Part III

Legitimating Code: Theory and Practice

6

The Digisprudential Affordances

> To promote the benefits of legality, and to prevent the disadvantages of legalism, we will require new forms of interaction with these systems.[1]

This third part of the book represents its main practical contribution. In this chapter, I synthesise the theoretical ideas from the past four chapters into a framework of 'digisprudential affordances' that can provide a baseline of regulative legitimacy. By employing affordances not just to describe what code does but also as a conceptual lens for what it *ought* to do, I provide a novel way to identify those new forms of interaction referred to in the quote above from Bańkowski and Schafer. One can appreciate from the review of the literature in the previous chapter that although the work in this area has considered criteria for the use of code as a regulator, this scholarship has not engaged in depth with the relational theories of design that I set out in Chapter 2, and has therefore not considered the desirability, efficacy, or indeed legitimacy of code from that perspective. This chapter contributes such an analysis through the synthesis of the legal-theoretical perspective of legitimacy with design theory, the outcome of which is a set of affordances that code ought to exhibit in order to be deemed legitimate.

I begin by mapping the characteristics of computational legalism onto the Fullerian and legisprudential principles. From this mapping relationships can be identified between those principles and the affordances that can help to embody their aims within the design of code. The affordances provide both a way of asking what features a given design provides, and a set of goals for what should be afforded in order to achieve legitimacy. Designers should not think only about what their code is intended to do from a commercial perspective; they ought also to make informed assessments of whether

[1] Z Bańkowski and B Schafer, 'Double-click justice: Legalism in the computer age' (2007) 1 *Legisprudence* 31, 46.

it is, and if not how it might become, legitimate. The digisprudential framework is a mechanism for guiding those anticipations from a legal-theoretical perspective.[2]

6.1 Assessing Decisions, or Assessing Design?

We saw in the previous part of the book a tendency to focus on background decisions as the target for tests of legitimacy, a perspective that implicitly views the code that embodies them as a separate product of those decisions. By contrast, digisprudence is concerned with the legitimacy of the resulting design itself, whatever those decisions were. This means that any ontological separation between the normativity of the code and the preceding decision animating its use must not result in a focus on only the latter half of the whole: the code is what does the work of instantiating whatever normativity ultimately comes into being, so to ignore it is to introduce a fundamental blind spot into our analysis. This involves grasping the technical nettle, but that is unavoidable. We saw in Chapter 2 how design has a direct influence on end-user behaviour. We also saw in Chapter 3 how representationalism means that the text of the 'rule' (that is, the source code) and its operation are two halves of a whole: isomorphism between source code and the materiality of the artefact is to a great extent a given in the computational context. Rather than querying the motivations behind the design of a particular artefact's code (important though this is), the task then is to query what the code in fact does, and whether the normativity that it imposes is itself legitimate, separately from those anterior motivations. The distinction is nuanced but fundamental, for if we focus on only the motivation behind a design but fail to look critically at how that motivation is in fact instrumentalised, we risk not only failing to observe what the artefact actually does, but also – and potentially worse – sanctioning it in the erroneous belief that because the decision to use code was sound the implementation must also have been.

The aim therefore is to guide production of the 'moreness' of code (its instrumentality; Chapter 2) in ways that reduce its 'lessness' (its computational legalism; Chapter 3). The question is ultimately one of what the design affords the end-user (contestability, choice, transparency, and delay), legal institutions (appropriate and sufficient quality of evidence), and its own designer or manufacturer (oversight). The ultimate goal of digisprudence is fidelity to the input criteria of legality (Chapters 4 and 5) that ensure that

[2] There is a parallel here with Verbeek's discussion of anticipating the morality of a given set of technological mediations, and imposing restrictions where necessary. See P-P Verbeek, 'Materializing morality: Design ethics and technological mediation' (2006) 31 *Science, Technology, & Human Values* 361, 370, 372.

its technological normativity, whatever its substantive output functionality, includes mechanisms that ameliorate the path dependencies of computational legalism.

6.2 Mapping the Criteria

Below I map the Fullerian and legisprudential principles discussed in Chapter 4 onto the relevant characteristics of computational legalism, showing how they apply across the separate normative orders of institutional law and code. I then consider how the digisprudential affordances reflect the purposive goals of the principles, providing a baseline of legitimacy. There are overlaps between the characteristics and the suggested affordances – any given affordance does not apply only to ameliorate the specific characteristic indicated; instead, the idea is to consider them in a holistic fashion, the goal being to achieve more legitimate technological normativity through a concurrent sensitivity to each of the issues raised. Wintgens makes a similar claim about legisprudence, plotting the notional justifications of various norms on a 'graph', the horizontal axis representing at one side the principle of normativity and at the other the principle of temporality, and the vertical axis similarly representing the principle of alternativity and the principle of coherence (Figure 6.1).[3]

A given norm will be more or less justified according to each of the four legisprudential principles; a notional norm that is equally justified according to each of the principles would be plotted in the centre of the graph. An important point is that the particular circumstances surrounding the proposed norm will affect the extent to which it must be justified by each of them. For example, emergency legislation following a natural disaster might be justified despite a relative lack of in-depth fact-finding or foresight, because the alternatives would not implement the powers and duties necessary to facilitate the (presumably widely supported) purpose of mitigating the disaster.[4] This might mean there is strong justification under the principle of alternativity, but less under the principle of coherence, but that on balance the former is sufficient to justify the norm in the circumstances. In that situation, respecting the principle of temporality can provide justification from another direction – because of the emergency the design of the norm may not perfectly cohere to the rest of the system of norms, but if mechanisms

[3] Figure 6.1 is reproduced from L Wintgens, *Legisprudence: Practical Reason in Legislation* (Routledge 2012) 282.

[4] Ibid. 307.

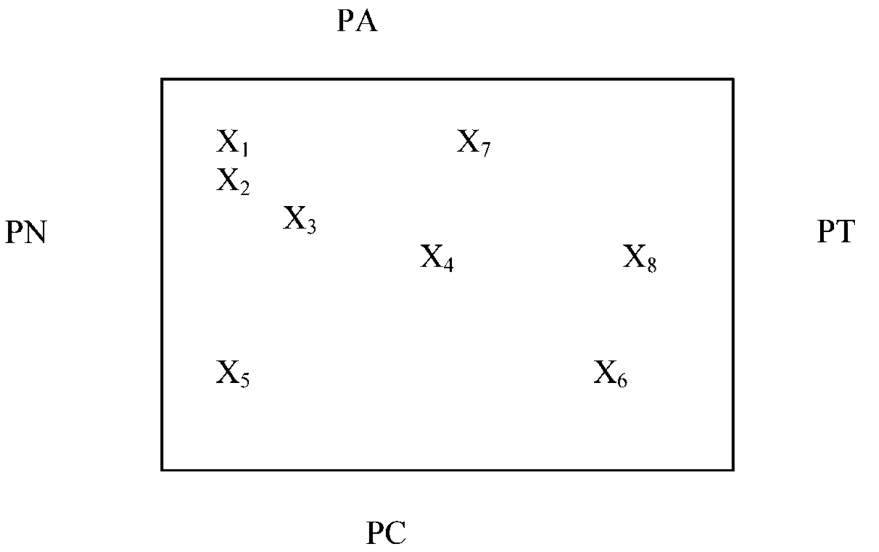

Figure 6.1 Differing justification of norms

for oversight are put in place, this may nevertheless justify its imposition, all things considered.[5]

In a similar fashion, the digisprudential affordances are not intended to be viewed in isolation, but rather to be seen as a set of elements that work in concert to achieve more legitimate configurations of technological normativity. Depending on the purpose and intended use of a particular artefact, the relevance of each principle will vary, and therefore so too will the justificatory contribution of each affordance. An artefact that includes none of the affordances is on balance unlikely to be legitimate.

The intention here is to contribute a set of 'normative anchor points'[6] that are explicitly oriented towards legal-theoretical concerns.[7] The exercise is

[5] At time of writing we are seeing this phenomenon play out in real time in relation to the COVID-19 pandemic, where lockdown regulations have built-in requirements for periodic reassessment. For an example, see section 12 of the Coronavirus (Scotland) (No. 2) Act 2020.

[6] E.g. R von Schomberg, 'A vision of responsible research and innovation' in R Owen, J Bessant and M Heintz (eds), *Responsible Innovation* (John Wiley & Sons 2013) 63 *et seq*.

[7] Cf. other frameworks for computer ethics. See for example Association for Computing Machinery, 'ACM Code of Ethics and Professional Conduct' (2018) <https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct> last accessed 4 March 2021; BC Stahl et al., 'From computer ethics to responsible research and innovation in ICT' (2014) 51 *Information & Management* 810.

necessarily qualitative to an extent, and therefore it requires human judgement and the willingness to do the right thing on the part of the designer. Table 6.1 maps the characteristics of computational legalism onto the Fullerian and legisprudential principles, and suggests the digisprudential affordances that can embody the purposive aims of those principles within code.

(a) Contestability as an Overarching Affordance

The central importance of facilitating contestability must be borne in mind as an overarching concern. This means keeping open the possibility of questioning the code (and by implication its creator) in a court of law. As we saw in the previous chapter, contestability is a central criterion for the maintenance of the rule of law in the computational context. From a strictly legal point of view, the ability to 'return' from the normative order of code to that of institutional law is an important part of retaining the role and the rule of law within the a-legal realm of code. Computational legalism militates against contestability: end-users must be able to understand the normativities they are being subjected to in order to mount any kind of legal challenge to them. As I will discuss below, resistance and transparency are aspects of this that are concerned with the ability of the end-user to 'see' and question the norms to which she is subjected. This is the user-centric side of the contestability coin, but on the other side are legal institutions, most importantly the courts.

Affording Evidential Scrutiny to Courts

In order for contestability to be properly embodied in code, courts must be afforded proper evidential scrutiny. If the end-user seeks to contest the code, she must be able to bring evidence of her complaint, and that evidence must be intelligible to the trier of fact. Whatever happens in the code-based normative order, the judicial process must retain its role as ultimate arbiter of any dispute. The affordance of contestability, then, is necessary not just to enable the end-user to understand the 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 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 order for an action to succeed; from a computational perspective, this means that affording those standards must be considered during the design process if valid contestability – and therefore legitimacy – is to be achieved. Without consideration of the need for contestability at the point of production, it will not be possible (or will be that much more difficult) at the stage of operation.

Table 6.1 Mapping legitimacy across domains: the digisprudential affordances

Characteristic of Computational Legalism	Fullerian Principle	Legisprudential Principle	Digisprudential Affordances
Ruleishness	Contradictory (5) or impossible (6) rules	Alternativity (PA); Normative density (PN)	**Transparency (provenance & purpose); Choice**
Opacity	Promulgation (2); Intelligibility (4)	Alternativity (PA); Normative density (PN); Coherence (PC) LoC_3	**Transparency (provenance & purpose); Transparency (operation)**
Immediacy	Contradictory (5) or impossible (6) rules; Frequency of change (7)	Normative density (PN); Temporality (PT)	**Delay; Choice**
Immutability	[In]frequency of change (7)	Temporality (PT); Coherence (PC) LoC_3	**Oversight; Choice**
Pervasiveness	–	Normative density (PN)	–
All of the Above	–	–	**Contestability**

The consequences of failing to afford contestability are demonstrated by the Post Office Horizon scandal, only recently resolved in favour of the sub-postmistress/sub-postmaster 'end-users' whose livelihoods were destroyed in part as a result of that failure.[8] There, a presumption of the infallibility of the Horizon computer system led in effect to a strict liability presumption that any discrepancies in the accounts of a Post Office branch could only be the result of fraud. At common law there is a presumption that deems a computer to be operating as expected unless there is explicit evidence to the contrary.[9] The evidential burden that this placed on the sub-postmistresses and sub-postmasters whose accounts showed discrepancies could not in practice be discharged, because the Horizon system removed their access to relevant system logs upon detection of an accounting shortfall.[10] Without access to evidence of the system's operation, those subsequently accused of fraud were unable to mount an effective defence to the effect that there must be bugs in the code causing the erroneous shortfalls. The code did in fact contain numerous bugs responsible for the mis-accounting, but these were not directly considered by the courts until *Bates* in 2019.[11] In the ensuing period – well over a decade – more than 1,000 sub-postmistresses and sub-postmasters lost their livelihoods, over 900 of them being wrongfully convicted of false accounting, fraud, and theft.[12]

Following *Bates*, it has been proposed that the common law presumption be reversed, so that code is assumed, instead, to be fallible.[13] This is on the

[8] K Peachey, 'Postmasters' huge step towards quashing convictions' *BBC News* (2 October 2020) <https://www.bbc.com/news/business-54384427> last accessed 4 March 2021.

[9] P Marshall, 'The harm that judges do – misunderstanding computer evidence: Mr Castleton's Story' (2020) 17 *Digital Evidence and Electronic Signature Law Review* 25, 26. The presumption has operated at common law since 1999, following (misguided) recommendations by the Law Commission of England & Wales that led to the repeal of s. 69 of the Police and Criminal Evidence Act 1984 that had required evidence of the computer's proper operation. See Law Commission, 'Evidence in criminal proceedings: Hearsay and related topics' (Law Commission 1997) LC245 para. 13.13–13.14.

