potentially legalistic nature (these are described in more detail below).

The statute that is improperly enacted or the contract that is improperly concluded are defeasible, i.e. presumed valid but nevertheless always open to challenge in, and reduction by, a court with the relevant authority.[32] The characteristics of code admit of no such possibility, within the domain of their execution: once its rules are 'promulgated', any 'illegality' has no bearing on the code's ability to execute and impose any latent technological normativity that it harbours. Put another way, the power need not be valid for it to be exercised.[33] From the point of execution onward the code will operate as though it was legitimately 'enacted', even where this is manifestly not the case. There is, therefore, a fundamental difference between invalid legal norms and 'invalid' code. With the former, the 'hermeneutic gap' that exists between text and action allows for a space in which validity can be considered, whereas with the latter there is no such opportunity, either to arrest execution or, in many cases, even to observe the invalidity.[34]

It is clear from the above that code is not the same as institutional law, but in observing this fact we must not reach the conclusion that it is not the proper concern of legal theorists. Indeed, it is precisely because of the ways in which code is *un*like law that it is necessary to consider the ex ante design choices that digisprudence is concerned with. Although code artefacts are to an extent facilitated by law (through contracts, intellectual property, and recursively via the legally-recognised infrastructure upon which the code runs), the relationship between the two is lopsided when it comes to behavioural regulation. The immediacy and normative power of code and the 'sovereignty' of the designer in defining its effects tip the balance dramatically away from law, rendering the latter 'a paper dragon in the age of the "digital tsunami"'.[35] Under code, the social and rhetorical power of legal fictions are swept aside by

---

[32] N. MacCormick, *Rhetoric and the Rule of Law: A Theory of Legal Reasoning* (Oxford ; New York: Oxford University Press, 2005) ch. 12.

[33] Cf. in law. See MacCormick, *supra* n. 30, pp. 158–165

[34] Diver, *supra* n. 16, p. 33

[35] Hildebrandt and Koops, *supra* n. 16, p. 440.

a 'digital virtuality'[36] that directly constitutes an empirical reality, shaped according to the whim of the designer, potentially with little respect for or even knowledge of the institutionality that law fundamentally relies upon.[37] Ex post adjudication is thus threatened by collapse into pre-determined obedience,[38] since the rule in the code also represents the immediate reality for the citizen, in many cases without any opportunity for re-interpretation or contestation. The materiality of the code is what matters, and it is in its profoundly ex ante nature that the problem lies.[39] Although to the citizen code may not consist of 'rules' in their commonly-understood textual form,[40] the purposive 'scripts' embodied in the design of software artefacts do channel our behaviour in ways that are to a greater or lesser extent predetermined, and pre-*envisaged*, by the rule-giver.[41] To that extent there is a strong parallel with top-down norms that succeed in ordering our behaviour in ways that differ from what we might naturally have done.[42]

I will return to the role of design later, but for now it suffices to note that the power that inheres in those who decide on the ends of those code rules is significant, outstripping the power that inheres

---

[36] C. Vismann and M. Krajewski, 'Computer Juridisms' [2007] *Grey Room* 90, p. 92.

[37] For an argument contrasting the materialities of text and code, see Hayles, *supra* n. 19.

[38] Z. Bańkowski and B. Schafer, 'Double-Click Justice: Legalism in the Computer Age' (2007) 1 *Legisprudence* 31, p. 48.

[39] As Longford puts it, code increasingly constitutes the very 'terms and conditions of existence and action.' See G. Longford, 'Pedagogies of Digital Citizenship and the Politics of Code' (2005) 9 *Techné: Research in Philosophy and Technology* 68, p. 71 See also A. Le Sueur, 'Robot Government: Automated Decision-Making and Its Implications for Parliament' in A Horne and A Le Sueur (eds.), *Parliament - Legislation and Accountability* (Oxford; Portland, Oregon: Hart, 2016) p. 201

[40] Koops and Leenes observe that designs which respect or disregard a particular value (privacy, in the case they consider), it can be viewed as the embedding of a 'rule' in that technology. See R. Leenes and B.-J. Koops, '"Code" and Privacy or How Technology Is Slowly Eroding Privacy' in E Dommering and L Asscher (eds.), *Coding Regulation: Essays on the Normative Role of Information Technology* (The Hague: TMC Asser Press, 2006) p. 191.

[41] M.T. Young, 'Artifacts as Rules: Wittgenstein and the Sociology of Technology' (2018) 22 *Techné: Research in Philosophy and Technology* 377. On inscription within design, see Akrich's classic account: M. Akrich, 'The De-Scription of Technical Objects' in WE Bijker and J Law (eds.), *Shaping Technology/Building Society: Studies in Sociotechnical Change* (MIT Press, 1992).

[42] This accords with Piekarski and Wachowski's notion of 'broad normativity'. See M. Piekarski and W. Wachowski, 'Artefacts as Social Things: Design-Based Approach to Normativity' (2018) 22 *Techné: Research in Philosophy and Technology* 400.

in other forms of private ordering that are, like law, fundamentally built around text that requires interpretation prior to its having any regulative force in the world. As Vismann and Krajewski put it,

> The quasi-sovereign power of the computer engineer's code stems from the ease by which posing, implementing, and applying a norm are achieved in technology compared with the cumbersome procedures that legal code must pass through. The swift effectiveness of a technological code, which cannot, when seen through legal eyes, appear as anything other than uncanny, renders any possible competition between law and computer pointless.[43]

Code is thus not law; it is instead both more, and less, than law. More, because of the instrumental power of design to constitute and regulate user behaviour. But simultaneously less, because it lacks the normative mechanisms that aim to keep its textually-bound sister in check.

Whereas traditional regulative norms derive their legitimacy in a constitutional democracy from their legality and their operation under the rule of law, code-based norms do not by necessity have the same (or any) democratic connection.[44] Whereas legal normativity *invites* the citizen to comply (she has always, at least notionally, the option to interpret the norm, contest it, or ignore it entirely), technological normativity can make compliance a necessity, either in the form of imposing a response to a given circumstance represented in the code's ontology, or by constituting at the outset all the courses of action that the user can possibly take. The fact that code is not 'true' law is not relevant to this central point, and indeed dismissing it on that ground is dangerous for all the reasons here discussed. As many have noted, normative systems exist in many contexts that have no explicit or implicit connection with the state.[45] Examples abound, from the rules regulating membership of a community association to the internal governance structures of a political party. What differs with code, however, is that the behaviour-shaping rules it applies are usually not open to scrutiny, and so the latent role of the law as the arbiter of last resort cannot be invoked. The hermeneutic gap between what the rule says and the

---

[43] Vismann and Krajewski, *supra* n. 36, p. 93 (my emphasis).

