THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Opening the black box

Petri nets and privacy by design

Citation for published version:
Diver, L & Schafer, B 2017, 'Opening the black box: Petri nets and privacy by design', *International Review of Law, Computers and Technology*, vol. 31, no. 1, pp. 68-90.
https://doi.org/10.1080/13600869.2017.1275123

Digital Object Identifier (DOI):
10.1080/13600869.2017.1275123

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
International Review of Law, Computers and Technology

Publisher Rights Statement:
This is an Accepted Manuscript of an article published by Taylor & Francis in International Review of Law, Computers & Technology on 22/02/2017, available online:
http://www.tandfonline.com/doi/full/10.1080/13600869.2017.1275123

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 12. Oct. 2024

Opening the Black Box: Petri nets and Privacy by Design

Laurence E. Diver and Burkhard Schafer

School of Law, University of Edinburgh, Edinburgh, Scotland

laurence.diver@ed.ac.uk (corresponding) and b.schafer@ed.ac.uk

Opening the Black Box: Petri nets and Privacy by Design

Building on the growing literature in algorithmic accountability, this paper investigates the use of a process visualisation technique known as the Petri net to achieve the aims of Privacy by Design. The strength of the approach is that it can help to bridge the knowledge gap that often exists between those in the legal and technical domains. Intuitive visual representations of the status of a system and the flow of information within and between legal and system models mean developers can embody the aims of the legislation from the very beginning of the software design process, while lawyers can gain an understanding of the inner workings of the software without needing to understand code. The approach can also facilitate automated formal verification of the models' interactions, paving the way for machine-assisted privacy by design and, potentially, more general 'compliance by design'. Opening up the 'black box' in this way could be a step towards achieving better algorithmic accountability.

Keywords: privacy by design; regulatory theory; Petri nets; formalisation; transparency, algorithmic accountability

This work was supported by the RCUK-funded CREATe Centre under grant AH/K000179/1.

Opening the Black Box: Petri nets and Privacy by Design

Laurence E. Diver and Burkhard Schafer[1]

1. Introduction

Questions of data privacy have never been more salient. More and more devices are being built with Internet connectivity, and it is expected that by 2020 the number of online devices will reach 50 billion, a two-fold increase from 2015 (Cisco Systems 2011, 3). The combination of increased volume and speed of data transmission and processing, together with the often irrevocable consequences of security or privacy breaches, renders insufficient traditional forms of regulation that focus on post-violation punishment or complex injunctive relief.

The need for data protection to be 'built in' to the digital systems which gather and process personal data is thus becoming ever more pressing. European institutions[2] have expressed a desire for market-driven approaches to 'privacy by design'[3] which can respond to the changing technological and economic landscape whilst maintaining a base level of protection for European citizens' fundamental rights (EDPS 2015, 8).[4]

In the midst of these legal and technological changes, a burgeoning literature has developed on the accountability of the algorithms and code that constitute and govern the behaviour of such systems and, by extension, that of their users. This new

---

1 CREATe Research Fellow, and Professor of Computational Legal Theory, respectively. The authors would like to thank the anonymous reviewers of an earlier draft for their helpful comments, as well as delegates who attended the Google Workshop at BILETA 2016, where an earlier version of this paper was presented. This research was supported by the RCUK-funded CREATe Centre under grant AH/K000179/1.

2 At the original time of writing, the United Kingdom referendum on European Union membership was pending. The eventual result fell, of course, in favour of leaving the EU. Despite the continuing uncertainty around the nature and extent of 'Brexit', the General Data Protection Regulation will nevertheless have extra-territorial effect (see Recitals 101, 104, 108 and Arts. 44-50). At any rate, and as discussed further below, the techniques presented here are limited neither to the field of data protection nor to any particular jurisdiction.

3 Although the General Data Protection Regulation refers to 'data protection by design', this paper treats the terms as synonymous.

4 On the question of European citizenship and the UK's membership of the EU, see *supra* note 2.

scholarship considers the impacts of opacity in the behaviour and decision-making of these systems, and how it affects societal values such as democracy, autonomy and, of course, privacy (Danaher 2016; Kitchin 2016; Pasquale 2015; Grimmelmann 2005).

1.1 Transparency vs by design

In seeking to hold to account the behaviour of digital systems, some authors have focused on the need to *observe* their workings directly, and have suggested that transparency is therefore the key aim for accountability. This could be achieved by, for example, mandating the use of open source software in public governance systems, as advocated by Citron (2008). While this is an important potential element in achieving transparency, disclosing source code alone is unlikely in practice to achieve the desired degree of visibility. The gap in expertise between legal/policy experts and those who write the code remains too significant, and the risk is that the former will continue to be at the mercy of the interpretations, biases, and programming (in)abilities of the latter.

Many modern technical systems are of such complexity that, even were it possible to review the entire source code, it is not possible for a human to compute (in the original sense) all of the data flows and computational events that will arise in practice when the code is executed. Furthermore, the difficulty, time and cost of this type of analysis means the danger arises that the notionally transparent system will be interrogated and tested only *after* a problem has occurred. As argued above, the speed of data transmission and processing makes such *post factum* interventions more and more inefficient: once sensitive personal data have been disclosed, repairing the damage through legal procedures is all but impossible.

An alternative species of approach is concerned with the idea of 'designing in' compliance from the outset, such that the substantive target of accountability is not so much the (bad) behaviour of an inscrutable system operating 'in the wild', much of

which might never be detected or even observed, but rather the design decisions that were taken that led to a technical architecture which prevented such bad behaviour from being possible in the first place.

This paper is intended as a practical contribution to this latter approach, itself a subcategory of algorithm studies more generally. It proposes the use of a technique known as the Petri net to model visually both legal and software processes, such that conflicts between them can be identified early on in the process of designing a new digital system. The approach has the potential to facilitate efficient and effective regulatory compliance and accountability at both the technical and organisational levels.[5]

Privacy by Design as a regulatory approach bridges two realms that, since the time of David Hume, had been considered categorically different: the *is*, and the *ought*. In this approach, what the system *can* or *cannot* do corresponds closely to what it *ought* or *ought not* do. Crossing the boundaries between conceptual spaces like this brings a number of challenges, both abstract-theoretical and practical. It requires both close collaboration between people of very different backgrounds and expertise, and a 'bridging vocabulary' that will allow them to communicate efficiently across the boundaries of their disciplines.