[10] Marshall (n 9) 26–7. See *Bates & Ors v Post Office Ltd (No 6: Horizon Issues)* [2019] EWHC (QB) 3408, *per* Fraser J at [1000].

[11] For a summary of the issues likely to result in further litigation, see *Bates* (n 10), *per* Fraser J at [965]–[1030].

[12] Marshall (n 9) n 14. The article provides an upsetting account of the experiences of those affected by a combination of unethical behaviour on the part of the Post Office/Fujitsu (Horizon's developer) and the courts' uncritical trust in the Horizon code.

[13] PB Ladkin et al., 'The Law Commission presumption concerning the dependability of computer evidence' (2020) 17 *Digital Evidence and Electronic Signature Law Review* 1. For an argument that this does not adequately take into account the asymmetries of information inherent between the roles of designer and end-user, see A Hicks, 'The role of usability, power

basis that code is rarely if ever bug-free, even in safety-critical applications.[14] Viewed this way, the notion of reliability can be reframed as both a social and a technical standard,[15] thus wresting assessments of efficacy away from the lure of automation bias that were so evident in the Horizon scandal. Affording contestability necessitates access, and a system that is designed to prevent that access will, without appropriate additional safeguards, struggle to meet an appropriate standard of legitimacy. I discuss in the next chapter some approaches that can assist in implementing this institutional aspect of contestability during production.

Contestability is thus an overarching concern; a necessary backstop. Whatever the other merits or demerits of the design from a digisprudential perspective, it must in the end always be possible for the end-user to resort to 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.

6.3 From Characteristics to Affordances

We come now to a central part of the book's synthesis, where the analysis of computational legalism and its negative effects meets the affordances that can ameliorate them. What follows is aspirational; the idea is to aim for legitimacy in privately designed code in the knowledge that on the one hand attaining absolute perfection is unlikely, but on the other that the characteristics of computational legalism make the attempt all the more important.

(a) Ruleishness

Code is extreme in its precision; it is not flexible by nature. While code rules can exhibit non-discrimination, in that they apply to every end-user completely regardless of their characteristics, this is a virtue only if the rule is

---

dynamics, and incentives in dispute resolutions around computer evidence' *Bentham's Gaze* (23 June 2020) <https://www.benthamsgaze.org/2020/06/23/the-role-of-usability-power-dynamics-and-incentives-in-dispute-resolutions-around-computer-evidence/> last accessed 4 March 2021; A Hicks and SJ Murdoch, 'Transparency enhancing technologies to make security protocols work for humans' in J Anderson et al. (eds), *Security Protocols XXVII*, vol. 12287 (Springer 2020).

[14] In that context, reliability rates of one defect per 1,000 lines of code may be deemed 'robust' (Ladkin et al. (n 13) 2). By way of comparison, the Windows 10 operating system contains 50–60 million lines of code.

[15] On which, see PB Ladkin, 'Robustness of software' (2020) 17 *Digital Evidence and Electronic Signature Law Review* 15, observing the central role of human concepts such as trust and the contextual acceptability of failure rates and severity in assessing code. This is relevant to the concept of 'tussle', discussed below.

legitimately designed.[16] As set out in Chapter 3, code rules execute in every situation where their predetermined requirements are met, they execute in no situations where those requirements are not met (no matter how close the circumstances are), and the exact consequences specified within the rule are all that will or can flow from its execution. They are thus by nature brittle in the extreme. This aspect of ruleishness lies at the heart of computational legalism.

This relates to the earlier discussion of constitutive norms and the threshold between a design's 'constitution', or the behavioural constraints which are 'wired in', and its merely 'regulative' aspects which provide the end-user with the latitude to decide whether or not to acquiesce to a suggested limitation on her freedom.

Digisprudential Affordance: Choice

Default configurations of code contribute to shaping an end-user's understanding of the behavioural possibilities it affords her. End-users tend to trust that the designer has made the 'right choice' for them, even where the code permits alternatives – the situation presented to the end-user is perceived to be normal, and even legitimate in pervasive systems. Once the artefact is operating, the outputs of its processes tend to be trusted by end-users, due in part to automation bias.[17]

The effects of immediacy and immutability can be countered through the appropriate affordance of configurability. For the latter characteristic, the ability to change the configuration of the system is by definition in opposition to the state of immutability. But the mere provision of choice is not sufficient on its own. To ameliorate ruleishness and empower the end-user, the choice must be between appropriate options and must be given at the appropriate time,[18] otherwise we have the code equivalent of long terms of use documents which notionally inform the end-user but which in practice leave her bewildered. Configurability *per se* can thus become a counterproductive burden, particularly for naïve end-users who may be confused and intimidated by too many options.[19] At any rate, customisation is viewed by many end-users as time-consuming, and so they avoid it even where objectively it could benefit them by enabling them to choose options that reflect their interests

[16] Bańkowski and Schafer (n 1) 46.

[17] DK Citron, 'Technological due process' (2008) 85 *Washington University Law Review* 1249, 1271–2.

[18] JP Kesan and RC Shah, 'Setting software defaults: Perspectives from law, computer science and behavioral economics' (2006) 82 *Notre Dame Law Review* 583, 601.

[19] Ibid. 627.

and/or preferences.[20] Thus, the ways in which code affords configurability must be considered in advance if the relevant audience is to be appropriately empowered by them.

'TUSSLES' AND DESIGNING FOR CHOICE

The notion of 'tussle' in computer science touches on the issue of choice and the role of design in responding to different parties' interests.[21] 'Tussle points' are points in the design of code that implicate conflicting interests. Such conflicts can be technical, legal, social, or economic, and ought to be anticipated by the designer:

> Our position is that the laws of men and the so-called whims of bureaucrats are part of the fabric of society, like it or not. They are some of the building blocks of tussle, and must be accepted as such. We, as technical designers, should not try to deny the reality of the tussle, but instead recognize our power to shape it.[22]

The commercial attractions of computational legalism are in conflict with the goals of legitimacy, and thus they create a 'tussle space': enterprise uses the former to advance its interests – ruleishness and immutability provide predictability; opacity provides protection of commercial secrets and can hide profitable but dubious normativities; immediacy gives feedback and marketable results. There is thus a potential tussle between the interests of the end-user and the enterprise that wishes to channel her behaviour in predictable (and profitable) ways. Digisprudential legitimacy seeks to uphold basic 'constitutional' safeguards against illegitimate regulation of behaviour, but these are by nature absent in the default conditions of computational legalism.

Anticipating points of tussle during the production phase is crucial to avoiding problems during operation. One approach to dealing with tussles is to design for choice, the premise being that the design ought to anticipate and allow for different possibilities: '[r]igid designs will be broken; designs that permit variation will flex under pressure and survive.'[23] Although Clark et al. are concerned mainly with infrastructural design, from a digisprudential perspective we can think of designs that afford spaces that facilitate end-user

[20] Ibid. 598. For an early and influential study on this point, see WE Mackay, 'Triggers and barriers to customizing software' in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Reaching through Technology – CHI '91* (ACM Press 1991).

[21] DD Clark et al., 'Tussle in cyberspace: Defining tomorrow's Internet' (2005) 13 *IEEE/ACM Transactions on Networking (ToN)* 462.

[22] Ibid. 473.

[23] Ibid. 466.

autonomy (all things being equal – the notional scope for exercising autonomy depends of course on the fundamental purpose of the design). This accords with the requirement, identified in Chapter 5, that code permit individuals to make choices in order to preserve their capacity for moral action.

Designing for choice also fits the legisprudential principle of alternativity, which as we saw concerns the legitimate use of 'rigid' rules that admit of no latitude, versus alternatives that incentivise or suggest courses of action. Leaving the tussle space open to allow for choice thus shifts constitutive power away from the designer.[24] The principle of alternativity in this context requires that (1) the implementation of unconfigurable normativity in the code be more desirable than not, and (2) the use of a strict rule *per se* rather than some less ruleish mechanism be necessary, for example a suggested default, or a configurable option. The first speaks to the enterprise's business model and how this is articulated in code, raising some potentially existential questions as to the fundamental desirability of a given approach or product. As Hartzog warns, 'too often industry wants the freedom to experiment on the public without accepting the responsibility for the harm they cause'.[25] We might then think of a kind of Hippocratic oath for code – 'first, do no harm' – leading the responsible designer/enterprise to conclude that the feature or product should not be developed at all.[26] Shifted into the present context, a computational principle of alternativity would assess first whether a given (dis)affordance/inscription is necessary for the operation of the artefact and the business model being pursued. If it is not, then *a priori* it should not be included in the design because it represents an unnecessary and unjustified limitation of the end-user's freedom.

If that element of the design is necessary, however, the question then becomes one of the ruleishness of the implementation – how 'wired in' does the functionality need to be to achieve the designer's goal? Could the code require the end-user to exercise a choice, or perhaps provide a passive configurable option? Or does the purpose of the code imply the need for 'nudging'/inscription, or even the requirement (wiring in) of one of the possible options to the exclusion of all others? The latter is the most 'ruleish' form of technological normativity, while the former approaches are progressively

[24] Ibid. 473.

[25] W Hartzog, *Privacy's Blueprint: The Battle to Control the Design of New Technologies* (Harvard University Press 2018) 85.

[26] AD Selbst et al., 'Fairness and abstraction in sociotechnical systems' (Social Science Research Network 2018) SSRN Scholarly Paper ID 3265913, 13–14 <https://papers.ssrn.com/abstract=3265913> last accessed 4 March 2021.

less constraining.[27] As with the legisprudential principle of alternativity, the decision to choose a more 'ruleish' (and therefore less choice-oriented) design approach must be justified because of the correspondingly larger limitation it places on freedom. I will consider this issue further in the discussion below of default choices.

The anticipation of conflicting interests that the concept of tussle implies is connected to the notion of agonism in a democracy,[28] the theory that adversarial debate can be fruitful where it enables contrasting points of view to be ventilated and compromise thereby to be achieved. 'Inviting dissent' in this way, which can be consciously facilitated by design,[29] is ultimately at 'the core of both democracy and the rule of law'.[30] For Hildebrandt, this is reflected in ex ante participatory design processes like constructive technology assessment[31] which aim to achieve a 'settlement' during the design process that takes into account the views of those with a stake in the outcome.[32] I have already suggested that such processes are unlikely to be used in many design contexts, and particularly in small and medium enterprises. As previously noted, however, design for all need not entail design with all.[33] Initiatives like constructive technology assessment seek to legitimise a design by the fact of having involved stakeholders in decisions as to its substantive characteristics – the design is legitimate because those affected by it have in some sense approved it. This is a separate concern from digisprudential legitimacy, since such processes are built around the value of participation, rather than an underlying theory of what provides legitimacy.[34] Recalling the earlier discussion of input and output legitimacy, these approaches follow a 'thick' conception of legitimacy that is based on particulars, rather than a 'thinner' formal conception that is separate from, and prior to, the substantive content of participants' views on the merits of a design. The 'constitutional' concerns

[27] Recall the discussion in Chapter 2 of the spectrum of technological normativity.

[28] M Hildebrandt, 'Algorithmic regulation and the rule of law' (2018) 376 *Philosophical Transactions of the Royal Society A* 20170355, 7–8.

[29] See generally C DiSalvo, *Adversarial Design* (MIT Press 2012) and JL Davis, *How Artifacts Afford: The Power and Politics of Everyday Things* (MIT Press 2020) chapter 6.

[30] Hildebrandt, 'Algorithmic regulation and the rule of law' (n 28) 7.

[31] JW Schot, 'Constructive technology assessment and technology dynamics: The case of clean technologies' (1992) 17 *Science, Technology, & Human Values* 36.

[32] Recall the discussion in the previous chapter contrasting input and output assessments of code legitimacy.

[33] A Pols and A Spahn, 'Designing for the values of democracy and justice' in J van den Hoven, PE Vermaas and I van de Poel (eds), *Handbook of Ethics, Values, and Technological Design: Sources, Theory, Values and Application Domains* (Springer 2015) 351.

[34] Ibid. 353.

of digisprudence are sited at an earlier point in the design process than is the question of whether stakeholders have had their views taken into account. The two do not conflict, and a design process can certainly involve both, but if the design is not otherwise digisprudentially legitimate then the fact that a participatory process was used to shape it will not by itself provide legitimacy of the more fundamental kind I propose.