[44] Cf. L. Winner, 'Do Artifacts Have Politics?' [1980] *Daedalus* 121, p. 129

[45] See for example MacCormick, *supra* n. 30, ch. 1; Fuller, *supra* n. 14, p. 125 *et seq*.

behavioural possibilities it imposes is thus collapsed.[46] It is precisely because code lacks the checks and balances of legality but nevertheless has immense power to shape behaviour that it is necessary to instantiate some form of 'constitutional' protection in the materiality of its design.

2.1 Code is both more and less than law

2.1.1 Design and the constituting of behaviour

The article has so far discussed the differences between law and code, and why the latter ought to be of particular interest to legal theorists. I have claimed above that code is 'more' than law because of its capacity to constitute individual behaviour through its material design, as distinct from text-based normative orders that must, by definition, be interpreted by those to whom the norms are addressed before the latter can have any effect on behaviour.[47] How does code do this? Various concepts from design and the philosophy of technology can help us in framing the answer to this question, in particular the notions of affordance (which I employ again in the normative part of digisprudence, discussed below), inscription, and technological mediation. I have already mentioned inscription above, which is the notion of embodying in the design of an artefact a particular 'story' or script that dictates what the user ought and ought not to do.[48] Many of these scripts are so embedded as to become second-nature to the artefact's user – consider, for example, the steps involved in saving a document or typing a URL into a web browser. However natural or 'ready to hand' these code artefacts and their processes might appear to us, they are none of them a given; every one has to whatever extent been purposively

---

[46] Diver, *supra* n. 16, p. 33f

[47] For a discussion of the 'efficacy gap' in law see MacCormick, *supra* n. 30, pp. 71–74. I have previously referred to the 'hermeneutic gap' as an important difference between the normativity of text-driven law and the immediacy of code's impact on behaviour – see Diver, *supra* n. 16, p. 35.

[48] Akrich, *supra* n. 41, p. 208; B. Latour, 'The Berlin Key or How To Do Words with Things' in P Graves-Brown (ed.), *Matter, Materiality and Modern Culture* (London: Routledge, 2000).

designed.[49] Of course, this idea of channelling or 'tunnelling'[50] behaviour can be used for different ends, but whatever the choices made by the designer, these invariably leave out other possibilities that existed at the outset. What is left in will constitute the affordances of the artefact, or the ways in which it can be used by a particular user, given her characteristics and those of the code in question.[51] Although many affordances are incidental relationships between user and artefact, they are often consciously designed as features of the system, in which case they will (usually) be signified to the user.[52] A common example is a pad on the surface of a door that signifies the affordance of pushing (but not pulling). Another is the underlining of a hyperlink on a web page, distinguishing it from plain text.

In contrast to the enablement of behavioural possibilities that designed affordances and their signifiers represent, the concept of *dis*affordance points to the conscious and strategic choice of a designer to 'enforce or restrict certain user behaviour'.[53] This builds on Lessig's notion of 'architectures of control',[54] and is of course central to the claim made here about the (il)legitimacy of such technological normativity.

Code is designed with a particular class of user in mind, and so its (dis)affordances, inscriptions, and mediations are all fundamentally affected by the directed choices made by the designers who produce it. Although some forms of action are emergent or open to (re)interpretation or resistance on the part of the user,[55] it is nevertheless true to a greater or lesser degree that design choices embed

---

[49] Lessig hints at this truth when he notes that 'there is no choice that does not include some kind of building. Code is never found; it is only ever made'. See L. Lessig, *Code: Version 2.0* (New York: Basic Books, 2006) p. 6.

[50] On the latter see B.J. Fogg, *Persuasive Technology: Using Computers to Change What We Think and Do* (Amsterdam; Boston: Morgan Kaufmann Publishers, 2003) p. 34 *et seq*.

[51] D.A. Norman, *The Design of Everyday Things* (Cambridge, Mass; London, UK: MIT Press, 2013) p. 11.

[52] *Ibid.*, p. 13 *et seq*.

[53] D. Lockton, 'Architectures of Control in Product Design' [2006] *Engineering Designer: The Journal of the Institution of Engineering Designers* 28. See also D. Lockton, 'Disaffordances and Engineering Obedience' <http://architectures.danlockton.co.uk/2006/10/22/disaffordances-and-engineering-obedience/> accessed 18 June 2020.

[54] Lessig, *supra* n. 49, ch. 4.

[55] This relates to Ihde's notion of *multistability*, see his *Technology and the Lifeworld: From Garden to Earth* (Indiana University Press, 1990) p. 144 *et seq*.

'programs of action'[56] within the artefact, and so significant normative power inheres in those who make those choices. When a designer embeds (dis)affordances in the design of her artefact, she affects what it is possible to do with that artefact, either expanding or contracting those possibilities.

All of this points to the ways that designers fashion the geography of the artefacts they create, thereby controlling, at least to the extent it plays a role in her experience, that part of the user's mediated reality.[57] The extent to which that control is imposed will differ depending on the artefact and how far it exemplifies the elements of computational legalism, discussed in the next section.

2.1.2 From strong legalism to computational legalism

One of the central problematics of code from a legal-theoretical perspective is its 'ruleishness', meaning its application of defined rules in all instances where certain fixed conditions, predetermined in the code itself, obtain.[58] In the technical context this is of course a major benefit: even the most complex body of rules can be expected to execute in pre-determined ways under precisely-defined and controlled conditions, giving a predictability that has facilitated rapid incremental innovation in modern technological society.

In the legal context, however, the rote application of rules is undesirable, at least in a society built around the ideals of democracy and legality. Linked with the Kantian categorical imperative, legalism is the jural equivalent of software code's ruleishness. Although it has more than one form in the literature, the strong conception of legalism that is reflected in code's character is closely connected with more ideological forms of analytical legal positivism,[59] according to which rules and the strict adherence to them are the proper fundaments of social ordering. That the state defines what is legal is sufficient to legitimise the substance of the legal norms it chooses to declare; in constituting the field of

---

[56] Latour, *supra* n. 19.

[57] On the technological mediation of experience see P.-P. Verbeek, *What Things Do: Philosophical Reflections on Technology, Agency, and Design* (Penn State Press, 2005) ch. 3. See also Ihde, *supra* n. 55, particularly ch. 5.

[58] J. Grimmelmann, 'Regulation by Software' (2005) 114 *The Yale Law Journal* 1719.