The concept of 'proof' is one such bridging concept: it is central to both law and computer science. Lawyers and computer scientists deploy very different notions of what 'proof' can mean, however. From a legal perspective justice must not only *be* done, it must also be *seen to be* done; this latter requirement puts constraints on what can and cannot count as proof in that domain. From a computer science perspective,

---

[5] The General Data Protection Regulation makes numerous references to the use of both 'technical and organisational measures' as means of implementation. See for example Recitals 78 and 156, and Arts. 24(1) and 25.

proven assurances of the correctness of algorithms take the form of highly-constrained formal representations using, for example, Hoare logic (London 1972).

As discussed in more detail below, the law requires not only that digital systems are privacy compliant, but that they are provably so, meaning that this fact must be rigorously demonstrable. Formal proofs can play an important role in this, but only if we can translate this notion into the language of *legal* proof, namely that justice must not only *be* done but also be *seen to be* done. The visual language of the law, the 'seen to be done', finds its correlation in formal systems that use visualisation techniques. This is the theoretical justification for the use of Petri nets here described: they act as a neutral bridging language that allows us to translate claims about the correctness of the system into claims about its legal compliance, in a way that is visible and intelligible to lawyers and other non-computer scientists.

There is a significant literature that analyses the benefits of visualisation for legal reasoning and legal comprehension, particularly in an educational setting (Voyatzis and Schäfer 2012, 149 and refs.). Since privacy by design is a paradigmatic example of a context in which lawyers/policy experts and systems designers must learn from and work with one another, the use of an appropriate visualisation technique as a translation tool seems *prima facie* appropriate. The specific tool demonstrated here bears significant resemblance to other techniques already widely used to visualize legal argumentation, adding further support to the claim made in this paper that it is an ideal candidate to connect legal and computer-scientific modes of thought (Verheij 2007).

Before setting out the foundations of the novel approach, the paper considers the main privacy by design provisions in the forthcoming General Data Protection

Regulation (hereinafter 'the GDPR'),[6] as well as some general conceptual issues arising from the idea of privacy by design. Thereafter it introduces and demonstrates the technique, assesses its suitability for implementation within the new regime of the GDPR, and suggests some areas for future research.

2. Privacy by Design

2.1 Origins and urgency

Privacy by design (hereinafter 'PbD') originates in work done in the mid-1990s by the office of the Ontario Information and Privacy Commissioner (Cavoukian, 2012). The first mention in European institutional literature is in a 2010 Commission Communication which states tersely, in a footnote, that

> [PbD] means that privacy and data protection are embedded throughout the entire life cycle of technologies, from the early design stage to their deployment, use and ultimate disposal. (European Commission 2010a, note 21)

This definition was repeated verbatim, again only in a footnote, in another 2010 Commission Communication (European Commission 2010b, note 30). Two years later, the concept of PbD was more fully articulated in Article 23 of the draft GDPR, where it was referred to as 'data protection by design'.[7]

Having gone through significant amendment in the European Parliament and Commission readings, the text of Article 25(1) now states:

> Taking into account the state of the art, the cost of implementation and the nature, scope, context and purposes of processing as well as the risks of varying likelihood

---

[6] Regulation on the protection of individuals with regard to the processing of personal data and on the free movement of such data. References are to the post-Trilogue text formally adopted in April 2016.

[7] The two terms are treated as synonymous in the remainder of this paper. Article 23 has since become Article 25.

> and severity for rights and freedoms of natural persons posed by the processing, the controller shall, both at the time of the determination of the means for processing and at the time of the processing itself, *implement appropriate technical and organisational measures, such as pseudonymisation, which are designed to implement data-protection principles, such as data minimisation, in an effective manner* and to integrate the necessary safeguards into the processing in order to meet the requirements of this Regulation and protect the rights of data subjects. (our emphasis)

Recital 78 uses similar language, suggesting that PbD measures 'could' involve (*inter alia*) minimisation of processing, pseudonymisation, and transparency. It states also that producers of digital systems which process personal data 'should be encouraged to take into account the right to data protection when developing and designing' those systems.

The mention of pseudonymisation and data minimisation in Article 25(1) is welcome, suggesting a starting point for the technical implementation of PbD which was absent in previous drafts of the GDPR (Article 29 Working Party 2012, 11). Nevertheless, there is little further guidance on how to achieve compliance under these provisions, leading to criticisms that they are vague (Gürses, Troncoso, and Diaz 2011), that they 'do not address technology producers [or] allow real technology design' (Pocs 2012, 644), and that they provide data controllers with 'little clue on how they should go about "designing in" privacy' (Koops and Leenes 2014, 162). Indeed, the statement in Recital 28 that the Regulation's references to pseudonymisation are not intended to preclude the use of other data protection measures suggests a certain self-consciousness about the (lack of) practical guidance provided by that instrument.

There is an institutional desire for market-driven PbD solutions that are responsive to technological and social changes (EDPS 2015, 8). But without practical

guidance on which species of technical approach are legally compliant the market will interpret the provisions minimally, potentially undermining the spirit of the Regulation.

2.2 The risks of poor PbD implementation

The ambiguity surrounding PbD is problematic, particularly given the GDPR will have direct effect. Innovation may be chilled if only those data controllers with the legal and financial resources necessary to defend their mistakes will have the confidence to develop new digital systems in areas where data protection law operates.