Nevertheless, the preservation of agonistic space can stand as a constitutional principle for code design, along the lines of tussle: by anticipating in advance the points at which tussle is likely to arise in the course of the design's operation, it is possible to avoid imposing one path or outcome for that tussle ahead of time, thus preserving the space for both choice and agonism. This expanded view of designing for choice enjoins the designer consciously to 'retreat' from any impulse to impose a constitutive outcome, thus preserving a space for agonism, for tussle, within the operating geography of the design itself. The domain of the morality of duty (computational legalism; external limitations on freedom) is reduced, and the domain of aspiration ('legality'; individual conceptions of freedom) accordingly expanded. This twist on agonism operates at runtime but is necessarily facilitated by decisions made at design time. 'Agonism' is in this sense a feature of the artefact's operation, rather than of the design process, although of course both may be present (that is, facilitating participation in the design process on the one hand, and the design affording space for agonism during its operation on the other). The extent to which implementation of this extended affordance of choice will be possible or plausible will depend on the intended use of the artefact. The design of a single-function Internet of Things (IoT) device like the Amazon Dash Button, for example, is less likely to allow for agonistic space than is the design of a social network service. This is an example of how different artefacts can and will reflect the digisprudential affordances to differing degrees.

An important practical design mechanism for facilitating tussle spaces is the modularising of an artefact's functionalities in ways that maintain a separation of interests: 'functions that are within a tussle space should be logically separated from functions outside of that space'.[35] This idea connects with the principle of normative density. In terms of the goals of that principle, the code should avoid bundling together code norms that are not conceptually related, forcing the end-user to acquiesce to multiple heterogeneous normativities simultaneously when she might be willing to accept some but would prefer to resist others. This idea is reflected in the GDPR's provisions requiring consent to be a genuine and free choice, requiring separate consents

[35] Clark et al. (n 21) 466.

for separate operations, and preventing the bundling of consent along with performance where the former is not necessary for the latter.[36]

When they exhibit computational legalism, the aggregated normativities of a system can lead to exponential negative effects as each of the legalistic characteristics amplifies the impact of the others. Modularisation can isolate discrete elements of normativity, perhaps along the boundaries of specific features or functions, thus enhancing the end-user's ability to accurately comprehend the system's effects by avoiding conflation of what should be conceptually isolated issues. Only the designer has the ability to 'modularise the design along tussle boundaries, so that one tussle does not spill over and distort unrelated issues'.[37] For example, in a smart thermostat, code profiling the end-user implicates a tussle involving different interests (the end-user's data protection rights versus the enterprise conducting its business). The functionality of that code is distinct from functionality intended to control the heating system efficiently, where the relevant interest is the end-user's need for domestic warmth balanced against wasteful energy consumption.[38] Modularising these discrete functionalities enables the end-user to understand them separately and to respond to them in different ways. This highlights the relationship (and tension) between default configurations and the affordance of choice.

DEFAULT CHOICES

Even in the 'offline' world, defaults are all around us – some initial configuration of architecture is an inevitability, which in turn implies the inherent non-neutrality of technologies.[39] We have seen at various points so far how this observation applies even more strongly in the computational context. It is not possible for a designer to leave open to interpretation what the design of the artefact should be in the way that it is possible for the legislator deliberately to leave the meaning of a textual norm somewhat open. Some choice must be made by the designer that constrains the infinite possibilities of the computational *tabula rasa*, and so intervening to ensure those initial decisions are legitimate becomes all the more necessary.

[36] GDPR, Arts. 7(2) and 7(4); Recitals 32, 42, and 43.

[37] Clark et al. (n 21) 466.

[38] The latter of course represents all our interests. For a critical discussion of the problems with smart thermostats, see B Krebs, 'IoT reality: Smart devices, dumb defaults' *Krebs on Security* (16 February 2008) <https://krebsonsecurity.com/2016/02/iot-reality-smart-devices-dumb-defaults/> last accessed 4 March 2021.

[39] M Hildebrandt, 'Legal protection by design: Objections and refutations' (2011) 5 *Legisprudence* 223, 238–9.

As Shah and Sandvig note, to rely on default settings is *de facto* to outsource decision-making to designers, shifting the agential emphasis away from both the end-user and the democratic state. It is thus necessary to 'push and prod developers to set default settings that comport with established societal concerns'.[40] If one of those concerns is the legitimacy of behavioural regulation, then the aspects of the code that are made 'chooseable' must in turn accord with that spirit. The number of choices and their quality (that is, what substantive functionality they enable the end-user to configure) are thus important design questions with respect to the scope of autonomy that they provide the end-user, and so too is the way in which these affordances of choice are communicated (signified) to her through the design. The provision of choice for choice's sake does not beget legitimacy if those choices do not facilitate the exercise of true autonomy.

Kesan and Shah identify a spectrum of design mutability, from 'wired-in' functionality that cannot be changed at one end, through default settings that can be changed, and on to the notional 'free choice' of full customisation at the other[41] (but it must be remembered that even this level of configurability can never be completely 'free', because as I noted above the initial commitments of the design by definition circumscribe possibilities, which in turn limits the area within which the end-user can exercise autonomy). The extent to which end-users are aware of the control they have over configuration is a core concern,[42] and is entirely contingent upon the affordance of choice being perceived – it is not enough if the affordance is merely real[43] but unknown (or so complex as in practice to disafford[44]). The authors identify two apparent motivations driving real-world design decisions about setting defaults, namely efficiency and the consideration of novice end-users.[45] They note the vagueness of these goals, particularly with regard to who might be considered a novice, and by whose standards 'efficiency' is to be judged, particularly since

[40] RC Shah and C Sandvig, 'Software defaults as de facto regulation: The case of the wireless internet' (2008) 11 *Information, Communication & Society* 25, 42.

[41] Kesan and Shah (n 18) 591 *et seq*. See also LF Cranor and RN Wright, 'Influencing software usage' (1998) arXiv:cs/9809018, 6–7 <http://arxiv.org/abs/cs/9809018> last accessed 4 March 2021.

[42] Kesan and Shah (n 18) 597.

[43] These distinctions were discussed in 'Real and Perceived Affordance' in Section 2.1.

[44] This is a common criticism levied at open source projects, whose configurability gives the end-user great notional power but whose ease-of-use may in practice fail to afford that configurability to those without the necessary expertise. See K Noyes, 'Is Linux really harder to use?' *PCWorld* (2 August 2010) <https://www.pcworld.com/article/202364/is_linux_really_harder_to_use.html> last accessed 4 March 2021.

[45] Kesan and Shah (n 18) 600.

the effect of a default will often impact on 'fuzzy' values that are difficult to calculate in such terms.[46] The latter principle plays a central role in programming practice; we will see below in the discussion of immediacy and the affordance of delay how the goal of increasing or improving 'efficiency' – pervasive among technologists – is not necessarily desirable in every case, even where there is technical scope for achieving it.

In terms of guiding design, Kesan and Shah draw on legal notions of default rules to consider the threshold between what configuration is 'wired in' (immutable) and what is set merely as a default.[47] Where the configuration is 'wired in', it must both notify the end-user and permit a judicial remedy (that is, be contestable). This accords with the affordances of contestability, transparency, and delay. The system ought to provide 'an easy-to-use interface that allows users to configure the software according to their preferences'.[48] This again is of course about the affordances of that interface, which should follow the design and usability conventions end-users are generally familiar with.[49] Added to this, a framework of principles guides designers in setting the initial defaults, the starting point of which is the 'would have wanted' standard. This requires anticipation of what the parties (the enterprise and end-user) would likely have negotiated, had that been a possibility.[50] This principle applies in situations where the setting does not materially affect a fundamental societal concern, such as (they suggest) privacy or online security. To those I might add normative legitimacy, with all the elements described here.

The next requirement is that where there is an information imbalance between designer and end-user, the default must be set to protect the latter, with appropriate information or guidance provided to her should she wish to change it. This is of course inconvenient if the functionality in question is the very purpose of the device. Kesan and Shah discuss the cookie settings in a web browser, arguing that the imbalance of information[51] means the default setting should be to reject them.[52] Should web companies wish them

[46] Ibid.

[47] Ibid. 614 *et seq*.

[48] Ibid. 615–16.

[49] Norman calls these 'conventions'. See DA Norman, 'Affordance, conventions, and design' (1999) 6 *interactions* 38, 40 *et seq*.

[50] Kesan and Shah (n 18) 618.

[51] That is, end-users do not readily understand what cookies are and what they are used for. See L Edwards, 'Data protection and e-privacy: From spam and cookies to big data, machine learning and profiling' in L Edwards (ed.), *Law, Policy and the Internet* (Hart 2018) 126 *et seq*.

[52] This is at odds with current practice. See the discussion ibid. and n 134 in Chapter 3 and its accompanying text.

to be enabled, presumably to facilitate the (increasingly discredited) business model of online behavioural advertising, they then must explain to the end-user their purpose and how to enable them[53] (this relates to the affordance of transparency of purpose, discussed below). The idea behind such 'penalty defaults' is that before the end-user can choose the non-default setting, the burden is on the designer to explain its effect. The default is therefore what the party with the greater understanding of the code (the designer or enterprise) would not have wanted, as a delaying mechanism that allows for the end-user to be informed (this is connected with 'friction' and the affordance of delay, discussed below).

The next principle suggested by Kesan and Shah also justifies this 'would not have wanted' standard, based on the economic concept of 'externalities', the broader (potentially negative) effects of the system on third parties. The default setting should reduce externalities or, if the stakes are particularly high, there should be no latitude and the beneficial option should be 'wired in'.[54] An example might be an IoT webcam that is configured by default to stream whatever it captures, with no authentication mechanism enabled by default, or a generic default password such as 'admin'. The negative effects of such designs, especially given the potential pervasiveness of such technology, can be significant.[55] The idea is that although it is convenient for the end-user (and therefore commercially attractive) if the camera starts working immediately, the 'would not have wanted' standard might require that streaming is not enabled by default and that a (strong) password must be set before the device will connect to the network. Anecdotally, this has been the direction of travel in the design of domestic routers, where instead of merely suggesting that end-users change the administration or WiFi authentication passwords/keys, the device is preconfigured with unique and strong options set at the factory.

In terms of design, the cognitive biases mentioned in Chapter 3[56] can strengthen the 'stickiness'[57] of a default setting, militating against the end-user exercising choice; this implies an even greater responsibility to design

[53] Kesan and Shah (n 18) 621.

[54] Ibid. 621–2.

[55] For chilling real-world examples of precisely this, see JM Porup, '"Internet of Things" security is hilariously broken and getting worse' *Ars Technica UK* (23 January 2016) <http://arstechnica.co.uk/security/2016/01/how-to-search-the-internet-of-things-for-photos-of-sleeping-babies/> last accessed 4 March 2021.

[56] See 'Default Configurations' in Section 3.2.

[57] CR Sunstein and RH Thaler, 'Libertarian paternalism is not an oxymoron' (2003) 70 *University of Chicago Law Review* 1159, 1175.

that initial configuration with the appropriate set of interests in mind.[58] Furthermore, the prominence of the setting in an interface can affect end-users' awareness of it, and indeed the designer can explicitly draw attention to defaults that require special attention but do not cross the threshold to merit being 'wired in' (for example the 'would not have wanted' defaults just discussed – making the administration password for a domestic router 'wired in' would be an odd design choice). Attention can be drawn by, for example, alerts requiring the end-user to confirm a choice. She might also be required to make a choice when the device is first used, with no preselected option to influence her decision or option to bypass the configuration request. Such measures can contribute to the beneficial form of delay I discuss below. Importantly, the design of these affordances of choice must take into account 'framing effects',[59] or the way in which wording affects comprehension of one or other of the available options. The design should not promote the enterprise's aim at the expense of digisprudential legitimacy, and of course the use of the adversarial design approaches that we saw in Chapter 2[60] is *de facto* illegitimate.

This analysis of choice *qua* configurability gives added texture to bare suggestions that technological normativity preserve the possibility of choice, and that more choice is *per se* better. The affordances of the artefact ought to reflect the spirit of digisprudential legitimacy at each point of the end-user's 'journey' through the artefact's inscriptions. More choice is not sufficient on its own to legitimate code if it is not the right kind of choice; the design must afford meaningful spaces for the exercise of autonomy and not simply more options. This might raise difficult existential questions as to the desirability of a given artefact or business model, but that of course is precisely the point.

Blockchain Applications

Blockchain applications pose potentially significant problems from the perspective of affording choice. We saw in Chapter 1 how one of their primary attractions is the potential immutability of the code, owing both to how it is stored on the chain and the decentralisation of the network. If the possibility of choice has not been anticipated in advance, the central benefit to blockchains of tamper-resistance becomes a hindrance to its exercise at runtime. To afford choice, then, it is especially necessary to choose carefully in advance how much of the code is ruleish and how much relies on external contingency, including end-user input. This is intimately connected to, and

[58] Kesan and Shah (n 18) 633.
[59] Sunstein and Thaler (n 57) 1179–83.
[60] See 'Disaffordance' in Section 2.2.

overlaps with, the issue of immutability, discussed below: the fixity of the code that flows from its storage on, and execution by, a blockchain means that the design of the threshold between wired-in and configurable code is centrally important, particularly given the additional complication of blockchain applications' execution of logic that is automatic and has potentially legally relevant implications.[61]

If blockchain applications are to be used to implement legally relevant operations such as transfers of assets, one approach to ameliorating ruleishness is to reconceive of them as mere 'custodians' of the multi-interpretability of language, and thus to change emphasis on what ought to be implemented computationally. The Ricardian Contract, for example, aims not to automate the purposive elements of a contract-like agreement (the goal of some smart contract maximalists), but rather to maintain the flexibility of textual agreement and to augment it with a limited amount of coded functionality that complements, rather than replaces, the latter.[62] The (natural language) text of the agreement is 'wrapped' in a minimal code semantics that enables basic code-based features such as tamper-resistance and provenance checking, through the use of hashing and public key cryptography. Because the actual text of the agreement retains all the possibilities of multi-interpretive nuance that natural language accommodates,[63] the notional immutability of the agreement can nevertheless be combined with the inherent flexibility of expression. In other words, the execution of the agreement remains 'human'; the contribution of code as a medium is in providing the benefits, ancillary to the substance of the agreement, of immutability and provenance checking. Any exercise of choice, then, takes place outside of the code. The extent to which this is practically useful from the perspective of the technology remains to be seen; limiting the ruleishness of blockchain code to such ancillary benefits in this way might in practice undermine their perceived value in the first place – but the benefit of avoiding a computationally legalistic outcome is evident.