[59] See Z. Bańkowski and N. MacCormick, 'Legality without Legalism' in W Krawietz et al. (eds.), *The Reasonable as Rational? On Legal Argumentation and Justification; Festschrift for Aulis Aarnio* (Berlin: Duncker & Humblot, 2000) and J.N. Shklar, *Legalism* (Harvard University Press, 1964) p. 7.

play (i.e. the legal system), the state legitimises *de facto* that which it consequently promulgates as the rules of the game (i.e. positive law). Constitutive facts in the form of natural laws, the social contract/constitution, or a mix of these, thus operate prospectively to legitimise any subsequent act of the sovereign.[60] The citizen is required to act unthinkingly – the rules are 'just there', and she need only act in accordance with their bare terms,[61] since by virtue of those constitutive facts the rules are 'imputed to [the people], as if they were its author.'[62] The legalistic outlook thus tends towards the 'narrow governance of rules, unleavened by [a] principled approach to interpretation.'[63] The apparent simplicity of this 'narrow governance' implies the risk of abuse: the prioritisation of heteronomy militates against critical reflection and the application of other normative principles that we might aspire to in a democracy (such as those of legality). The autonomy of the citizen to interpret the rules to which she is subject is seen as a crucial aspect of legality, without which those rules become 'implements of tyranny' and legalism a 'vice of narrow governance'.[64]

From this very brief summary of legalism one can begin to appreciate how code might exemplify these characteristics.[65] Even in the most tyrannical state there is space to interpret, and even to disobey – the hermeneutic gap between the text of a norm on the page and its translation into behaviour in the world makes this at least a notional possibility. In the environments where code is designed, however, the elision of that gap is not only easy to do but is entirely standard, not necessarily through malice or intentional obfuscation (although they are certainly a concern), but simply by virtue of the ontological characteristics of code, which by nature presents to the user norms that 'just are'. Even where the code does allow for choice via adjustable settings, the default configurations of code tend to be seen by users

---

60 Wintgens, *Legisprudence*, *supra* n. 1, chs. 5–6.

61 Z. Bańkowski, 'Don't Think About It: Legalism and Legality' in MM Karlsson, Ó Páll Jónsson and EM Brynjarsdóttir (eds.), *Rechtstheorie: Zeitschrift für Logik, Methodenlehre, Kybernetik und Soziologie des Rechts* (Berlin: Duncker & Humblot, 1993).

62 Wintgens, *Legisprudence*, *supra* n. 1, p. 208.

63 Bańkowski and MacCormick, *supra* n. 59, p. 194.

64 *Ibid.*

65 Bańkowski and Schafer, *supra* n. 38.

as 'a natural and immutable fact'.[66] The hermeneutic gap is thus closed, or at least significantly narrowed, because the 'text' of the 'rule' (the source code) constitutes directly the geography of the artefact: they are not just isomorphic, they are one and the same. Unlike institutional law, whose 'carrier' has hitherto been the inherently passive medium of text, software code allows us to, in Latour's words, 'conceive of a text (a programming language) that is at once words and actions'.[67] What we have with code, then, is potentially the apex of legalism: the normative collapses into the descriptive – what was once *requested* becomes simply what *is*, or what *will be*. There is no choice but to obey the rule as it is expressed by the designer, much less to view and contest it, since it by definition constitutes empirical reality.[68]

While code's *ruleishness* is a central characteristic of computational legalism, there are several additional characteristics that take it beyond even the strong legalism envisioned within the jural domain. The *opacity* of code makes its regulative operation inscrutable to the user. This is true in the course of the code's execution, but also in relation to the provenance of the code (who created it) and, by extension, its very purpose.[69] This characteristic is an amplification of the 'veiling', under strong legalism, of sovereign power, where the political reasons for a particular rule are deemed not to be the concern of the citizen – she must simply follow the rule without thinking about it.[70] These effects are compounded by code's *immediacy*, or the raw speed with which it imposes the predetermined

---

66 Goldoni, *supra* n. 5, p. 128. For an early recognition of code's legalism see J. Boyle, 'Foucault in Cyberspace: Surveillance, Sovereignty, and Hardwired Censors' (1997) 66 *University of Cincinnati Law Review* 177, p. 205.

67 Latour, *supra* n. 19, n. 1.

68 Bańkowski, *supra* n. 61; Bańkowski and Schafer, *supra* n. 38. Representationalism is a key element of the legalistic outlook – see L. Wintgens, 'Legisprudence as a New Theory of Legislation' (2006) 19 *Ratio Juris* 1, p. 5. Hildebrandt characterises this qualitative difference in her discussion contrasting architectural 'constitutive rules' (which define the scope of possible behaviour ex ante) and 'regulative rules' (which are akin to tradition laws that require to be interpreted to have any effect). See Hildebrandt, *supra* n. 9, p. 172 *et seq*.

69 Social networks, for example, are clearly much more than the benign facilitators of 'openness' they claim to be. See Asscher, *supra* n. 8, p. 84. See also Gürses and van Hoboken, *supra* n. 4, p. 584 and Z. Yu et al., 'Tracking the Trackers', *Proceedings of the 25th International Conference on World Wide Web - WWW '16* (Montreal, Quebec, Canada: ACM Press, 2016) which discusses the prevalence of hidden third-party trackers on websites.

70 L. Wintgens, 'Legislation as an Object of Study of Legal Theory: Legisprudence', *Legisprudence: A New Theoretical Approach to Legislation* (Oxford: Hart, 2002) p. 158; Bańkowski, *supra* n. 61, p. 46.

configurations of normativity that it embodies.[71] There, the user has little opportunity to arrest execution; unlike textual laws there is no inherent gap between the expression of the norm and its execution. Add to these code's *immutability* (the tendency, for either technical or economic reasons, for specific codebases to become settled in an artefact[72]) and its *pervasiveness* (that is, one piece of code regulating innumerable users via the same product, multiple artefacts regulating one user in the course of her connected life, or a combination of the two). Factoring in code's *private production*, all of this contributes to the problematic agglomeration of normative effects that mean that in many cases code is simultaneously more powerful and less adaptable than a law-system that is built around the characteristics of delay, flexible interpretation, and ex post remediation. Computational legalism thus tends towards a combination of brittleness, normative force, and lack of ex post control that is far in excess of even the most strongly legalistic of legal systems. Thus can code be said to be simultaneously *more*, and *less*, than law.

2.2 An example: blockchains

To give some practical context, this section will briefly analyse from a digisprudential perspective blockchain applications, or what are sometimes called 'smart contracts'.[73] Like digital rights

---

[71] On this point see Krajewski's discussion of code's 'closure' in M. Krajewski, 'Against the Power of Algorithms Closing, Literate Programming, and Source Code Critique' (2019) 23 *Law Text Culture* 119.