Smaller, dynamic innovators are likely to be discouraged by the prospect of legal action, significant financial penalties[8] and adverse publicity, should they misinterpret the PbD provisions or inadvertently fail to implement them altogether. Indeed, research has shown that awareness of privacy enhancing technologies ('PETs') in small and medium enterprises ('SMEs') is lower, and those who are aware are more apt to believe that the benefit of implementing them is outweighed by the cost (London Economics 2010). Thus SMEs may decide to take a risk and simply ignore the GDPR's requirement for PbD, especially in the initial stages after the new law comes into force and the meaning of the provisions is settling (this is perhaps reminiscent of the infamous EU 'Cookie law',[9] which has been significantly defanged by domestic data protection authorities owing to its ambiguity and the impracticality of carrying out its requirements[10]). Such an eventuality may be commercially attractive because of the de facto reduction in regulatory overhead, but it would self-evidently result in the hollowing out of the substance of Article 25.

---

8 The GDPR proposes administrative penalties of up to €10m, or 2% of worldwide turnover, for breaches of *inter alia* the Art. 25 PbD requirements (whichever amount is higher). See Art. 83(4).

9 ePrivacy Directive 2002/58/EC 2002, Art. 5.

10 The UK Information Commissioner's Office, for example, now assesses only the top 200 websites in that jurisdiction, and even then only quarterly. See <https://ico.org.uk/action-weve-taken/cookies/> (accessed 28 November 2016).

Bearing in mind the conflicts inherent in a market where the data controllers creating the systems that PbD is intended to regulate are often the same controllers who stand to gain from gathering personal data, there is a potentially concomitant effect arising from a centralisation of PbD innovation in those same companies. In the absence of true competition between PbD approaches there is perhaps an incentive for powerful, entrenched market players to form cartels, agreeing amongst themselves commercially-favourable PbD standards which minimise regulatory limitations on data processing as far as possible within the elastic limits of a mercurial legal norm. We may then be left with token gestures towards PbD rather than concrete implementations which are demonstrably effective in upholding users' rights (Gürses, Troncoso, and Diaz 2011, section 4.2).

The perceived difficulties surrounding the practical legislative guidance can to an extent be sidestepped by shifting the focus away from *post hoc* compliance with new and untested regulatory ideas towards the application of already-ingrained data protection values at a stage early enough in the design process that those values are de facto 'built in'. This is the basis of the approach described later in this paper, but before turning to that discussion we will first consider some conceptual issues surrounding the implementation of privacy-friendly digital systems.

2.3 Conceptualising PbD

The range of factors involved in designing privacy compliance into a digital system is complex. One might first consider the 'human-technical spectrum', ranging from the 'fuzzy' regulation of natural language-based policies and agreements to the concreteness of regulation-by-code (Lessig 2006). Then, one might consider the 'technical sophistication spectrum': how complex is the particular technical approach? Finally, central to the approach described below and to PbD generally, there is what

might be termed the 'design-runtime spectrum', where we ask at what point in the development and release cycle of the digital system the regulatory mechanism is implemented: nearer the beginning (the design stage), or attached to the end product as it operates out in the world (the runtime stage).

2.3.1 The human–technical spectrum

The two strands of Article 25 of the GDPR refer to the use of both technical and organisational (human) measures. Koops and Leenes (2014) are wary of too great an emphasis being placed on technical measures and consider that, due to the problems inherent in purely technology-based regulation, it will be organisational measures that are better placed to meet the aims of PbD. Spiekermann and Cranor (2009) usefully frame the two ends of this spectrum as 'privacy-by-policy' and 'privacy-by-architecture', with the former employing the traditional measures of privacy policies and other basic notice and consent mechanisms to achieve the aims of the legislation. This represents, in essence, the status quo.

At present businesses favour the primarily organisational, privacy-by-policy approach, because it can be easily applied to existing systems without much redevelopment, and it shifts responsibility onto the user. While this may have been sufficient in the past, it is now sub-optimal for at least three reasons. Firstly, from a user protection perspective the temptation will be to undermine user privacy as technical systems remain 'black boxes' from the user's perspective and because of the problems of requiring users to consent to something they cannot properly comprehend (Carolan 2016; Calo 2013). Secondly, from a commercial perspective, as users become more aware of privacy concerns in the online environment[11] they will rightly expect more

---

[11] The European Commission's most recent Data Protection Eurobarometer survey (European

explicit assurances as to how they are protected – anything less is likely to result in a chilling effect on users' activity online. Thirdly, again from a commercial perspective, businesses which focus solely on privacy-by-policy divert resources away from development of the core product onto designing and implementing policies and *post hoc* consent measures. The content of those policies might diverge from the actual technical behaviour of the system and/or the nature of the consent may be insufficient or misinformed, inviting the risk of litigation or censure from regulatory authorities, and in either case such approaches are wasteful insofar as their aims could be achieved within the design of the core product itself.

Although there are legitimate concerns about the creation of regulatory tools which rely solely on technical measures, this does not foreclose the use of computerised *assistance* in aiming to achieve regulatory compliance. The question is where the optimum point lies on the human–technical spectrum, and thereafter what level of computerised sophistication is necessary to achieve our aims, given the state of the art.

2.3.2 The technical sophistication spectrum

The technical approaches to implementing privacy within a software system have varying levels of complexity, efficacy, and expense. On the less sophisticated end of the spectrum we have the most generic PET, encryption. Its strength is that it is technologically mature and simple to implement, but the drawback is that it is blunt in its operation, with file/service access and the reciprocal provision of personal data generally being all-or-nothing.

For example, Transport Layer Security, the technology commonly used to encrypt web traffic, encrypts data transfers between the user's computer and the server

---

Commission, 2015) suggests that concerns around providing personal data online have increased since 2010.

she is communicating with. Crucially important though it undoubtedly is, TLS does nothing to guarantee the probity of the organisation that the user's browser is communicating with, since anyone can set up TLS on their web server at no cost.[12] If the organisation's processing practices are unlawful, unethical, or simply negligent, it makes little difference that the connection between its server and the user's browser is secure.[13] Simple encryption of this kind prevents fine-grained control (everything is encrypted, or nothing is encrypted), militating against user understanding and control of what is being communicated and how it is being processed, and thus contributing to the power imbalance between the controller and the user.