Other approaches that maintain the full(er) instrumental utility of code will require the design to afford the end-user choice at key moments. The

[61] KEC Levy, 'Book-smart, not street-smart: Blockchain-based smart contracts and the social workings of law' (2017) 3 *Engaging Science, Technology, and Society* 1, 3.

[62] I Grigg, 'The Ricardian Contract' in *Proceedings of the First IEEE International Workshop on Electronic Contracting* (IEEE 2004). This chimes with Wagner's argument in favour of 'more law and less software'. See RP Wagner, 'On software regulation' (2005) 78 *Southern California Law Review* 457.

[63] M Hildebrandt, 'The adaptive nature of text-driven law' (2021) 1 *Journal of Cross-disciplinary Research in Computational Law* <https://journalcrcl.org/crcl/article/view/2> last accessed 19 April 2021.

artefact's inscriptions then need to be sensitive to the architectural implications of blockchains – their particular brand of technological normativity and *de facto* immutability from the perspective of the end-user. The greater the normative impact of the code's logic (its normative density), the greater the need for choice to be preserved; in practice the implementation of this will vary, involving a mixture of notifications to the end-user, the appropriate selection of oracles, appropriately defined choices, and logic that can deal appropriately with the outcomes. Given the peculiar characteristics of blockchains, whether designers can anticipate all the relevant points where choice is necessary is at the very least questionable. Indeed, these requirements may fundamentally undermine the premise of blockchain applications, particularly those (such as distributed autonomous organisations) predicated on their ability to execute operations with minimal or no human input. The existential question is thus raised again of whether such applications can be legitimate *a priori*.

The Internet of Things

A common characteristic of IoT devices is a minimal or even absent interface. One way of affording choice is to provide better, more sophisticated interfaces, perhaps through the connection of the IoT artefact to another device that itself affords more complex interactions. This could be a smartphone or a television, which in turn facilitates the presentation and signifying of choice affordances to the end-user. This is a difficult balance to strike, because of course many IoT devices are intended to have a minimal number of functions. In the next section I discuss the Amazon Dash Button, which consists of a single button but whose background functionality is extremely complex; in that case the affordance of choice at the point of use is dramatically curtailed, this being central to its *raison d'être* – its only real use is the pressing of a button, but the number of configurable variables that are relevant to the process that is put in train by doing so is significant, as we will see.

(b) Opacity

In the computational context opacity is connected most closely with the Fullerian principles of promulgation and intelligibility, and the legisprudential principles of alternativity (PA) and normative density (PN). In terms of promulgation, Fuller is concerned that citizens should know (or be in a position to find out) the rules by which they are governed, in part as a check against them being disregarded by the authorities administering them (this relates also to his eighth principle, requiring congruence between the declared rule and the official action that flows from it). This of course enables citizens to observe their operation, a prerequisite for contesting it. In order to

be valid, the rules should also be intelligible; obscurity and incoherence can make legality difficult or impossible to attain.

Under legisprudence, opacity is targeted by the PA and the PN. Under the former, the inherent opacity of code again imbues the decision to implement a rule with extra normative impact as compared with a less ruleish measure. The inability of the end-user to see the rule she is subject to is emphasised in the computational context, and so its use is subject to a higher threshold of justification. As before, that threshold is lowered when a less 'ruleish' design measure is employed, but at all times the fact of code's opacity must also be kept in mind. This in turn speaks to the PN: the more opaque the code, the more difficult it may be for the end-user to appreciate the aggregate 'density' of technological normativity her behaviour is subject to. The PN expects there to be a proportionate connection between the policy aim and the means by which it is achieved, with threats of sanction at the 'denser' end of the scale and mere suggestions towards the 'lighter' end. In terms of justification, the use of a particular design technique must be justified in the context and in light of the other principles, particularly if there are alternative mechanisms that might have achieved the same end in a more digisprudentially legitimate fashion. We have seen how the geography of code is often taken by the end-user to be a 'natural fact', rather than merely one possibility among infinite others.[64] This opacity of normative impact is particularly strong where the configuration of (dis)affordances and inscriptions strongly guides the end-user's behaviour; the need to legitimate such impositions is all the stronger in such situations.

Digisprudential Affordance: Transparency of Provenance

An important aspect of affording transparency, connected to the affordance of contestability, is that of provenance. This can be problematic when even apparently simple digital systems are a bricolage of multiple components, often from different sources[65] – in many, perhaps most, cases, code artefacts are 'a mix of a Frankenstein and a Matryoshka doll concealing dozens of services'.[66] Designers ought to afford reasonable notice of the sources of the code

[64] See the affordance of choice above, and 'Default Configurations' in Section 3.2.

[65] P Swartz, 'White boys' code' in *Division III: Essays in Programs as Literature* (Hampshire College 2007) 34–6; 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* (Cambridge University Press 2018); D Oberle et al., 'Engineering compliant software: Advising developers by automating legal reasoning' (2012) 9 *SCRIPTed* 280.

[66] Gürses and van Hoboken (n 65) 584.

in their systems (or at least provide a means to find out[67]), so as to inform and to afford contestability.[68] Recent scrutiny of online behavioural advertising has shown just how large the network of unseen third parties operating in the background can be, including situations where the design of a website's interface might suggest to the end-user that there is only one provider involved.[69] The same is often true of other digital artefacts whose back-end processing relies on a host of services (and third-party code) that the end-user is unlikely to be aware of (I return to the theme of bricolage in code production in Chapter 7). Providing access to this information is a necessary part of facilitating *transparency of provenance*.

Digisprudential Affordance: Transparency of Purpose

The bricolage of code's provenance in turn raises the question of its purposive functionalities. From a legitimation perspective, there is a connection here between what the code is designed to do and the third ('environmental') level of coherence in the legisprudential scheme. According to the latter, a rule is never justified on purely internal legal grounds, but must be supported by some external, non-legal theory that independently justifies its character. In the code context, transparency under this rubric will require information as to the reason for a given piece of functionality, where this is not self-evidently the artefact's *raison d'être* – in other words, unexpected functionality must be justified according to something other than the commercial rationality, internal to the perspective of the designer-cum-legislator, of profit maximisation. We saw above the smart thermometer whose design creates a tussle of interests, between regulating the end-user's domestic heating and profiling her habits for profit, the latter functionality not being something reasonably within the scope of what the end-user would expect from a thermostat. The normativity of such functionality ought to be justified via the affordance of transparency, and where this cannot be done the function should not be included in the design. Another real-world example is the inclusion of a geolocator in a smartphone torch application – transparency ought to be

[67] I discuss the contrasting ideas of the 'monitoring citizen' versus the 'well-informed citizen' below.

[68] One method of charting the sources of third-party code involved in a project is known as a 'dependency graph'. See for example 'Exploring the dependencies of a repository' *GitHub Docs* <https://docs.github.com/en/enterprise-server@2.22/github/visualizing-repository-data-with-graphs/exploring-the-dependencies-of-a-repository> last accessed 4 March 2021.

[69] See for example Z Yu et al., 'Tracking the trackers' in *Proceedings of the 25th International Conference on World Wide Web – WWW '16* (ACM Press 2016). The authors of the study found that 95 per cent of websites accessed by its German participants contained potential third-party trackers.

afforded on a similar basis, because determining the end-user's location is by no means a standard function of a torch.[70] The designer must then consider from where this unorthodox function of geolocation is justified, given the affordances commonly expected of torches. This form of transparency might be termed *transparency of purpose*.[71]

With each form of transparency, the designer must not succumb to the transparency fallacy, where essentially any function can be included in the code provided the end-user is given ostensible 'notice and choice'.[72] In that vein, Pols and Spahn suggest the 'monitoring citizen' as a better normative ideal than the 'well-informed citizen' that such practices envisage.[73] While the notion of a fully informed end-user might in principle be desirable, the complexity and pervasiveness of code means it can only ever be a mirage.[74] The 'monitoring citizen', on the other hand, is empowered to find out information when she needs to: '[t]he Monitoring Citizen does not *know* everything that is going on but can *monitor* it successfully and can investigate and contest policy when needed.'[75] This ideal is more plausible than aiming for some notion of 'real' or 'full' transparency, and it provides a guiding principle for design: the aim is to empower the end-user by affording her access to appropriate information about the operation and purposes of the code she is interacting with. One can think here of an analogy with legislative procedure: in addition to the citizen being able to access directly the legislative rule in the statutory document (cf. Fuller's first principle), it is also possible for her to access explanatory notes, impact assessments, Hansard, and other ancillary material in order to glean a deeper understanding of the purpose of a piece of legislation.

[70] Hartzog (n 25) 24. See also Federal Trade Commission, 'Android flashlight app developer settles FTC charges it deceived consumers' (Federal Trade Commission, 5 December 2013) <https://www.ftc.gov/news-events/press-releases/2013/12/android-flashlight-app-developer-settles-ftc-charges-it-deceived> last accessed 4 March 2021.

[71] Here we begin to touch the edges of the fields of computer ethics and responsible research and innovation. These lie beyond the legal-philosophical analysis of technological normativity that I am principally concerned with, although the connection is worth exploring further. See for example S Vallor, *Technology and the Virtues: A Philosophical Guide to a Future Worth Wanting* (Oxford University Press 2016); von Schomberg (n 6); L Floridi, *The Ethics of Information* (Oxford University Press 2013).

[72] Hartzog (n 25) 68 *et seq*.

[73] Pols and Spahn (n 33) 348.

[74] Such a goal also has the effect of shifting responsibility away from the enterprise and onto the citizen as *homo economicus*, a neoliberal orthodoxy that digisprudence seeks to counter.

[75] Pols and Spahn (n 33) 348 (emphasis supplied; references omitted).

Digisprudential Affordance: Transparency of Operation

The most obvious mechanism here is transparency in the imposition of normativity, in the form of documenting the use of a design that lies at a particular point on the normativity spectrum or actively informing the end-user of this fact. As we saw above, however, transparency is a contested concept. In the context of machine learning systems, it has been criticised as a means by which engineers can 'whitewash' decisions that are antagonistic to end-users' interests;[76] similar concerns apply in the code-driven context. Transparency as a goal can be framed such that including descriptions of functionality in lengthy terms documents that notionally inform the end-user is legitimate, when in practice it does little to illuminate what is actually going on.[77] The (ideological) belief is that simply providing more information will enable end-users to make informed choices about which products can and cannot fulfil their preferences, thus leading to greater competition and better products by dint of the operation of the market[78] (see the normative distinction between the 'monitoring citizen' and the 'well-informed citizen' just discussed).

Other work argues for solutions that do not involve the end-user directly. For example, the source code that underpins regulatory software systems could be required to be open – viewable – in order that it can be audited by third parties.[79] Proposals of this kind may be plausible for public sector regulators,[80] but as the 'code wars' in the late 1990s to mid-2000s showed, commercial enterprise has been reticent if not actively hostile to the idea of opening up the proprietary code at the core of its products and services.[81] Others have suggested an escrow system, where an artefact's source code is

[76] See for example L Edwards and M Veale, 'Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for' (2017) 16 *Duke Law & Technology Review* 18.

[77] Hartzog (n 25) chapter 2; M Piekarski and W Wachowski, 'Artefacts as social things: Design-based approach to normativity' (2018) 22 *Techné: Research in Philosophy and Technology* 400, 414–15.

[78] O Ben-Shahar and CE Schneider, 'The failure of mandated disclosure' (2011) *University of Pennsylvania Law Review* 647.

[79] DK Citron, 'Open code governance' (2008) *University of Chicago Legal Forum* 355; L Lessig, *Code: Version 2.0* (Basic Books 2006) chapter 8. Both authors are specifically concerned with abuse by public institutions.

[80] The European Commission, for example, follows such a mandate. See its 'Open Source Software Strategy' (European Commission 2019) <https://ec.europa.eu/info/departments/informatics/open-source-software-strategy_en> last accessed 4 March 2021.