[72] This is particularly true for Internet of Things devices whose code may be costly or impractical to update – see W. Hartzog and E. Selinger, 'The Internet of Heirlooms and Disposable Things' (2016) 17 *North Carolina Journal of Law & Technology* 581, p. 597. For a real example see L. Franceschi-Bicchierai, 'Hacked Toy Company VTech's TOS Now Says It's Not Liable for Hacks' [2016] *Motherboard* <http://motherboard.vice.com/read/hacked-toy-company-vtech-tos-now-says-its-not-liable-for-hacks> accessed 10 October 2016. Lack of oversight is also a concern for blockchains built around the notional benefits of immutability. See P. De Filippi and A. Wright, *Blockchain and the Law: The Rule of Code* (Cambridge, Massachusetts: Harvard University Press, 2018) p. 201 and the case study below.

[73] This latter term is controversial. See for example E. Felten, 'Smart Contracts: Neither Smart nor Contracts?' <https://freedom-to-tinker.com/2017/02/20/smart-contracts-neither-smart-not-contracts/> accessed 2 January 2019. Buterin, the creator of the Ethereum blockchain and environment, discussed below, has lamented his adoption of the term and suggested that 'persistent script' would have been better: V. Buterin (*Twitter*, 13 October 2018) <https://twitter.com/VitalikButerin/status/1051160932699770882> accessed 5 January 2019.

management ('DRM') systems before them, blockchain applications are an explicit example of the transposition of rules (legal or otherwise) into the technological normativity of a digital artefact, in ways that guide or limit the user's behaviour.[74]

To assess blockchain applications from a digisprudential perspective we must consider the basic elements of their design.[75] Blockchains are public databases, stored on a number of computers ('miners') that together constitute a peer-to-peer network. (Although private blockchains do exist, they are generally used internally within an organisation and so they lack the focus on user behavioural regulation I am primarily concerned with.[76]) Adding new records, or transactions, to a blockchain requires consensus among the network's nodes, and so a new 'block' of data will only be added if a majority of the miners agree that its addition is in accordance with the set of rules (the protocol) that governs how that particular blockchain should operate.[77] Two prominent examples of blockchain protocols are Bitcoin,[78] the cryptocurrency and the original application of a blockchain design, and Ethereum,[79] the first protocol to enable sophisticated automation through its support for a variety of general-purpose programming languages.

A blockchain's protocol includes a mechanism for miners to reach a consensus on what should be stored on the blockchain, including metadata about transactions that have taken place and new application code ('smart contracts') that will be executed by it. The challenge of anonymous computers reaching consensus is an example of the 'Byzantine fault problem', where temporally and geographically

---

[74] The question of digital versus legal jurisdiction in blockchain applications has received some recent attention, albeit with limited engagement from the internal technologist's perspective. For a nuanced account from a legal perspective see for example F. Möslein, 'Conflicts of Laws and Codes: Defining the Boundaries of Digital Jurisdictions' in P Hacker et al. (eds.), *Regulating Blockchain* (Oxford University Press, 2019).

[75] For a useful, in-depth discussion see M. Pilkington, 'Blockchain Technology: Principles and Applications' in FX Olleros and M Zhegu (eds.), *Research Handbook on Digital Transformations* (Cheltenham, UK; Northampton, MA: Edward Elgar Publishing, 2016).

[76] For more on private blockchains see V. Buterin, 'On Public and Private Blockchains' (*Ethereum Blog*, 8 June 2015) <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/>.

[77] De Filippi and Wright, *supra* n. 72, p. 2.

[78] S. Nakamoto, 'Bitcoin: A Peer-to-Peer Electronic Cash System'.

[79] 'Ethereum White Paper' (*The Ethereum Wiki*, 22 August 2018) <https://github.com/ethereum/wiki/wiki/White-Paper>.

dispersed machines have copies of the blockchain, potentially in different states. For the blockchain to be workable, consensus must nevertheless be reachable despite this distributed nature. Blockchain protocols overcome this using a combination of public key cryptography and 'hashing'.[80] The former is a mechanism for conclusively identifying each node within the network by a unique public signature ('key'), while the latter is a method for generating a unique signature (hash) from any given volume of data, which in the case of blockchains is the totality of the data and code it contains at the moment a new block is added to the chain. At that point, the new block is assigned a hash generated from a combination of that block's new contents (data and application code) and the hashes of all the blocks already stored on the chain. The mathematical properties of the hashing algorithm mean that changing even one character anywhere within the entire data structure of the chain will result in a different hash being generated, and thus one can reliably deduce from two identical hashes that the two data sources from which they were generated are also identical. This in turn means that if two nodes in the distributed network derive the same hash from their copies of the blockchain, they can be sure that those copies are identical (the converse is also, therefore, true).

When a miner solves the computationally-intensive mathematical challenge specified in the chain's protocol – miners are incentivised to do this because finding the answer means a reward – the proposed solution is broadcast to the network for the other miners to verify. They independently generate a new hash from a combination of their local copies of the chain and the broadcast solution, and if it meets the requirements of the protocol's rules, each miner adds the block to their local copy of the chain and the winning miner receives its reward. The process then starts again for the next block. In this way the copies of the chain are kept identical and up-to-date across the many anonymous miners that store them. An important corollary of this consensus mechanism, particularly in its use of hashes that represent the historical state of the chain, is that once a block has been added its contents are de facto immutable[81] and verifiable by external observers.[82]

---

[80] Pilkington, *supra* n. 75, p. 228.

[81] *Ibid.*, p. 233f.

[82] *Ibid.*, p. 227.

Copies of the blockchain, including both its protocol and the data that it stores are replicated across the network, providing resilience and decentralisation.[83] This absence of a centralised authority controlling what gets added to the chain is part of the ideology behind the design of the technology: provided participants follow the rules contained in the protocol, they get the notional benefits of a tamper-resistant, 'trustless' database with no centralised controlling authority.[84]

2.2.1 Blockchain applications

At present blockchains are probably best-known as the foundation of cryptocurrencies, but another related application that is more problematic from a legal perspective are so-called 'smart contracts' (SCs). Blockchain platforms provide varying levels of sophistication. The Bitcoin protocol provides some very basic programming capabilities which can allow very limited blockchain applications (BAs) to be written. Some other platforms, known as 'sidechains', provide more sophisticated computation that runs separately from the primary Bitcoin blockchain but which rely on its relative stability as the ultimate store, or ledger, of transaction activity.[85] Yet others are completely separate from the Bitcoin blockchain, providing both an independent transaction ledger and a protocol that is specifically designed to provide a more sophisticated programming foundation for BAs. Of these 'smarter' platforms, Ethereum is the most prominent[86] and is therefore the focus of the following discussion.