At the opposite end of the technical sophistication spectrum there is the translation of regulatory norms into representations which are directly comprehensible by artificial intelligence. With this 'hard-coding' (Koops and Leenes 2014), the machine can act directly on a digital 'concept' which is directly analogous to its real-world counterpart. The notional benefits can be readily appreciated: computers could interpret and enforce regulatory norms directly, without the need for time-consuming and costly legal processes or human interpretation. This is the apotheosis of Lessig's concept of code-as-law (Lessig 2006; see also Reidenberg 1997; Zittrain 2008, 107 *et seq.*). But, rather than code representing just one of Lessig's four regulatory modalities exerting force upon the regulated subject,[14] hard-coding envisions it instead as actively *subsuming* the normative content of the law, combining the already formidable power of regulation-by-code with the force of physical-world law. Such a perfectly isomorphic result is notionally ideal: legal norms, borne of the democratic legislative process, are

---

12 Indeed, the recently-launched LetsEncrypt service aims to make TLS encryption available to everyone at no cost. See <https://letsencrypt.org> (accessed 28 November 2016).

13 Some types of TLS certification can vouch for a server's ownership, but this speaks only to the question of trust and not the concrete technical behaviour of the machine.

14 According to Lessig's influential thesis (2006), the others are the market, social norms and the law. It is not proposed to consider criticisms of that thesis here.

enforced by code that can apply them perfectly (in the absolute rather than the normative sense (Lessig 2006, preface)).

While these benefits are attractive, there are substantial concerns arising from regulation by autonomous computer agents. There is a danger that the development and enforcement of regulatory norms becomes centralised in the creator of the technology, hidden behind walls of trade secrecy and proprietary code. The resulting absence of the traditional separation of powers, coupled with a lack of transparency, threaten to encourage abuses of power normally inhibited by these mechanisms of oversight (Yeung 2008, 94; Cohen 2016).

Furthermore, in contrast to circumstances where humans play a role in the processes of evaluating, adapting and enforcing regulatory norms, a kind of absolutism, or even authoritarianism, can arise from the combination of the 'ruleish' brittleness of software regulation, its resistance to (legitimate and/or exploratory) non-compliance, and its immediacy (Grimmelman 2005; Lessig 2006, 135; Brownsword 2008). A reliance on faster and cheaper 'ambient regulation' (Yeung 2008, 89-90) has the potential to discourage a deeper consideration of the normative regulatory environment within which digital systems operate, and therefore the values, rights and duties upon which they have a bearing. Similarly, programmers and systems designers may be unaware of how their values and biases can inadvertently be reflected in the products they create (Citron 2007, 1260-1263; Bamberger 2010, 706).

These sophisticated forms of hard-coding should not to be confused with less complex techniques which aim to formalise the law into logical representations of legal artefacts and relationships whose predefined connections a computer can follow in order

to achieve rudimentary automated legal reasoning.[15] The latter lie closer to the middle of the technical sophistication spectrum; the computer does not understand the intension (internal substantive meaning) of the formalised elements and relationships, merely the logical connections that have been mapped out between them.

Numerous approaches exist to distil legal provisions into this kind of computational representation, for example KORA ('Konkretisierung rechtlicher Anforderungen' – concretisation of legal requirements) (Hammer, Pordesch, and Roβnagel 2007; Hoffman et al 2012), Breaux et al's work on semantic representation of legal norms (2006), and Oberle et al's particularly relevant work on formalisation and automated legal reasoning (2012). From the perspective of those designing software systems these approaches are potentially very front-loaded, however, requiring a combination of legal and technical knowledge that the average systems designer (or lawyer) is unlikely to have, which in turn decreases the likelihood that controllers will embrace them (Otto and Antón 2007).

2.3.3 The design–runtime spectrum

While there are numerous PETs available 'off-the-shelf' which can be grafted *post hoc* onto a software system, these are not strictly privacy *by design*; privacy is not a value represented within the 'DNA' of the system if it is merely an adjunct considered only after the design of the core functionality has been completed. In such cases that core functionality does not inherently reflect values of privacy; the question of whether the original idea could have been implemented without processing personal data, with the resulting design and implementation adapted if necessary to suit, has not been asked. As Hoepman points out

---

[15] For an example of such an approach which uses an ontology to represent entities and their relationships, see R. Hoekstra et al (2007).

> [d]uring software development the availability of practical methods to protect privacy is high during actual implementation, but low when starting the project... at the start of the project, during the concept development and analysis phases, the developer stands basically empty handed. (Hoepman 2014, 1)

As discussed above, an ideal PbD solution should balance the extremes of the policy-architecture spectrum, take account of cost and the state of the art, and be sited at the point in the software development cycle where privacy values can be reflected most efficiently and economically in the design.

Much of the literature focuses on the identification of technical mechanisms which can be implemented in a software system to ensure privacy compliance, but such approaches lie at a point in the design process which *follows* the optimum stage for the implementation of PbD. Rather than focussing on the nature of the PET itself, we should instead ask whether such a technology is necessary in the first place, with the aim of giving software designers an early opportunity to side-step the issue altogether by avoiding technical behaviours which have an impact on privacy.

2.3.3.1 Rethinking 'by design'

On this view, we might identify two perspectives on the concept of 'by design'. The orthodox perspective considers whether a product or service has privacy-friendly features which mitigate or remove what would otherwise be privacy-unfriendly behaviour. This understanding of PbD relies on PETs as the mechanism for ensuring privacy-friendly behaviour. Such systems do include privacy in their design, but it is an external addition rather than being, as contemporary Silicon Valley parlance would have it, 'baked in'.

Taking a different view, we might instead focus on the software environment within which the product is designed, rather than simply the output of that environment.