[81] This may be changing, however. See for example E Angelova, 'Microsoft embraces open source' (2018) *Fordham Intellectual Property, Media & Entertainment Law Journal* <http://www.fordhamiplj.org/2018/11/28/microsoft-embraces-open-source/> last accessed 4 March 2021.

held by a trusted third party who can be required to release it by a court if litigation should arise.[82]

Neither of these suggested approaches takes into account the full context and texture of the code's materiality. (Dis)affordance and inscription speak to more than just the bare logic of the code, so while it is true that a great deal can be gleaned from code by studying it (I will discuss some possibilities in Chapter 7), a broader sensitivity to qualitative design concepts is required to fully appreciate its effects in operation, particularly those that impact the end-user. Perhaps more importantly, such approaches are not based on input legitimacy, because they operate as a kind of insurance policy to be invoked after malfeasance has been suspected or detected. Relying on ex post assessment does nothing to avoid the production of illegitimate code in the first place; by the same token, if no issues are detected, the harmful code will simply continue to operate, potentially indefinitely.

The goal of transparency in this context should not, therefore, be limited to the literal openness of source code. As we saw in Chapter 2, design can signify to the end-user what the functionality of the system is and what it allows her to do.[83] What matters is comprehension, not just notification, and so it is incumbent on the designer to ensure as far as reasonably possible that the end-user's mental model of the system matches what it actually does.[84] This model is constructed from various sources, to a greater or lesser extent under the designer's control, including advertisements, press releases, instruction manuals, and of course the artefact's interface itself. Empathy with the end-user also requires acknowledgement from the designer that, with her intimate knowledge of the system's operation, her own conceptual model is likely to differ significantly from the situated (and necessarily less informed) understanding of the end-user.[85]

Similar considerations arise in relation to the legisprudential principle of coherence (PC). In terms of internal coherence (levels 0 and 1), the code should be consistent in its design language (cf. the grammar and basic meaning of words under the PC), and it is the designer's role to ensure the artefact

[82] Cf. JL Mezrich, 'Source code escrow: An exercise in futility' (2001) 5 *Marquette Intellectual Property Law Review* 117.

[83] See also Hartzog (n 25) 27.

[84] Ibid. 278; DA Norman, *The Design of Everyday Things* (MIT Press 2013) 26, 31.

[85] LA Suchman, *Human–Machine Reconfigurations: Plans and Situated Actions* (2nd edn, Cambridge University Press 2007) chapter 11; PE Agre, 'Conceptions of the user in computer systems design' in PJ Thomas (ed.), *The Social and Interactional Dimensions of Human–Computer Interfaces* (Cambridge University Press 1995); Norman, *The Design of Everyday Things* (n 84) 31.

is 'understandable and usable'.[86] To avoid misapprehension, end-users should not be confronted with conflicting or inconsistent design. In terms of the first level of coherence, this includes arbitrary changes that can confuse or trick the end-user, especially after she has developed a familiarity with how the code works.

This is *transparency of operation*, or designing in such a way that the end-user can, within reason, understand what the code is doing as it does it. This form of transparency is an ongoing concern, linked with the affordance of oversight, discussed below. Because many systems are frequently updated with features added and removed, it is incumbent on the designer to appropriately communicate such changes where they 'reconfigure' the relationship between end-user and enterprise.[87]

Blockchain Applications

On public blockchains, the code of an application can generally be viewed by anyone. From a transparency perspective this potentially repeats the problem of source code transparency described above – having access to the application's code does not automatically render it intelligible to those likely to be affected by its operation. Various initiatives in the cryptocurrency community seek to ameliorate this problem. For example, the 'Ethereum Natural Specification Format' ('ENSF')[88] is a form of code commentary that allows the designer to descriptively tag the elements of an Ethereum application, from which a natural language explanation of the application's operation can be automatically generated. The result is a commentary of the blockchain application, for example:

```
Send 1.125 BTC from the account of ABC to an account
   accessible only by XYZ
```
[89]

This message is generated from the following tags, immediately preceding the actual code performing the action:

```
/// @notice Send `(valueBTC / 1000).fixed(0,3)` BTC
   from the account of `message.caller.address()` to
   an account accessible only by `to.address()`
```

[86] Norman, *The Design of Everyday Things* (n 84) 32.

[87] Gürses and van Hoboken (n 65) 594.

[88] Ethereum Foundation, 'Ethereum Natural Specification Format' in *The Ethereum Wiki* (Ethereum Foundation 2018) <https://github.com/ethereum/wiki/wiki/Ethereum-Natural-Specification-Format> last accessed 4 March 2021.

[89] This and the next example are simplified versions of those found in ibid.

One can see how the elements between the backticks (`) are placeholders for the actual values generated within the application's logic, from which the accessible commentary above can be derived.

This provides a measure of transparency of operation, in that the logic of the blockchain application can theoretically be explained to the end-user. A continuing problem, however, is that such approaches rely on the designer's subjective understanding of the code. For the approach to work she must accurately model, in a combination of natural language and code placeholders, the logic of the application. If she fails to do this, intentionally or mistakenly,[90] the end-user might end up with both an erroneous understanding of the system and a misplaced confidence in that understanding, an outcome which is arguably less desirable than if there were no explanation at all. The result of this is similar to the earlier-noted tendency of legal scholars to focus on the decisions which led to the use of code, rather than the normativity that it actually implements. Descriptions of this sort add an additional layer of interpretation between the end-user and the normativity of the code, increasing the likelihood of errors and misinterpretations on the part of both the designer and the end-user. In this case, the re-emergence of the hermeneutic gap, this time between the commentary-text and the code's instrumentality, is no welcome thing.

Progress might be made by leveraging the programmer of the programmer, in the form of the integrated development environment (IDE) detecting instances during code writing where such tags might be included in the code and to suggest the designer add them. I discuss this further in Chapter 7.

The Internet of Things

Affording transparency in the IoT is a complex challenge for several reasons. IoT devices are often intended to be embedded and pervasive, creating a network of devices that communicate with one another to create 'ambient intelligence' or 'ubiquitous computing'.[91] As a consequence of such devices 'receding' into the background of everyday life, they often have either minimal (or no) interfaces with which the end-user can interact in order to observe what is actually going on. Many IoT devices offer few or even no perceptual affordances, and with such minimal means of communicating their presence

[90] As noted above at n 14, even safety-critical applications often have software errors.

[91] M Weiser, 'The computer for the 21st century' (1991) *Scientific American* 94.

and/or purposes to the end-user, the opacity of their normativities is all the more impenetrable. As Matassa and Simeoni warn,

> the existing affordances in connected and technologically augmented objects are becoming unable to immediately communicate to people their actual values and meanings . . . The impossibility of establishing a clear connection between objects and functionalities could become a threat for humans, since they are missing their innate ability to understand what they can do only based on their knowledge and perception of the surrounding context.[92]

This relates to real and perceived affordances, discussed in Chapter 2. The invisibility of IoT devices, and/or their minimal interfaces, means that the communication of even perceived (dis)affordance is already limited, much less the real (dis)affordances embodied in the design. The scope then is all the greater for the end-user to experience dissonance between what she thinks is happening, what her possibilities for action are, and what is actually taking place.[93]

It may be, then, that to achieve legitimacy an IoT device must be designed actively to facilitate understanding on the part of the end-user, even where this is not necessary for the product's purpose to be achieved.[94] The normativity of the artefact should on this account be made apparent to the end-user. IoT devices are hybrids – they combine the up-front physicality of a tangible object with the background processing of (networked) code, the latter being made even less tractable by the absence of an interface.

I mentioned earlier that one means of facilitating intelligibility is via a separate device (a smartphone or television) that provides an interface through which the user can interact with and monitor the IoT device. 'Smart' thermostats and doorbells follow this approach. The degree to which the obscured (dis)affordances/inscriptions embodied in the device are communicated to the end-user will vary according to the complexity of the device's functions. For example, after it is configured, the Amazon Dash Button provides the most minimal of interfaces: a simple adhesive push-button which when pressed reorders the product indicated by a logo on its surface.[95] The apparent

[92] A Matassa and R Simeoni, 'Eliciting affordances for smart objects in IoT era' in *Internet of Things: User-Centric IoT* (Springer 2015) 77–8.

[93] Ibid. 78.

[94] As Robertson suggests, designers have a responsibility to facilitate end-user agency, which can only be achieved through the provision of 'resources for awareness'. See T Robertson, 'The public availability of actions and artefacts' (2002) 11 *Computer Supported Cooperative Work (CSCW)* 299.

[95] Amazon, 'Amazon help: Set up your Dash Button' <https://www.amazon.co.uk/gp/help/customer/display.html/ref=amb_link_1?nodeId=201746340> last accessed 4 March 2021.

simplicity of the single-button interface belies the black-boxing of the complex set of operations involved: infrastructurally, there are multiple networking processes (WiFi connection, TCP/IP and HTTPS handshaking, and authentication of the connected Amazon account via the company's API), a financial transaction (communication between Amazon's servers and the provider of the end-user's bank account, usually one or more third-party payment processors), and of course the generating of data deepening Amazon's profile of the end-user's preferences and shopping habits. The gap between the simplicity of the device and what actually goes on demonstrates the dissonance referred to by Matassa and Simeoni, between the purposive end of the code and what it is actually doing. In Fullerian terms, the 'rule' as declared may not be congruent with 'official' action: the Dash Button may well 'reorder dishwasher fluid', but what does this tell us about the layers of activity that pressing the button in fact sets in motion?

(c) Immediacy

The immediacy of code is especially problematic when combined with the contradictory or impossible rules Fuller warns of in his fifth and sixth principles. In the code context, design language can be confusing to the end-user at the level of the interface if it lacks consistency. Impossible rules can guide end-users into situations where there is no logical way out. For example, website cookie notices often give the illusion of providing the choice of whether or not to consent but in reality require acquiescence in order to gain access. Frequent changes to the code can also be problematic – end-users can become accustomed to one way of working with an artefact then find this being changed or reversed by a software update. Depending on the kind of artefact, the scope for such changes can vary; changes to the design of online platforms' interfaces have often disoriented end-users to the point of backlash.[96] Beneath the surface of code, alterations to functionality can also have important effects: the periodic tweaks to Google's search algorithm significantly alter what material is found on the web, with reflexive societal implications.[97]

From a legisprudential perspective, code's immediacy invokes the principle of normative density. The immediate imposition of a given normative

[96] For example, one might recall Facebook's move to a 'news feed' layout in the mid-2000s. See J Leyden, 'Users protest over "creepy" Facebook update' *The Register* (7 September 2006) <https://www.theregister.co.uk/2006/09/07/facebook_update_controversy/> last accessed 4 March 2021.

[97] See Moz, 'Google algorithm change history' *Moz* (2018) <https://moz.com/google-algorithm-change> last accessed 4 March 2021.

configuration makes ex ante consideration of its design especially necessary. We saw above in the discussion of opacity how the density of technological normativity is a crucial concern; the immediacy of its application heightens this. This in turn implicates the legisprudential principle of temporality, requiring sensitivity to the moment of the imposition of normativity and ongoing justification to ensure the mechanism for achieving the aim of the norm continues to be appropriate in light of the other principles.

Digisprudential Affordance: Delay

Speed and immediacy of execution are quintessential elements of computational legalism. As we have seen, the design of the artefact reflects an 'intentionality' which reflects some normative conception of how things ought to be done.[98] As Floridi suggests, in the computational era the lack of 'informational friction' contrasts with preceding historical periods where the inherent makeup of the social fabric meant that information could not travel above a certain speed or beyond a certain geographical radius.[99] For him, information privacy is facilitated in part by the 'ontological friction' within a system, which operates to oppose the flow of information and increase the effort required to gain access to it.[100] This chimes with Hildebrandt's arguments to the effect that the affordances of text as a medium are what have resulted in the existence and character of law as we know it.[101] When instantiated in text, the meaning of legal norms is under-determined, but their expressions are nevertheless stable enough to facilitate the understanding and consensus (always contingent and defeasible) that can, through incremental democratic evolution, respond to societal change.[102] The affordances of text as a technology open up spaces for these processes to take place.

[98] Recall the discussion of Ihde's comparison of a fountain pen and word processor in 'Code Mediating Action' in Section 2.2.

[99] L Floridi, *The Fourth Revolution: How the Infosphere is Reshaping Human Reality* (Oxford University Press 2014) chapter 5. See also W McGeveran, 'The law of friction' (2013) 2013 *University of Chicago Legal Forum* 15.

[100] Floridi, *The Ethics of Information* (n 71) 231 *et seq*.

[101] M Hildebrandt, *Smart Technologies and the End(s) of Law: Novel Entanglements of Law and Technology* (Edward Elgar Publishing 2015) *passim*. See also WJ Ong, *Orality and Literacy: The Technologizing of the Word* (3rd edn, Routledge 2012); J Goody, *The Logic of Writing and the Organization of Society* (Cambridge University Press 1986); J Goody and I Watt, 'The consequences of literacy' (1963) 5 *Comparative Studies in Society and History* 304.

[102] Hildebrandt, *Smart Technologies* (n 101) chapter 3. I have contrasted Hildebrandt's conception of institutional law as an affordance of text and the printing press with the idea of code's affordances being compatible with the substantive requirements of the law. See L Diver, 'Law as a user: Design, affordance, and the technological mediation of norms' (2018) 15 *SCRIPTed* 4, 30 *et seq*.