Ethereum seeks to compliment the architectural characteristics of blockchains with a fully-fledged programming execution environment that combines 'Turing-completeness, value-awareness, blockchain-awareness and state'.[87] Programming languages have been created specifically for the

---

[83] De Filippi and Wright, *supra* n. 72, p. 2.

[84] For a critical analysis of the ideological underpinnings of blockchain design, see D. Golumbia, *The Politics of Bitcoin: Software as Right-Wing Extremism* (University of Minnesota Press, 2016) <https://www.upress.umn.edu/book-division/books/the-politics-of-bitcoin> accessed 13 February 2017.

[85] For a discussion of sidechains see T.I. Kiviat, 'Beyond Bitcoin: Issues in Regulating Blockchain Transactions' (2015) 65 *Duke Law Journal* 569, p. 604 *et seq*.

[86] For an empirical overview of the current major SC platforms, including Bitcoin and Ethereum, see M. Bartoletti and L. Pompianu, 'An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns' [2017] *arXiv preprint arXiv:1703.06322* <https://arxiv.org/abs/1703.06322>.

[87] 'Ethereum White Paper' *supra* n. 79

development of BAs on its platform,[88] allowing the combination of rich computation with the notional immutability, decentralisation, and 'trustless trust' described above. The code of Ethereum BAs can thus define complex conditions, execute arbitrary behaviours when certain conditions are met, maintain and monitor states over time, and record the outcomes in the underlying blockchain. All of this can take place automatically; once conditions are defined in the 'contract' it remains 'live', awaiting the appropriate change(s) in external conditions to trigger the rules it contains. In this sense BAs, then, are not passive textual instructions on what the 'contracting' parties ought to do, but rather they are 'like "autonomous agents" that live inside of the Ethereum execution environment, always executing a specific piece of code when "poked" by a message or transaction'.[89] The notional result then is a combination of software's immense plasticity with the inherent anti-plasticity, or 'persistence', of blockchain architecture.

Multiple BAs can be bundled together by a central business logic (itself written in code and stored on the blockchain) to create a 'distributed autonomous organisation' ('DAO')[90] which can operate without human input.[91] The code of such an 'organisation' could, for example, require a majority vote from its (human) members as a condition of another BA being executed (for example purchasing some asset on behalf of the DAO). Again, the decentralised and 'trustless' nature of blockchain design obviates the need for a trusted centralised authority (i.e. a traditional board or committee), and so notional governance of the organisation can be achieved even where the membership is geographically dispersed, or even unknown.[92] BAs can also consult external sources of data, known in this context as 'oracles',[93] executing code when predetermined external condition is met, for example a share price

---

88 For example Solidity, Serpent, and LLL.

89 'Ethereum White Paper' *supra* n. 79

90 V. Buterin, 'DAOs, DACs, DAs and More: An Incomplete Terminology Guide' <https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/>.

91 *Ibid.*

92 A. Wright and P. De Filippi, 'Decentralized Blockchain Technology and the Rise of Lex Cryptographia' (Rochester, NY: Social Science Research Network, 2015) SSRN Scholarly Paper ID 2580664 pp. 15–16 <https://papers.ssrn.com/abstract=2580664>.

93 Cardozo Blockchain Project, '"Smart Contracts" & Legal Enforceability' (New York: Benjamin N Cardozo School of Law, 2018) p. 6.

gaining a certain value. This might then result in real-world effects through the BA's automated interaction with the open interfaces (APIs[94]) of other services. When coupled with a cryptocurrency, it becomes possible to effect automated commercial transactions, even using APIs to automate the involvement of human actors, such as those working in the so-called 'gig economy',[95] or physical devices, such as drones, to, for example, deliver goods.[96]

2.2.2 The computational legalism of blockchains

The normative power of such code is intuitively appreciable. When specific conditions that are computationally represented (and *representable*) are met, the code self-executes according to its internal logic, and the outcomes are enforced regardless of any (relevant) external circumstances or considerations. With the outcomes of the code's execution being stored in the underlying blockchain alongside the code itself, what this means is both its logic and its results are immutable once they are 'enacted', executed, and stored. Thus code, in a very real and legally-significant sense, becomes 'law', through the 'collapsing [of] contract formation and enforcement into a single instrument'.[97] This coincidence of form and substance means that when a smart contract is executed, the material effects of that execution are governed by the dictates of pure code, regardless of any ambiguity or subjective understanding that might exist in the minds of the human individuals involved.

Code becomes at once rule and reality, with the normative *ought* collapsed into the descriptive *is* (or, rather, *will be*). This is the ruleish aspect of blockchain architecture, embodied in the applications that run atop it. Between the bright lines of ruleishness and the ex ante fixity of immutable blockchain

---

94 Application programming interfaces.

95 Buterin describes a DAO as 'an entity that lives on the internet and exists autonomously, but also heavily relies on hiring individuals to perform certain tasks that the automaton itself cannot do'. See Buterin, 'DAOs, DACs, DAs and More', *supra* n. 90.

96 De Filippi and Wright, *supra* n. 72, p. 156. For a recent real-world example of the latter, see J. Perez, 'XYO Game-Changer: We've Executed a Smart Contract With a Drone!' *Medium* (21 November 2018) <https://medium.com/xyonetwork/xyo-game-changer-weve-executed-a-smart-contract-with-a-drone-4deb414af67b>.

97 K.E.C. Levy, 'Book-Smart, Not Street-Smart: Blockchain-Based Smart Contracts and The Social Workings of Law' (2017) 3 *Engaging Science, Technology, and Society* 1, p. 3.

code, we see the need for the affordances of *choice*, *transparency of operation*, and *oversight:* choice for the user to have some role in what normative outcomes the code imposes at runtime, transparency about what the code is doing as it does it (which also implies *delay* to facilitate intelligibility), and oversight for those who might have an interest in altering that normativity ex post. The latter will include, at least, the designers of the code who, under certain blockchain configurations, might in effect be 'locked out' of making changes after-the-fact because of the code-imposed requirement to hold a majority stake in the distribution of mining power on the chain. In some circumstances such a majority might be impossible to achieve in practice.[98] It will also include the courts who, under the requirement to afford *contestability* in service of the rule of law, must ultimately be empowered to make performative orders that facilitate such changes.

This is a symbiotic relationship that hints at the potentially troubling nature of blockchain applications in terms of governance and legitimacy: if the court orders performance on the part of its designers to alter the code in order to remedy some illegality, it might not be technically possible to follow that order, for the reason just described[99] – and so we circle back to the question of ex ante legitimacy and the fundamental issue that animates digisprudence: can a blockchain protocol that does not afford ex post oversight, and thus does not afford contestability, be said to be legitimate, *particularly* given the exemplification by the technology of the problematic characteristics of computational legalism? If the answer is no, the design ought not to be released.