Rather than burdening the end product with the technological and usability overheads that can come with PETs and other on-the-fly regulatory measures,[16] we might instead aim towards the creation of *design environments* where the aims and values of those measures are part of the creative process itself and are subsequently reflected, *by design and by default* (to quote Article 25 of the GDPR), in their output. On this view, the concept of 'by design' might therefore be embodied not in a new PET, but instead in an augmented design process which includes checks to ensure that the end product is *a priori* legally compliant, without the need for runtime execution of adjunct code designed to impose regulatory compliance on the underlying, otherwise privacy-hostile, system. The development environment will help the software designer to answer the question: is my design inherently privacy-friendly? At an abstract level, this is akin to Cavoukian's third principle of PbD:

> [Privacy] is not bolted on as an add-on, after the fact. The result is that privacy becomes an essential component of the core functionality being delivered. Privacy is integral to the system, without diminishing functionality. (Cavoukian 2011)

References in the PbD literature to the 'early design stage' (for example European Commission 2010a, note 21) remain too broad; in a software system of any complexity the 'design stage' can be so long that considerations of privacy become de facto *post hoc* if they take place late on enough. To avoid this, and to realise the alternative vision of 'by design', there needs to be some means of identifying, early on *within* that stage, where there is functionality that is potentially privacy-unfriendly.

---

[16] Whitten and Tygar (1999) discuss how privacy-aware software designers often implement technically-effective, but user-unfriendly, privacy systems, such as the Pretty Good Privacy communications encryption system.

3. Towards a novel solution using the Petri net

Conceived over fifty years ago, the Petri net is a mature, standardised (ISO 2004) formal modelling approach designed to represent processes in terms of *states* and *transitions* (Petri 1962). It has generated a significant literature applying it in many diverse fields, including banking, nuclear power, web services (ISO 2004; Hamadi and Bentallah 2003; Murata 1989, 542). Importantly, it has also been used to model legal processes (Meldman and Holt 1971; Meldman 1977; Freiheit et al 2006).

Petri nets are useful in the beginning stages of the design of 'systems of all kinds in which regulated flows of objects and information are of significance' (Reisig 1992, 1), and are particularly suited to 'by design' approaches because they allow 'specification prototypes to be developed to test ideas at the earliest and cheapest opportunity' (ISO 2004, Introduction). These features accord well with the suggestion that

> [a] first step in privacy-aware system design is to *analyze the need for information*, to *graph flows among the various system participants*, [and] to analyze *how the information flow can be minimized* (Feigenbaum et al 2002, 91, our emphasis)

Petri nets can achieve these aims simultaneously, while maintaining a simplicity of presentation that can help bridge the gap between technical and legal/policy expertise. Indeed, they were designed intentionally to facilitate an easy understanding of complex systems (Reisig 1992, 2). This apparent simplicity belies some very powerful characteristics, however. The graphical simulation of the flow of a Petri net is simple to perform,[17] and, crucially, its outcome can be formally (mathematically) verified. Importantly, then, Petri nets can balance both intuitive comprehension and analytical

---

[17] Using open source tools such as GreatSPN, which was used to draw and verify the models in this paper. See <http://www.di.unito.it/~amparore/mc4cslta/editor.html> accessed 28 November 2016. For a full list of tools, see the list at note 31, *infra*.

certainty in a way which other superficially similar modelling approaches, such as Unified Modelling Language, do not (Salimifard and Wright 2001, 667).

Since Meldman's work on United States civil procedure in the 1970s (Meldman, 1977), the literature on Petri nets as applied to legal processes is sparse. One relatively recent contribution from a European Commission FP6 project investigated methods for making judicial procedures intelligible to non-experts and to foreign legal practitioners.[18] That project was concerned mainly with abstract Petri net representations of contingent judicial processes (Freiheit et al 2006), but unfortunately it appears to have stalled, with much of its online presence now defunct.[19] Nevertheless, its assessment of the strengths of using Petri nets in the legal domain, and in particular as a method of clarifying complex processes, is instructive (Freiheit et al 2006, 22-25).

Petri nets can go some way to bridging the gap between high-level abstract thinking about processes (what lawyers and policy experts do) and low-level, detailed consideration of concrete technical behaviour (what software designers and developers do). They can thus assist systems designers in making concrete design decisions under the new regime of the GDPR, as well as in holding those decisions properly to account.

3.1 Petri nets: a brief primer

The Petri net is made up of symbols representing the *states* and *transitions* in a given process.[20] These are connected by *arcs* (arrows) representing the flow of the process. These three primitives are the essence of all Petri nets.[21] Figure 1 below shows a simple

---

18 Sixth Framework Programme, 'eJustice: Towards a global security and visibility framework for Justice in Europe' <http://cordis.europa.eu/project/rcn/74600_en.html> accessed 28 November 2016. FP6 ran from 2002 to 2006.

19 The URL for the project's Lexecute tool was <http://rechtsinformatik.jura.uni-sb.de/ejustice/lexecute/>.

20 In the literature the term 'places' is sometimes used instead of 'states'. The latter implies a status rather than a physical location, however, so seems more appropriate for our purposes.

21 Note that this is necessarily a brief overview of Petri nets and their basic concepts, as a fuller

net.[22]

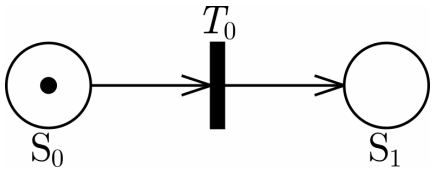

Figure 1: A simple Petri net

States are represented by circles, while transitions are represented by a rectangle.[23] Although there are several variants of the Petri net, this paper focuses on the 'timed' variant, which allows transitions to be prioritised such that that they *fire* (occur) in a particular order.

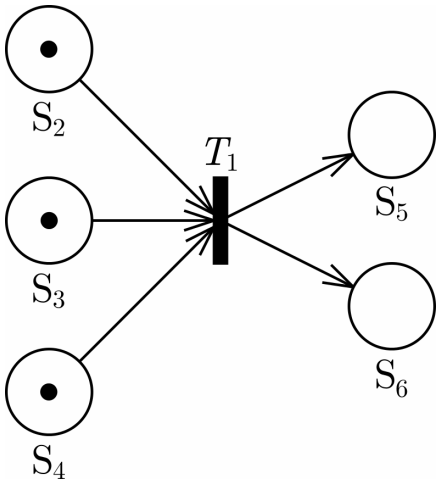

Figure 2: Multiple simultaneous states in a Petri net

A state containing a *token* (a dot) 'holds', which is to say that that state, and any others which might hold, represent the status of the process at any given moment. A particular status (configuration of tokens) is called the net's *marking*.