Such spaces are in principle susceptible to conscious implementation by the designer. Where the technology is particularly plastic – like code – this requires a conscious and serious commitment, particularly when the presumption amongst technologists is that 'inefficiency' and 'friction' are undesirables that *a priori* operate in opposition to the end-user's aims. Such positions perhaps betray a market-centred rationality that presumes too much about the values the end-user holds, both instrumental and intrinsic, and how these can and ought to be reflected in system design. What matters is identifying the points at which purposely avoiding this instrumental notion of 'efficiency' is necessary for the protection of some broader value.[103] I am therefore not advocating a naïve approach to inefficiency, where designers simply stop optimising their code. This would be arbitrary in any case, since the forms of delay that were retained or introduced would be contingent on both how they arise 'naturally' within a given system and the designer's expertise and conscientiousness in identifying an ameliorating them. It might also be irresponsible, where the lack of optimisation undermines what should be universal goals, such as reducing unnecessary energy consumption. The point, rather, is to identify what matters for a given value, and consciously to implement delays in the code's inscriptions that facilitate respect for that value.

DESIRABLE INEFFICIENCY

In recent work Ohm and Frankle posit the notion of 'desirable inefficiency',[104] where code's efficiency (its ruleishness and its immediacy) is consciously tempered to protect some value that might otherwise be undermined. For them, efficiency in computer science is 'the extent to which [code] minimizes the consumption of time, energy, space, or cost in satisfying a specification of correctness for a given problem'.[105] A desirably inefficient approach is one that sacrifices this goal in service of solving some other problem.

The 'basic problem' is the technical outcome the designer seeks, while the 'enhanced problem' is one requiring 'human judgment, values, or discretion in the definition of success or failure'.[106] Sometimes the latter requires

[103] I discuss the normative role of delay in the context of legal practice in L Diver, 'Computational legalism and the affordance of delay in law' (2021) 1 *Journal of Cross-disciplinary Research in Computational Law* <https://journalcrcl.org/crcl/article/view/3> last accessed 19 April 2021. See also C Storni, 'The problem of de-sign as conjuring: Empowerment-in-use and the politics of seams' in *Proceedings of the 13th Participatory Design Conference on Research Papers – PDC '14* (ACM Press 2014).

[104] P Ohm and J Frankle, 'Desirable inefficiency' (2019) 70 *Florida Law Review* 1.

[105] Ibid. 28.

[106] Ibid. 32.

the conscious imposition of 'inefficiency', making space for a human to do something that only a human can. The authors argue in favour of desirable inefficiency as a set of design patterns, part of a call for a 'new interdisciplinary research agenda investigating how values can be embedded into code'.[107] An example is a smartphone's passcode screen, which locks for a progressively longer period when incorrect attempts are registered. Designs like this balance the inconvenience that a forgetful end-user experiences with the security of the device that might otherwise be compromised in the hands of a thief.[108] The technical 'basic problem' is providing the end-user with secure access to her smartphone, while the societal 'enhanced problem' is preventing access to thieves, and in turn the disincentivising of smartphone theft.[109] Blockchain proof-of-work is another example,[110] where what would otherwise be near-instant (storing a transaction's outcome in a database) is made sufficiently 'inefficient' to allow reintroduction of values of trust and 'clock time'.[111] The basic problem is tamper-resistant validation of transactions, while the enhanced problem is their fair validation.[112]

Ohm and Frankle's analysis focuses on the underlying logics of computation, explicitly excluding designs that 'do no more than slow down the operation of a computer to match the speed of human processing systems'.[113] There are myriad circumstances, however, in which the service of 'human processing systems' – that is, humans – is ultimately what matters. Indeed, at the heart of what makes many if not most 'enhanced problems' enhanced will be humans, or human interests. There is therefore value in applying the notion of desirable inefficiency to the design of end-user-facing code, especially where doing so can help to facilitate another human value such as respect for autonomy. Even where greater efficiency is possible from a technical perspective,

[107] Ibid. 5.

[108] Ibid. 15. Or indeed the authorities; recall attempts by the US Department of Justice to compel Apple to unlock a phone belonging to a perpetrator of the 2015 San Bernardino shooting. The phone was on the verge of wiping its memory because the passcode had been entered incorrectly numerous times. See B Bailey, 'Apple vs. FBI – what happened?' *Associated Press* (29 March 2016) <https://apnews.com/article/c8469b05ac1b4092b7690d-36f3409a4a> last accessed 4 March 2021.

[109] Ohm and Frankle (n 104) 29–30.

[110] This is the 'mathematical challenge' discussed in 'Blockchain Design' in Section 1.1.

[111] Ohm and Frankle (n 104) 19–22. Proof-of-work is often extremely wasteful of energy, but that is inefficiency of an undesirable kind.

[112] Ibid. 29–30.

[113] Ibid. 35–6. Ohm and Frankle describe humans as 'unpredictable and fiddly devices', apparently akin to other peripherals such as printers and scanners, for whose benefit the computer must limit its intrinsic speed.

in some cases it will be better to opt for a less efficient design where doing so makes it possible to separate points at which the artefact's design implicates diverging or conflicting interests (this is the idea of tussle discussed above).[114] The goal, then, is to frame slowness and 'inefficiency' as potentially beneficial features rather than as tolerated bugs, at least when their conscious inclusion in the code's design can help facilitate respect for some broader normative value.[115]

When framed in terms of (dis)affordances and inscriptions, the concept of desirable inefficiency can be applied fruitfully to end-user-facing code where that inefficiency operates to throttle computational immediacy in service of comprehension and empowerment. Kitchin and Fraser's notion of 'slow computing' puts the human *qua* human centre-stage, consciously reducing 'time compression, fragmentation, densification and stresses' in end-user interactions with code.[116] This notion can be seen in the work of the Slow Research Lab, which aims 'to evoke a quality of being, characterized by critical thinking, deep spaces of reflection, and the unique forms of creative expression that are born of them'.[117] Pols and Spahn connect this outlook with critical theories of technology that view it as a fundamental threat to democracy and justice.[118] From these perspectives, social spheres in which democratic values ought to be given time and space to operate are in danger of being limited by a 'technological rationality that centers on efficiency and strategic manipulation'.[119] Democracy and justice depend on 'communicative' rather than 'strategic' rationality, and thus on the provision of open spaces within which the former can be allowed to happen. The affordance of delay is thus about circumscribing 'technological rationality' (speed, efficiency, certainty) in favour of those spaces.[120] There is a thematic link here to the counterintuitive notion of intentionally fostering ambiguity in a design's affordances, so as not to constrain the end-user's responses to the artefact to only those

[114] Clark et al. (n 21) 467.

[115] Cf. Diver, 'Computational legalism and the affordance of delay in law' (n 103).

[116] R Kitchin and A Fraser, *Slow Computing: Why We Need Balanced Digital Lives* (Bristol University Press 2020).

[117] See Slow Research Lab, 'Slow Research Lab' <https://slowlab.net/about> last accessed 4 March 2021.

[118] Pols and Spahn (n 33) 342 *et seq*.

[119] Ibid. 345.

[120] See also Cohen's notion of the exploratory 'play of everyday practice', necessary for the autonomous individual to explore and exploit the space between predictability and contingency. See JE Cohen, *Configuring the Networked Self: Law, Code, and the Play of Everyday Practice* (Yale University Press 2012) chapter 2.

possibilities constituted by the designer.[121] This echoes the postphenomenological concept of multistability, discussed in Chapter 2,[122] and the crucial distinction I have already explored between constitutive and regulative normativity.

FRICTION

Discussions of code friction in the literature generally refer to its opposite, especially in the context of sharing on social networks. As with efficiency, the presumption is often that less is better,[123] when in truth a lack of friction often stands in opposition to the exercises of autonomy exemplified by deliberation and the weighing up of choices and consequences.[124] 'Frictionless sharing' refers to the ease and speed with which the design of social networks afford sharing,[125] for example through the use of metadata standards like Open Graph that provide attention-grabbing previews, or widgets that make sharing to social networks from third-party websites so easy as to enable 'viral' posting. This reduction in friction can be taken even further, to the point where everyday events like visiting a shop or going for a run are automatically shared by the code on social media platforms, without the end-user's input.[126] Before these affordances existed, the act of sharing online meant going through various manual steps: copying the URL of the item into an email or instant message, choosing the recipient(s), and composing a note to give the item context. All of this requires thought and conscious decision-making, in contrast to the single-click, one-to-many forms of sharing described above.

Designs that include especially efficient affordances can have unforeseen and undesirable consequences if they are not accompanied by appropriately

[121] WW Gaver, J Beaver and S Benford, 'Ambiguity as a resource for design' in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (ACM 2003) <http://doi.acm.org/10.1145/642611.642653> last accessed 4 March 2021. See also M Chalmers and I MacColl, 'Seamful and seamless design in ubiquitous computing' in *Proceedings of Workshop at the Crossroads: The Interaction of HCI and Systems Issues in UbiComp*, vol. 8 (2003).

[122] See also Gaver et al. (n 121) 236–7.

[123] McGeveran (n 99) 51; Ohm and Frankle (n 104) 10–13. Calo also argues in favour of a reduction in friction, which he pits in opposition to the 'facilitation' of the end-user's aims. See R Calo, 'Code, nudge, or notice' (2013) 99 *Iowa Law Review* 773.

[124] Narayanan et al. go so far as to say frictionlessness 'robs' end-users of such opportunities, leading them to follow their 'baser impulses'. See A Narayanan et al., 'Dark patterns: Past, present, and future' (2020) 18 *ACM Queue* 25, 82.

[125] McGeveran (n 99). See also C Reed and A Murray, *Rethinking the Jurisprudence of Cyberspace* (Edward Elgar Publishing 2018) 120.

[126] T Bucher, 'A technicity of attention: How software "makes sense"' (2012) 13 *Culture Machine*.

informative signifiers. A problematic aspect of Facebook's frictionless sharing has been some end-users' misunderstanding of who precisely they are sharing intimate posts with.[127] In that respect, McGeveran connects the idea of friction to design: 'the amount of friction is a complex design choice, which inherently helps some users and burdens others. We cannot avoid making some choice, whether through code or law; there is no "natural" state of online friction.'[128] McGeveran suggests a design principle according to which the ability to share should not be made available before the act itself has taken place. This 'law of friction' states that 'it should not be easier to "share" an action online than to do it'.[129] A similar principle might apply to any computational step that will have normative effect: the end-user should be afforded the opportunity to consider it first. The idea here is to consciously design friction into the appropriate parts of the artefact's inscriptions, so that end-users are given an opportunity to take stock before the code moves on to the next step in its logic.[130] The interface of an artefact is like a keyhole; the breadth and depth of the mass of code steps that are in fact being executed are like the vast bulk of an iceberg hidden beneath surface waters.[131] Whereas text as a normative medium is shallow (even with interpretative flexibility taken into account), code has depth that is simultaneously difficult to observe and difficult to comprehend, on account of its inherent complexity.[132] The challenge then is to design interfaces that afford the appropriate pacing of computation, alongside an appropriate level of technical feedback, in order

[127] McGeveran calls this a 'misclosure'. See McGeveran (n 99) 39 *et seq.*, citing KE Caine, 'Supporting privacy by preventing misclosure' in *CHI'09 Extended Abstracts on Human Factors in Computing Systems* (ACM 2009). See also S Sengupta, 'Private posts on Facebook revealed' *The New York Times* (28 January 2013) <https://bits.blogs.nytimes.com/2013/01/18/private-posts-on-facebook-revealed/> last accessed 4 March 2021.

[128] McGeveran (n 99) 53–4.

[129] Ibid. 63. Recent changes to Twitter's interface implement McGeveran's 'law' by asking users whether they would like to read an article before retweeting, the idea being to give less exposure to sensationalist headlines. See C Jee, 'Twitter wants you to read articles before you retweet them' *MIT Technology Review* (11 June 2020) <https://www.technologyreview.com/2020/06/11/1003333/twitter-wants-you-to-read-articles-before-you-retweet-them/> last accessed 4 March 2021.

[130] Ohm and Frankle (n 104) 51–2; McGeveran (n 99).

[131] C Vismann and M Krajewski, 'Computer juridisms' (2007) *Grey Room* 90, 100; MC Marino, 'Critical code studies' (2006) *electronic book review* <https://electronicbookreview.com/essay/critical-code-studies/> last accessed 15 April 2021 13–14.