---

[98] U.W. Chohan, 'The Decentralized Autonomous Organization and Governance Issues' (Rochester, NY: Social Science Research Network, 2017) SSRN Scholarly Paper ID 3082055 <https://papers.ssrn.com/abstract=3082055> accessed 19 June 2020.

[99] For a vivid example see the recent New Zealand case of *Ruscoe v Cryptopia Ltd (In Liquidation)* 728 (High Court of New Zealand) in which the hack of an online cryptocurrency exchange led to unauthorised transfers of around NZD 30m worth of cryptocurrency. One can perhaps detect an element of surprise in Gendall J's judgment when he states at [13] that "It seems this transfer is irreversible". Thanks to Tom Barraclough for notifying me of this case.

3. Achieving legitimacy in code

In the previous section I hinted at various elements that a blockchain application would require to have in order to be deemed legitimate. This section begins to fill out those requirements as they ought to apply to regulative code more generally, setting out what it is that they seek to emulate in those a-legal normative orders, and how that might be achieved.

3.1 The aspiration of legality

We saw above in the discussion of computational legalism how code can come to impose a kind of rule-bound heteronomy on those subject to its normative force. In the legal realm, it is sometimes noted that the ruleishness of legalism is one end of a spectrum, at the other end of which can be found the aspirational concept of *legality*. Legality seeks to maintain a connection between law as a system of behaviour-governing norms on the one hand and the principles that legitimate the sovereign's creation of those norms on the other. Legality is considered to be of fundamental importance in constitutional democracies; Bańkowski goes so far as to say it is 'something worth living for; something worth dying for.'[100] For Hildebrandt it reflects not just a commitment to legal certainty, justice, and purposiveness, but also the rule of law and the binding of the sovereign to constitutional rules constraining its legislative power.[101] For Brownsword, legality is about dignity and the creation and maintenance of conditions that 'make moral community possible', those conditions requiring formal as well as substantive legitimacy.[102]

In Fuller's influential discussion of legality, he sets out eight principles that together make up formal requirements that he argues should be reflected in the creation of all good positive laws, regardless of any reasonable controversy there might be about their substantive content (their 'external

---

[100] Bańkowski, *supra* n. 61, p. 45.

[101] Hildebrandt, *supra* n. 25, pp. 157–158, echoing Radbruch's antinomian understanding law as the tension between legal certainty, justice, and purposiveness. See G. Radbruch, 'II. Legal Philosophy' in K Wilk (ed.), *The Legal Philosophies of Lask, Radbruch, and Dabin* (Cambridge, MA and London, England: Harvard University Press, 1950).

[102] Brownsword, *supra* n. 21, p. 1324f.

morality').[103] The eight principles require that rules (i) have general rather than arbitrary application, (ii) are made available for scrutiny by those subject to them, (iii) are not retroactive, (iv) are articulated clearly, avoiding obscurity or incoherence, (v) are not contradictory, (vi) do not require the impossible, (vii) are reasonably constant through time, and (viii) are congruent between their terms and how they are implemented through official action.[104] One can appreciate that several of these suggest design constraints for legal norms, regardless of their positive content. Fuller uses the language of design on various occasions, referring to law-making as a 'craft'[105] and to the eight principles as 'those laws respected by a carpenter who wants the house he builds to remain standing and serve the purpose of those living in it.'[106] The idea of a rule's legitimacy being in part contingent on formal aspects of its design reflects the temporal aspect of norm-creation, and the notion of 'input' and 'output' reasons for particular rules and decisions.[107] As we shall see, this idea of ex ante legitimation is one that has particular salience in the code context and the goal of digisprudence of ensuring that the fixity of code is as legitimate as it can be from the outset.

Wintgens' principles of legisprudence are concerned to an even greater degree with the legitimation of prospective legal norms through the imposition of formal limits.[108] For him, respect for individuals' subjective notions of freedom ought to be a guiding principle of both politics and law, and any limitation on those notions by legislative rules can only be legitimate if it is justified according to the four legisprudential principles.[109] The principles are concerned with whether a *rule* is desirable at all (principle of alternativity), whether the proposed rule is proportionate to the issue the legislator seeks to address (principle of normative density), whether its design enables an ongoing assessment of its efficacy (principle of temporality), and finally whether it is coherent at the semantic, temporal, intra-

---

[103] Fuller, *supra* n. 14, ch. 2.

[104] *Ibid.*, pp. 46–81.

[105] *Ibid.*, pp. 43, 156.

[106] *Ibid.*, p. 96.

[107] See Goldoni, *supra* n. 5, p. 127, citing J. Waldron, 'The Core of the Case against Judicial Review' (2006) 115 *Yale Law Journal* 1346.

[108] Wintgens, 'Legisprudence as a New Theory of Legislation', *supra* n. 68. For an explanation and history of the term 'legisprudence', see Wintgens, *Legisprudence*, *supra* n. 1, pp. 231–235.

[109] Wintgens, *Legisprudence*, *supra* n. 1, p. 220.

systemic, and extra-systemic levels (principle of coherence).[110] The following of rules thus remains a necessary part of the legal order, but fidelity to them is now via a 'weak' instead of a strong legalism. Rules that respect the four principles cannot (*inter alia*) be arbitrary exercises of sovereign power, and they are thus, according to Wintgens, justified incursions on individual freedom, and ought therefore to be followed.

These theories provide an important lens for ex ante assessments of other, a-legal forms of normative rule-making, such as we have been discussing.[111] The principled frameworks discussed above foster sensitivity to numerous factors in the creation of legislation that are absent in the private ordering of code. The goal of digisprudence is to adapt and import them into the design context, so that the creation of normative order by and through code can be made (more) legitimate from the outset, in turn mitigating the negative effects of computational legalism.

3.2 The need for an ex ante focus on code's production

If the above theories are concerned with the production of legal rules that are legitimate, the question is how to import those ex ante design constraints into the design process to ensure code rules are legitimate. I have already mentioned the importance of widening our focus to include the production of code, in addition to the usual ex post assessment of its operation. This relates to the definitions of legality and legitimacy just set out – we saw the relevance of the design of a rule to the question of whether it meets those standards. To combat computational legalism demands a focus on production, because once the code is 'out there' it can be very difficult or impossible to remedy its illegitimacy.[112] Moreover, unlike a law that can in principle be ignored by those subject to it, the code will continue to impose itself without regard to any such illegitimacy (recall the idea of the hermeneutic gap between text and action being collapsed). The elements of computational legalism – code's ruleishness, opacity, immediacy, immutability, and pervasiveness – mean that whatever code is promulgated by the designer

---

[110] Chapter 4 sets out the principles in greater detail in section 2.3 (Wintgens' legisprudence).