---

exposition of the wealth of literature they have generated is outwith the scope of this paper. For a starting point with the literature, see Petri's original thesis (1962) (in German), or Murata (1989).

22 These figures are inspired by those used in Meldman (1977), because of their clarity.

23 Or sometimes a square, or a line perpendicular to the arrow.

A transition can only fire if the states which lead to it all hold, and once a transition fires all states leading to it will no longer hold and those which lead from it will at that point hold. If an arc has a number next to it, this means it can only 'carry' that number of tokens, and so the transition it points to can fire only if the preceding state holds at least that many tokens. No number means the default (one token) is required. For example, in Figure 3 the transition cannot fire because there is only one token, and not two or more, in S_0.

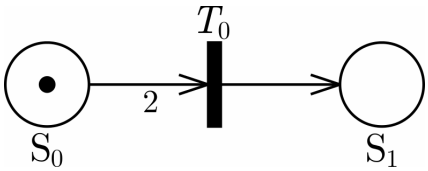

Figure 3: Arc weighting

Two or more transitions can be in conflict with one another, whereby only the transition with the necessary preceding states will 'win'. Figure 4 demonstrates this – there T_2 will win over T_1, and thus S_5 will hold.

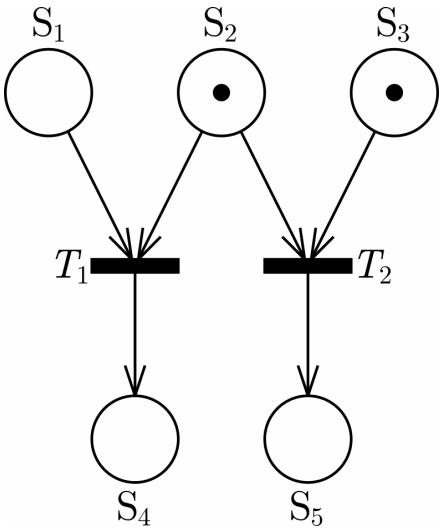

Figure 4: Competing transitions

This limited palette of symbols allows complex real-world processes to be reduced to simple logical representations that are especially suited to computational simulation and analysis. In complex nets the transitions themselves can represent nested 'sub-nets', thus mirroring the basic architecture of object-oriented computer programming, where discrete methods process input data and pass on their output to other functions (in fact, nesting nets in this way is sometimes referred to as 'object Petri nets').

Meldman applies this concept in his complex Petri net model of United States civil procedure (Meldman 1977, 141 *et seq.*). There he identifies repeating sub-nets,[24] each of which he abstracts into a generic, black-boxed (in the cybernetic sense[25]) 'function' that can be called upon repeatedly, as required, in the overall net (for example, 'apply for court hearing'). Thus a very complex model, with potentially hundreds of symbols, can be transformed into a smaller number of high-level nets that are abstracted to a level that can aid legal analysis without forsaking technical isomorphism. This concept is demonstrated in Figure 6 below.

3.1.1 A basic example

An example might help to illustrate some of the above ideas to a non-technical lawyer. Consider a legal norm that prescribes a legal consequence if a set number of conditions is met, for example 'expert evidence will be supressed from the trial if the expert is not accredited by an appropriate professional body and that non-accreditation has been raised by the defence'. Here, we have two 'triggering conditions', one substantive (the

---

24 Meldman calls them 'general events'.

25 Indeed, the characterisation of this form of black box by STS scholars Kendall and Wickham (1999) is directly analogous to an abstracted Petri net: 'arrows indicate what goes into the black box, and what goes out, but the actual contents and workings of the box are not examined.' (1999, 83). This functional definition is in contrast to Pasquale's normative critique of the 'black box' as a broader socio-technical phenomenon (2015).

non-accreditation) and one procedural (the objection was raised). If they are both met, the consequence is 'fired', that is the whole system moves into a state where the evidence of that expert is not permitted during trial. In addition to this we can imagine that, just as in many real-world legal systems, such an objection can be raised by the defence only once, in order to prevent undue delay and repeated ruling on the same issue.

Each of these elements can be represented by a Petri net. Returning to Figure 4 above, S_2 could represent the objection of the defence against the expert, S_1 could represent 'expert is not accredited, but has all the qualifications needed for accreditation', and S_3 could represent 'expert is not accredited and also lacks the qualifications that are required for accreditation'. The two possible 'result states', S_4 and S_5, would in this case represent 'the trial is delayed until the expert is accredited' and 'the evidence is ruled inadmissible and the trial continues', respectively. Like these real-world procedural consequences, the subsequent flow of the Petri net will be altered depending on which one of these two states holds. Furthermore, the idea of the objection being 'single use' maps onto the Petri net's 'token' mechanism; once the litigant raises it the relevant token is exhausted, much like their opportunity to repeat that same objection in the real-world courtroom. With these concepts in mind, we now apply the approach to PbD.

3.2 Petri nets and privacy by design

The Petri net is potentially a very powerful tool for the implementation of PbD. A prototype digital system, and its constituent parts, can be repeatedly tested according to a model of the relevant part(s) of data protection law. Designers and developers can thus focus on their core task of implementation without having to take on the cognitive load

of recalling and applying a complex body of regulatory norms – a task which if carried out inadequately will be to the detriment of regulatory compliance.

From a wider perspective, a modelling process of this kind can also provide 'clarification of ambiguities and inconsistencies in the natural language descriptions of systems' (Meldman and Holt 1971, 65). Interestingly, Meldman observes from his Petri net that a possible procedural outcome which was not readily intelligible from a reading of the legislative provisions alone came 'right to the surface when attempting to describe the rules in the Petri-net language' (Meldman 1977, 145).

The broader potential then is not just in passively aiding systems designers and developers in applying extant regulation, but also, through the clarification and abstraction of complex legal norms, to promote a greater general awareness of that regulation within the technical environment. If the modelling process can give designers 'a detailed understanding of the relevant processes as well as stakeholder needs' (Spiekermann and Cranor 2009, 69), we might then move closer to what should surely be the overarching goal of privacy 'by design': systems designers internalising and applying the substance of regulatory norms reflexively, without the need for an external tool to prompt them.