[132] NK Hayles, 'Print is flat, code is deep: The importance of media-specific analysis' (2004) 25 *Poetics Today* 67.

to facilitate a sufficiently detailed mental model for the end-user that she can make reasonable predictions of what is going to happen next.[133]

HUMAN IN THE LOOP

A primary mechanism for forcing delay into code-mediated processes is the 'human in the loop' ('HitL') principle. A distinction is drawn between those elements of the technical process that can appropriately be executed mindlessly by the machine, and those that have social, ethical, or legal import that must therefore be made by a human, either independently or to ratify the machine's output.[134] The classic application of the HitL principle is in lethal autonomous weapon systems, where a military engagement is automated up to the final decision on whether to strike, which is taken by a human controller.[135] In the policing context, Hartzog et al. suggest a 'conservation principle', requiring inefficiency and indeterminacy to be conserved through retention of human judgement at specific points within the criminal justice process.[136] For them, HitL is a necessary bulwark against the determinism of inflexible code, and they suggest that where one of either surveillance, analysis, or crime detection are automated in code, the (desirable) inefficiency and indeterminacy of the other two should be increased proportionately.[137]

In the context of consumer code, the human in the loop is the end-user herself. In order to facilitate desirable delays in that context, then, interfaces must afford end-users notification and choice at appropriate moments before execution takes place. As discussed above, information about these scenarios should not be front-loaded in terms and conditions documents that are not read, but rather should be delivered piecemeal at appropriate moments in the end-user's journey through the code's inscriptions. This can be achieved by, for example, employing 'just in time' notifications, akin to those provided in

[133] Hartzog (n 25) 278. For a related argument for making end-user experiences less simple in certain circumstances, see K Roose, 'Is tech too easy to use?' *The New York Times* (12 December 2018) <https://www.nytimes.com/2018/12/12/technology/tech-friction-frictionless.html> last accessed 4 March 2021.

[134] One must, however, be sensitive to the potential reflexivity of this arrangement. For analysis of this phenomenon (among others) in the 'legal tech' context, see the work of the ERC Advanced Grant research project 'Counting as a Human Being in the Era of Computational Law' (COHUBICOL) <https://www.cohubicol.com> last accessed 4 March 2021, of which I am a member.

[135] N Sharkey, 'Grounds for discrimination: Autonomous robot weapons' (2008) 11 *RUSI Defence Systems* 86.

[136] W Hartzog et al., 'Inefficiently automated law enforcement' (2015) *Michigan State Law Review* 1763.

[137] Ibid. 1778.

the Android operating system for the ad hoc granting or denying of permissions to applications at the moment they request them, instead of in bulk at the moment of installation when the end-user might not foresee the relevant implications.[138] The goal is to granularise permissions and make them contextually relevant, empowering the end-user at the point she can make an informed decision.

HitL is also a necessary element of retaining indeterminacy, the quality of a circumstance not being adequately reflected in the code (or data) that comes to represent it (recall the discussion in Chapter 3 on code's limited ontology being reductive of the world).[139] Whereas code can impose such interpretations, 'underdeterminacy'[140] should be preserved, to allow for responses that are sensitive to the contingent and irreducible texture of the real world. The human who is in the loop has a role in 'completing the narrative' in such scenarios, filling in the contextual gaps which computational representations are incapable of showing sensitivity to but which are nevertheless central to the pursuit of justice or of end-user autonomy.[141] The goal, then, is to ensure that the design affords HitL input at all appropriate points in its inscription, so that the aspirations of freedom and autonomy are not effaced by the 'duty' of wired-in code.

Blockchain Applications

Many of the considerations of ruleishness discussed above also apply to immediacy. Levy notes that '[b]ecause they are based on code', blockchain applications 'can be *immediately and automatically* effectuated, without reliance on manual transfer, or the intervention of institutions like courts.'[142] One of the putative benefits of blockchain applications (promoted in particular by smart contract enthusiasts[143]) is their removal of the perceived inefficiency

[138] 'Android developers guide – permissions overview' <https://developer.android.com/guide/topics/permissions/overview> last accessed 4 March 2021. For a recent discussion of *kiaros*, or the notion of 'right-time' in algorithmically mediated systems, see T Bucher, 'The right-time web: Theorizing the kairologic of algorithmic media' (2020) 22 *New Media & Society* 1699.

[139] F Pasquale, 'A rule of persons, not machines: The limits of legal automation' (2019) 87 *George Washington Law Review* 1, 49 *et seq*.

[140] M Hildebrandt, 'Legal and technological normativity: More (and less) than twin sisters' (2008) 12 *Techné: Research in Philosophy and Technology* 169, 177.

[141] Hartzog et al. (n 136) 1785 *et seq*.

[142] Levy (n 61) 2 (emphasis supplied).

[143] Pasquale (n 139) *passim*. For an example, see Mattereum, 'Mattereum Protocol: Turning code into law' (Mattereum Project 2018) <https://www.mattereum.com/upload/iblock/784/mattereum-summary_white_paper.pdf> last accessed 4 March 2021.

of ambiguity and processual costs.[144] This is potentially deeply problematic, especially if the code has been poorly designed. When combined with the immutability of a blockchain, the consequences can be serious indeed. As De Filippi and Wright suggest, '[t]he automated nature of smart contracts, combined with the inability to readily alter their underlying code, could further lead to situations where a faultly [sic] piece of code would repeatedly run, to the detriment of all parties involved.'[145]

There is thus a need for ex ante consideration of the implications of automated and immediate execution: assets or funds could be transferred, goods and services ordered, or a person's legal status altered, all according to the predetermined logic of the blockchain application, without any human intervention or oversight. This will happen near-instantaneously if the conditions in the code are met. As with the affordance of choice, providing delay therefore requires the identification of appropriate moments in which the end-user must be afforded the opportunity to consider the situation before execution of the code continues.[146] Given the impossibility of anticipating every outcome of execution,[147] contingency ought not to be the province of the code, and any attempt so to 'enclose' it is perhaps likely to set up unforeseen and undesirable results. Simultaneously, however, imposing friction in blockchain applications is arguably anathema to their very ethos. This may be necessary for them to be deemed legitimate, however, given their potential exemplification of computational legalism.

The Internet of Things

In the discussion of default choices above I mentioned IoT webcams that have problematic 'out-of-the-box' configurations, such as insecure default administrative passwords.[148] We have seen how end-users often trust designers to know better than they do, and so assume that the default configuration must be the most sensible one. Such configurations are especially problematic in the IoT, because the object itself might be 'plug and play', meaning it starts

[144] Levy (n 61) 2.

[145] P De Filippi and A Wright, *Blockchain and the Law: The Rule of Code* (Harvard University Press 2018) 201.

[146] For a wider developmental and environmental angle on 'considering the situation', which takes into account much more than the end-user's immediate interests, see the combined blockchain/IoT project BitBarista: L Pschetz et al., 'Bitbarista: Exploring perceptions of data transactions in the Internet of Things' in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (ACM 2017).

[147] Cf. CD Clack, VA Bakshi and L Braine, 'Smart contract templates: Foundations, design landscape and research directions' (2017) arXiv:1608.00771 [cs], 4.

[148] Krebs (n 38).

operating according to its default configuration as soon as it is switched on. This single action may be enough by itself to set off various undesirable path dependencies, for example joining an open wireless network and connecting to a remote server to register its existence. Designing in delay in this context, then, could involve ensuring that IoT devices have all defaults set initially to prevent any functionality that is not immediately and obviously signified by the physical characteristics of the device. This relates back to the 'would not have wanted' standard, discussed above.

In the example of the smart fridge, switching it on for the first time would mean it immediately starts to refrigerate because that is its inherent purpose, but all 'smart' (networked) functionalities would remain disabled until the end-user takes the active step of configuring and enabling them. Building in a delay before the normative code executes can open up space for the other affordances to be facilitated: the end-user can consider the implications of the device's provenance and purpose, giving her a chance to respond to any misgivings she may have before potentially opaque harm is done. If she decides to go ahead, she can then think about which configurable choices best fit her interests. Like the suggestion below regarding a 'floor' of security in the IoT, we can imagine a baseline delay where no normative functionality that is not clearly signified by the physical properties of the artefact can be enabled prior to the end-user taking the active choice to do so, even (and particularly) where this is in opposition to the commercial interests of the manufacturer. In addition to a 'floor' of security, one can therefore imagine an initial 'ceiling' of affordance, extendable only by the conscious choice of the end-user.

(d) Immutability

The problems of code's immutability overlap with those demonstrated by its ruleishness and immediacy. Fuller's principle regarding frequency of change applies, but in the opposite sense: the fact that certain media are resistant to being changed must be borne in mind at design time;[149] the threshold between 'duty' and 'aspiration' must be set in the knowledge that path dependencies might arise that lock end-users into the constraints of a particular design. This relates to the legisprudential principle of temporality, which, as previously mentioned, requires sensitivity to the concreteness of the imposed rule particularly where there is less scope for future alteration, in which case the need for justification is all the stronger. Given the ways in which immutable code crystallises a particular configuration of technological normativity, there is a need both to justify that configuration and to balance it with the affordance

[149] Hartzog (n 25) 76–7.

of oversight. There is a connection here too with the principle of coherence, which under its third level requires a broader societal justification for the rule and not just coherence in accordance with the system's internal rationale. From that perspective, the digisprudential strategy of oversight means that a change in the external justification must be capable of being reflected via alteration of the code; a failure to afford this would mean the continued operation of illegitimate code, regardless of its legitimacy at original time of release.

Digisprudential Affordance: Oversight

The manufacturer ought not to release code unless the necessary conditions are in place for them to maintain oversight of it and to correct any (unforeseen) negative consequences. This is similar to the concept of *revocability* in the HCI-Security literature, where the end-user must be afforded the possibility of revoking any permissions she has granted within the system.[150] In this context, the concept of revocability requires that the creator of the code be capable of maintaining some control over it.[151] In this light, maintaining legitimacy requires that the design anticipate ex ante the potential need to make changes ex post. Respect for this principle requires that the design itself permit it; any design that does not is *prima facie* illegitimate.[152]

Consider again the Sony BMG DRM scandal discussed in Chapter 1. The problematic effects of its design were amplified by its storage on an inherently immutable medium, the compact disc. Although the system was ultimately revoked, this was only as a result of the significant public relations impact of the scandal, and that revocation took the form of a laborious, expensive, and wasteful physical recall of more than seven million CDs. Similar issues arise

[150] K-P Yee, 'User interaction design for secure systems' in R Deng et al. (eds), *Information and Communications Security* (Springer 2002).

[151] As Winner puts it, 'men release powerful changes into the world with cavalier disregard for consequences', discussing the central theme of Shelley's *Frankenstein; or, the Modern Prometheus.* See L Winner, *Autonomous Technology: Technics-Out-of-Control as a Theme in Political Thought* (2nd edn, MIT Press 1977) 314. For an annotated edition of the latter that illuminates the novel's themes from an engineering perspective, see M Shelley, *Frankenstein: A New Edition for Scientists and Engineers*, ed. E Finn, D Guston and JS Robert (MIT Press 2017).

[152] This goal is echoed in recent EU developments mandating that product designs afford repairability to reduce waste and extend the lifespan of consumer electronics. See for example M Anastasio, 'EU governments support first set of laws for more repairable products' (EEB – The European Environmental Bureau, 13 December 2018) <https://eeb.org/eu-governments-support-first-set-of-laws-for-more-repairable-products/> last accessed 4 March 2021.

in the IoT, where slim margins on inexpensive devices mean the incentive to invest in long-term updates and support is diminished. Code is thus rushed to market with neither the capacity for ex post software updates nor the ongoing commitment to provide bug and security fixes.[153]

From a legitimacy perspective, the design must afford oversight by the designer or enterprise so that any necessary changes to the code can be made. This will involve anticipation of software updates, now a fairly standard feature in modern networked devices.[154] As with elsewhere within the digisprudential framework, the implication is that if the enterprise cannot commit to such standards of oversight then the legitimacy of the design has *de facto* not been demonstrated and its technological normativity is not justified. Similarly, where the design does not permit updates by its very nature (for example due to limited connectivity or processing power) then the scope of wired-in functionality should be to that extent limited to ensure that the unchangeable code will not cause future negative effects. The design must therefore anticipate external change, either by the facilitation of remote updates or by restricting the scope of its normativity from the outset. Where it proves too difficult to anticipate these eventualities, ex post remedial measures of the sort envisaged by Hartzog and Selinger (for example third-party maintenance or insurance) must be put in place. If none of this is possible, the inevitable conclusion is that the design is *a priori* illegitimate.

SUNSETTING AND 'LOBOTOMY SWITCHES'

Discussing the Sony BMG DRM scandal, Halderman and Felten suggest the inclusion of 'sunsetting', a mechanism that renders the system inert after a specified period or date.[155] Depending on the business model being pursued, this might avoid some of the problems of code executing indefinitely, particularly if it is especially difficult to alter it ex post (as with physical CD media). The Sony BMG system could have been designed to execute for only as long as there was commercial benefit in enforcing copyright by means of code. Halderman and Felten suggest a period of three years, during which most of the revenue from disc sales would have been raised. Whether an approach

[153] W Hartzog and E Selinger, 'The Internet of Heirlooms and Disposable Things' (2016) 17 *North Carolina Journal of Law & Technology* 581.

[154] In the security context, Hartzog and Selinger suggest a 'minimum expectation of servicing' standard, and a 'floor of data security even for disposable items' (ibid. 597). See also ENISA's recent report on IoT supply chains: 'Guidelines for Securing the Internet of Things' (European Union Agency for Cybersecurity (ENISA) 2020).