[111] Fuller appreciated this point, applying his principles to the rules governing a college dormitory. See Fuller, *supra* n. 14, p. 125 *et seq*.

[112] On this point in relation to DAO blockchain applications, see Chohan, *supra* n. 98.

must be legitimated from the outset. Designers limit individual and collective freedom in ways that have not been ratified by the democratic polity, via mechanisms that are technically and socially opaque, and which are not straightforwardly susceptible to public contest, redress, and (judicial) review. They are therefore potentially illegitimate exercises of power, and their negative effects are difficult to arrest or to ameliorate when diffused across potentially millions of devices, often with little or no technical means of applying retrospective fixes. Through the ex ante guidance of designers' production of technological normativity, we can help ensure that the illegitimate effects toward which computational legalism tends are minimised as far as possible.

3.3 Synthesising digisprudence: the affordance of legitimacy in code

As Koops suggests, 'a good place to start looking for criteria for acceptability of normative technology is to study criteria for law.'[113] We have seen above how Fuller's and Wintgens' models are concerned with providing criteria for good law-making. Undoubtedly these theories do not map directly onto the digital context, and so digisprudence translates their goals into a framework of normative affordances that ought to be present for code to be deemed legitimate. Table 1 below distils Fuller's and Wintgens' principles into a first set of these *digisprudential affordances*, mapping the characteristics of computational legalism first onto those principles and second onto the affordances that can ameliorate them. If a code artefact with regulative effect exhibits these affordances – i.e. its design relates to its user in these ways – then it can be argued that it meets a certain minimal threshold of formal legitimacy.

[113] Koops, *supra* n. 8, p. 162.

Characteristic of computational legalism	*Fullerian principle*	*Legisprudential principle*	*Digisprudential affordance(s)*
Ruleishness	Contradictory (v) or impossible (vi) rules	Alternativity; Normative density	**Transparency (provenance & purpose);** **Choice**
Opacity	Promulgation (ii); Intelligibility (iv)	Alternativity; Normative density; Coherence	**Transparency (provenance & purpose);** **Transparency (operation)**
Immediacy	Contradictory (v) or impossible (vi) rules; Frequency of change (vii)	Normative density; Temporality	**Delay;** **Choice**
Immutability	[in]Frequency of change (vii)	Temporality; Coherence	**Oversight;** **Choice**
Pervasiveness	-	Normative density	-
All of the above	-	-	**Contestability**

Table 1. Mapping legal legitimacy onto code legitimacy

The affordances are simultaneously general and to an extent concrete: they provide a design goal that should be reflected in all user-facing code, regardless of the type of technology, its substantive functionality, or the underlying business model.

An important corollary of this is that certain designs or business models will be *a priori* illegitimate if the affordances are not or cannot be included to a sufficient degree. This does not ignore the fact, however, that there will inevitably be cases where the affordances are less easy to envisage or to implement. Just like the Fullerian and legisprudential foundations upon which it builds, digisprudence is aspirational: the affordances are intended to encourage better (if not perfect) design. As Fuller suggests, perfect legality is 'utopian';[114] Wintgens for his part notes that respect for the legisprudential

[114] Fuller, *supra* n. 14, p. 41. See also p. 43.

principles is about 'the aspiration to do the job as well as possible'.[115] The same can be said of what is proposed here.

3.3.1 Contestability: an affordance both for citizens and institutions

It is crucial also to bear in mind the importance of affording contestability as an overarching concern. This central criterion of legality means it must be possible to question the code, and file a suit against its creator, in a court. The ability to 'return' from the normative order of code to the institutions of law, and especially the courts, is an important part of retaining the role and the rule of law, even within the a-legal normative order of code. If the nature and extent of the code's problematic effects cannot be observed, however, traditional mechanisms of legal redress cannot be meaningfully invoked. As discussed above, we fall into a trap if we assume that positive law is capable of operating in its usual way where code is the subject of its regulation.[116] Computational legalism militates against contestability: users must be in a position to understand the normativities they are being subject to in order to mount any kind of legal challenge to them. For citizens, resistance and transparency are concerned with the ability of the user to 'see' and question the code-based norms to which she is subjected, following which she can contest them legally. But for contestability fully and properly to be embodied in code, legal institutions must be afforded proper evidential scrutiny. If the citizen seeks to contest the code legally it is axiomatic to say that she must be capable of leading evidence of her complaint, and that evidence must be intelligible to the trier of fact. Here we maintain the connection between the realm of code and the rule of law, ensuring that whatever happens in the code-based normative order the judicial process can still perform its proper democratic function as the ultimate arbiter of any dispute. The affordance of contestability, then, is necessary not just to enable the citizen to understand code's normativities sufficiently well that she can choose to contest them, but is necessary also to enable legal institutions to grapple with the code from an evidential perspective.

This raises further questions of due process vis-à-vis evidential quality and propriety, and how these interact with the design process. From an evidential perspective, certain standards must be met in

---

[115] Wintgens, *Legisprudence*, *supra* n. 1, p. 280.

[116] Hildebrandt and Koops, *supra* n. 16, p. 440.

order for an action to proceed and succeed; from a design perspective this means that the affordance of those standards must be considered ex ante during the design process if both aspects of contestability are to be facilitated. If designers do not account for the affordance of contestability at the stage of production, it will be that much more difficult (and in some cases impossible) at the stage of operation.

Contestability thus operates as an overarching concern, suffusing digisprudence as an ultimate backstop that maintains a connection with the rule of law and institutional legal processes. No matter the merits or demerits of the design from a digisprudential perspective, it must in the end always be possible for the user to resort to a court action to determine illegality of whatever substantive form. This ensures the continuing role of the rule of law, notwithstanding code's existence as a separate a-legal normative order.