3.2.1 Modelling data protection law

To show the Petri net in context this section models one of the provisions of the Data Protection Directive (hereinafter, the 'DPD').[26] Many of the DPD's provisions might be thought of as test questions which the controller is expected to ask herself before engaging in processing, such as 'are these data personal?', 'do I need to seek explicit consent before processing?', 'is there an exception that permits processing?', and so on.

---

[26] Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data.

Each question is a 'gate' that affects which question(s) must subsequently be asked, and ultimately whether the processing is lawful according to the Directive. If (and only if) the answer to a given question is positive are the data permitted to 'travel' along the arcs of the system, for example from the user's device to the online platform provider – just as electricity can flow along the arcs of a circuit only when all the gates are in the right state.

Figure 5 shows a Petri net model of Article 8 of the DPD, which concerns the processing of sensitive categories of data. In practice the initial marking of the net would be set by the outputs of other nets; this example is intentionally simple to give an overview of how the approach might work. If sensitive data are being processed (top-right box), processing_can_continue cannot be reached, unless one of the exceptions applies (bottom-right box). In this example, sensitive data are present but, because there is no applicable exception, the processing_can_continue state cannot be reached. The Petri net is thus at an impasse, or is *deadlocked*.[27]

[27] Note the *inhibitor arc* between sensitive_data_present and the non_sensitive_data, which flips the logic so that the absence of a token causes the transition to fire.

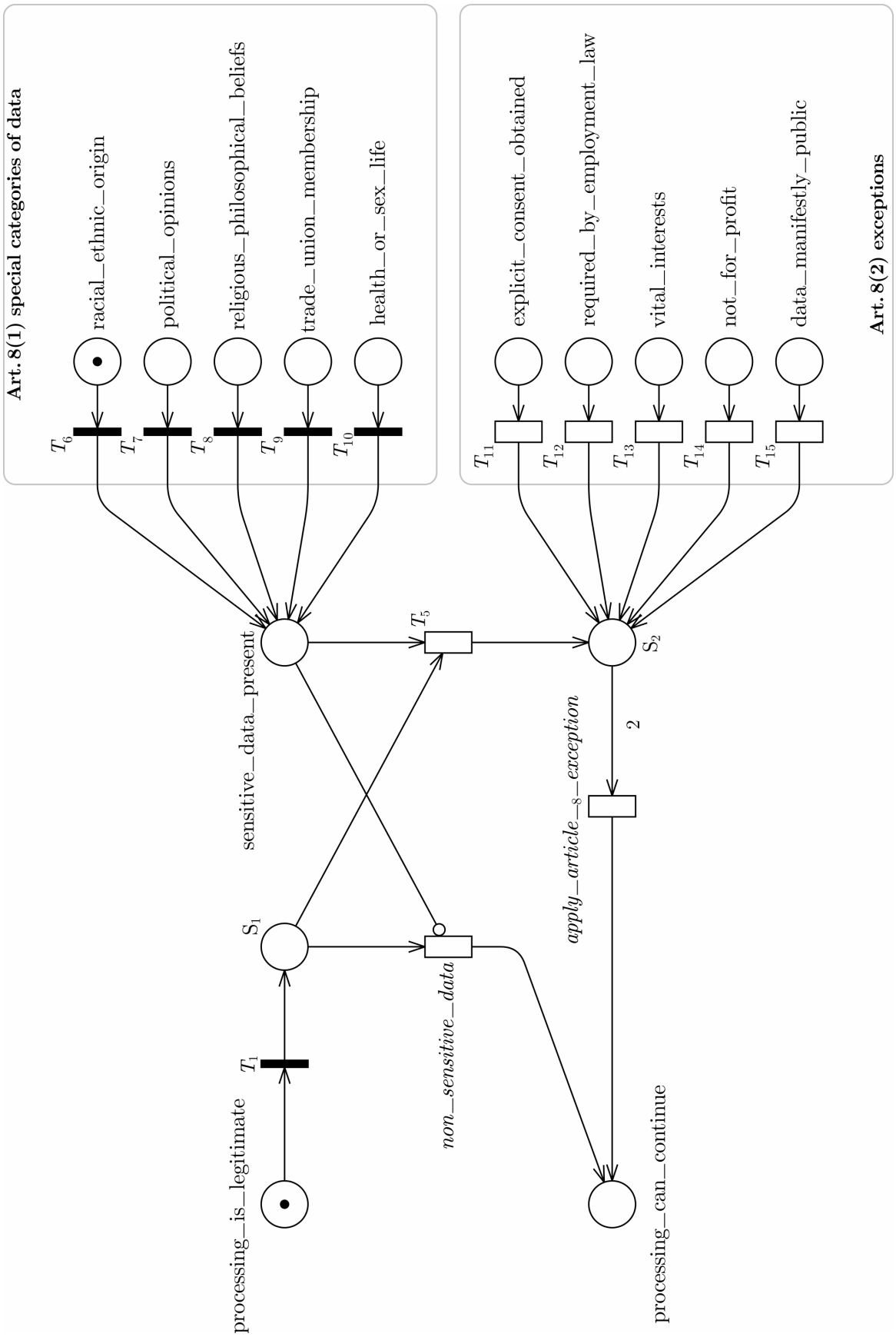

Figure 5: Petri net of Article 8 of the Data Protection Directive 1995

3.2.2 Interlinking software and legal models

Figure 5 is a simple model representing a part of the legal, rather than the software, process. But the Petri net approach permits the interlinking of both. For example, in Figure 6, in order for the software process to continue from S_0 to S_4, the legal net must be traversed. Evidently this is an abstracted example, but we can imagine the legal net in this case being a 'black boxed' version of Figure 5 above, which we could 'open' if we desired to see the specifics of the test being performed therein. The information required to perform the test in Figure 5 might be supplied by other parts of the software system – in that case whether there are any sensitive data present. This is demonstrated abstractly by the state S_2 in Figure 6.