[155] JA Halderman and EW Felten, 'Lessons from the Sony CD DRM episode' in *15th USENIX Security Symposium* (USENIX Association 2006) 89.

of this kind is effective depends on the business model – the economics of CD sales have of course changed significantly since the Sony BMG scandal. Nevertheless, the principle is valid: designers ought to consider the medium- and long-term effects of the technological normativity they embody in their systems, and where oversight over such a period is anticipated to be difficult or impossible, sunsetting is a mechanism that can limit the possible effects of the code operating blindly in unforeseen contexts.

Related to sunsetting is what Hartzog calls a 'lobotomy switch', which reduces the system to a set of core functions while disabling any optional affordances (particularly network access).[156] This is the mirror image to the discussion of core affordances above in the section on delay: basic functionality is retained, but optional 'smartness' is disabled. Hartzog gives the example of a child's Internet-enabled doll: once the lobotomy switch is flipped, the doll can still be played with, but its potentially security- and privacy-harming connectivity is disabled. Again, the efficacy of this approach depends on the type of device; if networking is a central aspect of its utility (as for example with the Amazon Dash Button) then disabling it through a lobotomy switch might render the device essentially useless. It is also complicated by questions of who should control the switch, and under what conditions it might be activated. These questions represent a point at which the institutional law might reprise its traditional regulative role.[157]

In any event, the overarching question of legitimacy operates, raising the thorny question of whether the device should have been designed in such a way in the first place. If the manufacturer cannot commit to (1) supporting the device with updates and maintenance for a reasonable period, (2) 'sunsetting' the device (or the relevant parts of its functionality) after a specified period, or (3) retaining sufficient control to permit a 'lobotomy' to be performed should this turn out to be necessary, then the design is not legitimate, because it does not afford the necessary level or quality of oversight.

Blockchain Applications

One of the notional selling points of blockchains is that data stored on them is tamper-resistant.[158] From the perspective of orthodox contract law this is problematic, since the ex ante interpretation formalised in the code of the

[156] Hartzog (n 25) 272. Even the simplest networked devices can be zombified as part of a bot net, used for example in distributed deniable of service (DDOS) attacks.

[157] Ibid. 273; B Schneier, 'I've seen the future, and it has a kill switch' *Wired* (26 June 2008) <https://www.wired.com/2008/06/securitymatters-0626/> last accessed 4 March 2021.

[158] De Filippi and Wright (n 145) 35–7.

blockchain application is what will be executed when the relevant conditions arise, regardless of any intervening factors which might otherwise have invited more flexibility.[159] The technical necessity for consensus to be reached in order to make changes, coupled with the inability unilaterally to breach the 'contract', makes these artefacts especially problematic in terms of oversight. Observing the fact of the application's execution may be possible because the output of a blockchain application's execution is generally stored on the underlying chain. This is a different kind of oversight, however. What matters from a digisprudential perspective is ongoing maintainability and revocability, to ensure that the code's normativity can be accounted for; both are undermined by the immutability of a blockchain. If one 'party' to the application's 'contract' changes her mind, or is incapacitated, or the codified norms are otherwise illegal, the code will in principle remain on the blockchain and will execute as stored, regardless of such contingencies.

This goes to the very heart of the kind of ex ante anticipation that digisprudence is concerned with. Designers of blockchain applications must be aware of contingencies well in advance. Because of the immediacy of the code, they must therefore limit the normative scope of the latter to those facts that they can be reasonably certain of. Even if the context of the code's operation entails emergence or complexity, this will not prevent the application from operating unless its code is designed to include some external check of such contingencies.[160] The question must then be faced of whether it is feasible to predetermine all the relevant contingencies that might arise, and whether even those that are foreseen are supported by reliable third-party sources of information. Oracles – third-party sources of contingent data used in the application[161] – might not provide the necessary information at the moment it is needed, or that information might be inaccurate, incomplete, or not in a format the code is equipped to 'understand'. Even where these issues do not arise initially, there is an inherent assumption that the oracle will continue to operate as it did at the time the application was designed, but this may not be the case if the third party alters the oracle's code, or shuts down altogether. Furthermore, in terms of contestability, even where a judicial process might in theory be invoked to attempt to address any conflict that arises, it may be difficult to identify the parties from the application's code in order to demonstrate legal standing to contest or seek decree, because identification

[159] K O'Hara, 'Smart contracts – dumb idea' (2017) 21 *IEEE Internet Computing* 97, 98.

[160] RH Weber, '"Rose is a rose is a rose is a rose" – what about code and law?' (2018) 34 *Computer Law & Security Review* 701, 5.

[161] Cardozo Blockchain Project, '"Smart contracts" & legal enforceability' (Benjamin N Cardozo School of Law 2018) 8.

is by means of anonymous public keys rather than names. In any event, even were it possible, such an appeal to judicial process would take place after the code has executed and its negative effects have been felt.

Two factors might ultimately militate against blockchain applications in terms of the affordance of oversight. First, if the designer cannot be sure whether certain crucial facts will obtain at point of execution, she must limit the 'wired-in' elements of the code to exclude these. Difficulties arise in identifying the threshold between what Clack et al. call the 'operational aspects' of the blockchain application, namely those that are automatable, and those that are 'non-operational' and cannot or should not be automated.[162] Too much automation and many or all of the effects of computational legalism are amplified; too little and what remains automated in the blockchain application's logic may be so simplified as to obviate its 'smartness'. This might be a useful limitation, however, rendering the code a 'mechanism'[163] for the execution of a real-world agreement between humans, the latter retaining responsibility for managing any ambiguity.[164] Thus, the social level of agreement (including institutional legal contracting) continues to be the locus of the contingent parts of 'real-world' human arrangements, while the role of the blockchain application is constrained to those limited factors susceptible to reliable and predictable code-based representation and enforcement.[165] This is, in a sense, to flip the 'lobotomy switch' ex ante, limiting the design of the application from the beginning, in the knowledge that it might otherwise harbour too much normative power. Whether or not such a notionally legitimated blockchain application would retain any commercial attractiveness remains to be seen, but exposing business models that rely on illegitimate code would be a price worth paying.

The second factor militating against the use of blockchain applications relates to the code's ability to respond to contingent facts. If the designer is unwilling to forego the 'smartness' of the application in the manner just described, the external contingent facts that it relies upon must be verifiable at the point of execution. This implies the use of oracles that are themselves trustworthy and accurate. This will be problematic from an oversight

[162] Clack et al. (n 147) 5.

[163] Felten suggests this term as an alternative to the confusing 'smart contract'. See E Felten, 'Smart contracts: Neither smart nor contracts?' *Freedom to Tinker* (20 February 2017) <https://freedom-to-tinker.com/2017/02/20/smart-contracts-neither-smart-not-contracts/> last accessed 4 March 2021.

[164] De Filippi and Wright (n 145) 199–200; Cardozo Blockchain Project (n 161) 4.

[165] Levy (n 61).

perspective because it devolves the determination of a crucial element of the artefact's logic away from the manufacturer, thus undermining its ability to oversee the operation of its own design. One response to this is to design in a kind of 'meta-contingency', somewhat akin to sunsetting, where the blockchain application will simply lie inert if, at point of execution, it cannot confirm a given fact to the requisite degree of certainty. Of course, any such safety valve must be consciously designed into the logic of the code. Whether the precise set of facts that would come within this bracket can be identified by a designer in advance (rather than by a court ex post, with all the benefits of expert evidence and time to deliberate), and whether they can be provided by an oracle in a form that is susceptible to computational representation, are questions that are themselves contingent on many external conditions being in place (for example a facility providing information that the relevant end-users are still alive and *capax*, or that the property or goods that the application purports to transact with still exist and are in the possession of the relevant party who retains a right of disposal). The complexity and variety of factors that ought to be taken into consideration might mean these standards of oversight cannot be met, which again will call into question the legitimacy of such applications *a priori*.[166]

The Internet of Things

We have already seen in the discussion above some suggestions relating to IoT devices specifically. Because they tend to be low-cost, there are numerous examples where manufacturers have under-invested in the ongoing maintenance of their devices.[167] The resources required to track and fix bugs and to provide infrastructure for delivering updates to the devices can mean that in a febrile market resources are directed instead towards developing new products. Some manufacturers have even resorted to altering legal terms in an attempt to contract out of responsibility for the technological normativity of their designs. For example, after a serious breach of personal data toy manufacturer Vtech simply changed their terms document to shift responsibility onto the end-user, instead of altering the design of their product.[168] Whatever

[166] Pasquale (n 139) 24 *et seq*.

[167] L Edwards, D McAuley and L Diver, 'From privacy impact assessment to social impact assessment' in *2016 IEEE Security and Privacy Workshops (SPW)* (IEEE 2016).

[168] 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> last accessed 4 March 2021.

the legal (de)merits of this approach, it does nothing to make legitimate the technological normativity of the device.

For IoT devices, then, oversight must be designed into the system itself, including the ability to update its code should this be required in future (which implies a commitment to support such updates). As Hartzog and Selinger suggest,

> [i]magine a system where companies told users how long they think a wired object will last and how long the company will commit to providing security patches. In the event that a company goes bankrupt before then, companies would work quickly to either notify users of its impending shut down or facilitate the [transfer of] responsibility for security patches to a third party.[169]

This might be combined with sunsetting facilities that either warn the end-user that the device has an expected operating life of a specified period or, if the supporting infrastructure becomes unavailable (for example due to insolvency of the manufacturer), that either there will be a third-party support mechanism, or the system will gracefully degrade (sunsetting/the lobotomy switch) or be disabled altogether. What such measures might mean for consumer protection or contract law remains to be seen; of course, as with all the other digisprudential affordances, if the manufacturer of the device cannot commit to producing a design that embodies a sufficient level of foundational legitimacy then the conclusion must always remain open that the design is *a priori* illegitimate and it should not be released.

(e) Pervasiveness

The pervasiveness of code connects with the idea of 'juridification' and the legalistic proliferation of 'ever more refined and rigid systems of formal definitions'.[170] This is an implied problem that the legisprudential principle of normative density aims to reduce, by increasing the level of justification required in proportion to the limitation of freedom (criminal sanction being the 'densest' example). The concept of juridification takes this wider to consider not just the 'density' of a given norm's limitation on freedom, but the aggregate impact on freedom of the proliferation of legal normativity more generally.[171]

In the legal sphere, the effects of juridification are limited by both institutional resources and human cognitive capability. Beyond a certain threshold,

[169] Hartzog and Selinger (n 153) 597.

[170] JN Shklar, *Legalism* (Harvard University Press 1964) 2.

[171] LC Blichner and A Molander, 'What is juridification?' (Centre for European Studies, University of Oslo 2005) 12 *et seq*.

citizens cannot comprehend the body of norms they are subject to, and there are limited resources for enforcing every applicable norm. There is thus a natural limit to pervasiveness within the legal domain. In the computational realm, however, such limits do not exist (or their threshold is much higher); the number of norms or the aggregate normativity that can be created and enforced by and through code is potentially unlimited, at least within the domain of the artefact. Pervasiveness under computational legalism thus exemplifies these dual aspects of juridification: the density of the individual norms, and complex aggregations of normativity embodied in the inscriptions of both individual and networked collections of devices. We have already seen how technological normativity can have an immediate regulating effect in a way that law cannot (and ought not[172]); whereas traditional legal norms can be directed at whole populations (or even large classes of individual), their text-bound character dramatically limits the real-time imposition of their normative effect. The ways that code differs in this respect, discussed earlier, are made all the stronger when the artefact has widespread adoption – large numbers of individuals can be subject simultaneously to the normative effect of even a single design decision.[173] We can therefore adapt the legisprudential principle of normative density to take account of this collective dimension of code normativity, in terms of both the effects of multiple artefacts and the effects on multiple end-users. When combined with the other digisprudential affordances, the question of aggregate technological normativity becomes extremely salient; pervasiveness takes the other qualitative aspects of computational legalism and adds a quantitative element into the legitimacy equation.

6.4 Conclusion

This chapter has strengthened the relationship between legal-theoretical notions of legitimacy and the practical question of what legitimate code ought to afford (1) the end-user (contestability, choice, transparency, and delay), (2) legal institutions (evidential standards), and (3) code's own creators (oversight). The framework of digisprudential affordances is set out, providing a basis for guiding the design of code towards legitimacy.

[172] Here the distinction between 'legal protection by design' and 'legal by design' is crucially important. See M Hildebrandt, *Law for Computer Scientists and Other Folk* (Oxford University Press 2020) 302 *et seq*.

[173] A Huldtgren, 'Design for values in ICT' in J van den Hoven, PE Vermaas and I van de Poel (eds), *Handbook of Ethics, Values, and Technological Design* (Springer 2014) 741.

By discussing their potential application to concrete technologies, I have also tried to begin bridging the divide between legal-theoretical notions of normative legitimacy and their practical, real-world instantiations. The next chapter pushes further in this practical direction, focusing on the code development cycle and how the 'programmer of the programmer' can be employed as a 'constitutional' guide, encouraging the production of legitimate code.