3.3.2 Technological constitutionalism and the programmer of the programmer

So where to impose the above 'constitutional' requirements in the code-making process? Recall relationship (e) in Figure 1 above, which exists between the programmer of the programmer[117] ('PoP') and the product designer. The PoP designs the tools that the product designer in turn uses to create the products and services that are ultimately destined for the user. Situated at a 'constitutional' level of the code production process, the decisions made by the PoP fundamentally impact on what the product designer can and cannot do. In this way, the PoP has a crucial power to define the rules of the production game before it even begins. This idea of 'technological constitutionalism' suggests a potential point where digisprudential requirements can be implemented. If parliaments are restricted by constitutions in the ways in which they can make rules, and the formal qualities that those rules can have, then perhaps the development environment used by the designer to create digital products can similarly contain that work within constraints that encourage or even ensure the production of legitimate code. The power that inheres in product designers operates at the meta-level, through the design of programming languages, integrated development environments (IDEs), and code development

[117] Vismann and Krajewski, *supra* n. 36. For an earlier discussion that alludes to a similar concept see J. Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation* (San Francisco: Freeman, 1976) p. 100 *et seq*.

processes such as the agile methodology. These are areas ripe in potential for the application of constitutional requirements to the code production process.

4. Digisprudence: a recap

Coming toward the end of the article, it may at this point be useful briefly to recap its central claims.

First, code can have regulative effects on behaviour that are more pervasive and direct than law is capable of. Together, these regulative effects form a corpus of norms that are separate from positive institutional law. The ontological characteristics of code mean that in many cases this separate normative order will be enforced whether or not it is compatible with positive law. These characteristics, which collectively I call *computational legalism*, are its ruleishness, opacity, immediacy, immutability, pervasiveness, and private production.

Second, in a democratic society, the norms that regulate citizens, of any kind and from any source, ought to be legitimate. In the legislative context this can be achieved by ensuring certain formal qualities are present in the design of those norms, as a separate concern from their substantive content. Examples of frameworks that suggest such qualities include Fuller's principles of legality and Wintgens' legisprudence. These legal-theoretical approaches are concerned to an extent with the *form* of the legal norm, as distinct from its political content. This formal notion of legitimacy acts as a kind of basic design constraint, which through the binding of the sovereign to the inclusion of certain characteristics limits to some extent what the substantive content of a norm can be.

Third, given that in the private, commercial contexts where code-based norms are produced there are no such formal principles constraining their design, the question arises of whether the mechanisms of producing legitimate normativity in the *legal* sphere might be transposed into the *computational* sphere. Some form of ex ante legitimation is crucial, because the instrumentality of computational legalism is potentially more problematic than illegitimate legal norms that, because of the hermeneutic gap between text and action, can be ignored, re-interpreted, or annulled by a competent court.

Fourth, any such legitimation of a code rule must, by definition, take place at design time. This is because, unlike positive law, there is little or no scope for reinterpreting the code after-the-fact; code and (latent) reality are one from the point of distribution onward.

Digisprudence thus adapts and imports the ex ante frameworks of legitimation from the legal world into the domain of code, using the language of design theory to remain sensitive to how code actually regulates and how designers produce it. The synthesis of these various theoretical strands is the framework of digisprudential affordances, set out above.

5. Conclusion and future work

This article is an initial, high-level articulation of digisprudence, a theory concerned with the legitimacy of the design of digital artefacts that have regulative effect on human behaviour. It builds on the philosophies of both law and design in a dialectical structure, the synthesis of which is the framework of digisprudential affordances that it argues should be present in all citizen-facing code. These affordances must by definition be consciously implemented at production time, which might be facilitated in part by the design environment to impose 'constitutional' constraints on what kind of code can ultimately be produced.

The discussion above of blockchain applications and the digisprudential affordances can only hint at the levels of analysis opened up when the domains of design and legal theory are brought together in this way. Other specific technologies such as machine learning, the Internet of Things, and robotics will raise many questions when analysed through a 'digisprudential lens', both in terms of individual artefacts themselves and the particularities of their production processes. Interesting questions are also raised by the context of a code's deployment, for example in automated public administration, the care sector, or the expanding 'legal tech' domain, where computational legalism might have reflexive effects that require to be considered in advance.[118] From a more operational perspective, the idea of

[118] For a discussion of some of these issues in the 'legal tech' sector, see L. Diver, 'Normative Shortcuts and the Hermeneutic Singularity' (*COHUBICOL research blog*, 4 June 2019) <https://www.cohubicol.com/blog/normative-shortcuts-and-the-hermeneutic-singularity/> accessed 19 June 2020.

technological constitutionalism embodied in the framework raises fascinating practical questions around development methodologies, integrated development environments, and the nature of programming languages as the means of expressing normative requirements. These are all sites where the notional programmer of the programmer plays a pivotal role in structuring, from the outset, how regulative code is produced.

Analysis in the legal literature has so far avoided the material characteristics of design, focusing instead on the human factors in the decision to use code to implement a norm. This is a significant gap that ought to be remedied. Digisprudence simultaneously adopts an internal legal perspective on what constitutes legitimacy within that discipline whilst also looking outward at the concrete realities of how the digital artefacts we critique do what they do.[119] This cross-disciplinary analysis can be facilitated by concepts in the philosophy of technology that sensitise us to the relational nature of technologies and how they shape our experiences. Once those experiences are conceptualised, we can begin to think seriously about what legal philosophy ought to say about them, avoiding legalistic or positivist perspectives that might see computer code as something 'just there' and textual rules as the only proper concern of the lawyer and the legal theorist.

Power is shifting away from publicly-accountable legislators onto private actors who have pervasive control over the digital products and infrastructures that permeate contemporary life.[120] The ability of code to supplant law as the dominant normative enterprise, and the privatised nature of the environments within which code is produced, raise the question of how to ensure that that new normativity is legitimate. As Lessig argued, 'if code is a lawmaker, then it should embrace the values of a particular kind of lawmaking.'[121] We do not have to accept the claim that code is law in order to accept that it nevertheless has important normative effects, and that that those effects ought to be legitimate. If the legitimacy of behavioural regulation, broadly conceived, is a concern within constitutional

---

[119] See for example Verbeek, *supra* n. 57.

[120] Brownsword, *supra* n. 21, p. 1324; M.J. Radin, 'Regulation by Contract, Regulation by Machine' (2004) 160 *Journal of Institutional and Theoretical Economics (JITE)* 142.

[121] Lessig, *supra* n. 49, p. 328.

democracies, we cannot ignore the very real role played by the materiality of the code that imposes it, and in turn by the processes through which it is produced.

6. Acknowledgments

I would like to thank Burkhard Schafer for 'shepherding' me through the writing of the thesis from which this article is derived, as well as Roger Brownsword and Claudio Michelon for their productive interrogations of my arguments. I'd also like to thank colleagues and friends in LSTS (particularly Irina Baraliuc, Tatiana Duarte, Gianmarco Gori, Mireille Hildebrandt, Emilie van den Hoven, Liisa Janssens, Paulus Meessen, Manuel Sabin) for their support and our many stimulating discussions on this and closely related topics. Thanks also to two anonymous reviewers whose inputs on an earlier draft helped me to hone the final text.