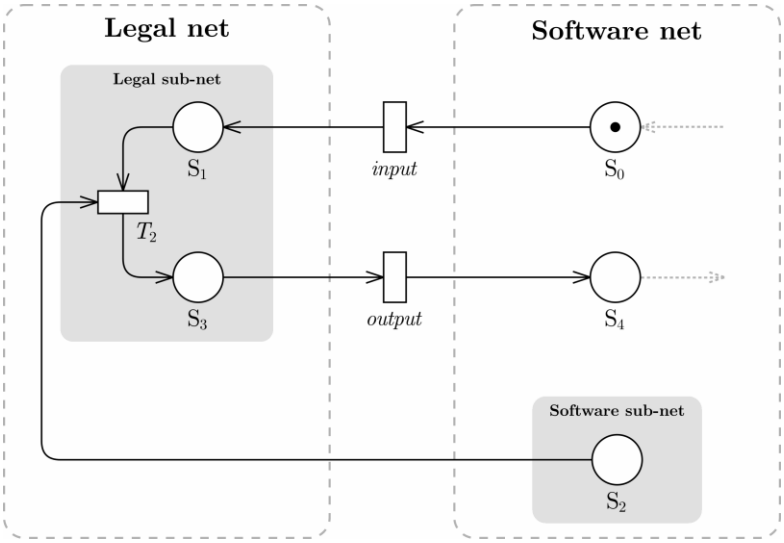

Figure 6: Communication between legal and software models

A real-world example will illuminate this concept further. We might imagine that the right side of Figure 6 represents a software system which presents the user with a registration form that must be completed in order to use the service in question. On that form, there is an optional question asking the user to specify their ethnic origin. Should they choose to do so, the fact of their having provided that information can be

communicated to the legal net by means of S_2 holding. If the user did supply that information, then Article 8 sensitive data have been gathered, and if not, then they have not (*ceteris paribus*, of course). Passing the Article 8 test modelled above is thus contingent on this element of technical functionality, and the effect of either possibility is readily intelligible to legal and technical experts alike.

This is of course a very abstract and simplified example; the approach would in practice include states and transitions sufficiently detailed to map accurately the inputs and outputs of both the legal and software processes.

3.3 Verifying the model

In software development there are two main approaches to verifying an application's behaviour. Dynamic analysis is used on 'live' (compiled[28]) code running in a testing environment analogous to the end user's. Static analysis, on the other hand, is like a theoretical test run, where rather than emulating the software's operation in a real-world context, it is the underlying logic which is evaluated. The strength of static analysis is that it 'can be used in the initial phase of testing to identify definite program errors such as deadlocks that are guaranteed to occur' (Shatz and Cheng 1988, 343). As a tool for static analysis Petri nets are a tool well suited to this aim because

> [they allow] specification prototypes to be developed to test ideas at the earliest and cheapest opportunity. Specifications written in the technique may be subjected to analysis methods to prove properties about the specifications, before implementation commences, thus saving on testing and maintenance time and providing a high level of quality assurance. (ISO 2004, Introduction)

These attributes are self-evidently attractive from a commercial perspective where the compliance of prototype designs is required by law. Research has shown how

---

[28] Compilation is the process of turning source code into machine-executable instructions.

to represent a software system as a Petri net ready for inclusion in the kind of process described here. For example, Shatz and Cheng (1988) demonstrate one method for translating Ada code into a Petri net ready for analysis. The concepts have been also applied in other contexts also relevant to our purposes, including more common C-based programming languages (Lin 1998),[29] business process modelling (Hinz, Schmidt, and Stahl 2005), and web services (Hamadi and Bentallah 2003). The wide range of literature demonstrates the flexibility of Petri nets, a characteristic that could mitigate some of the friction likely to arise in attempting to implement a singular approach across the working processes of a diverse range of data controllers and processors.

As mentioned above, a crucial aspect of the Petri net approach is the possibility of formally (that is, mathematically) proving the 'reachability' of the states within the net. Provided the legal and software systems have been modelled accurately,[30] one can thus certify that the design of a given system is or is not compliant with a particular law – that is, it allows only those data flows that are not prohibited.

This certification can be achieved through a 'reachability analysis' to determine, for a given initial status of the net, whether a particular state can ever hold. In a net such as that in Figure 6, if the reachability analysis determines that the state S_4 can hold, then the legal test has been 'passed' and the software system can continue.

The literature on Petri net provability is highly technical, and a replication of a formal proof is outwith the scope of this paper. It should be noted, however, that the analytical methods are well-established and have been integrated into numerous

---

29 Most modern software is written in C, or languages based on C.

30 The question of isomorphism is, of course, a crucial one. See for example Bench-Capon and Coenen (1992).

software analysis tools which are already available.[31] The novelty in what is proposed here lies in finding a new application for what is a well-understood technology. Proper integration of such tools into software development environments would mean that the underlying mathematical analysis need not be exposed to the software designer, who can instead focus on modelling and testing her proposed system.

3.4 Leveraging the analysis: triaging deadlocks

Returning to Figure 6, if the software net cannot proceed to S_4 because sensitive data are being gathered and no Article 8(2) exception applies, then the system hits a deadlock. Knowing this early on in the software development process gives the designer, and the wider enterprise, a chance to consider three options:

(1) reassessing the overall functionality and aims of the system to determine whether, given the business model being pursued, it would be feasible to achieve the same functionality without gathering those data;
(2) redesigning the relevant parts of the system to incorporate technical measures which will make the functionality in question compliant with the regulation; or
(3) changing the business model altogether if its aims are found to be fundamentally incompatible with the requirements of the law.

The first option gives the data controller an opportunity to assess the prototype system's behaviour, perhaps with a view to altering it so that the collection and use limitation principles of data protection are applied more strictly.[32] This accords both with the GDPR's suggested use of processing minimisation as a potential strategy for achieving

---

[31] For an exhaustive list, see University of Hamburg, 'Petri Nets World: Petri Nets Tools Database Quick Overview' <http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html> accessed 28 November 2016.
[32] DPD, Art. 6(1)(b) and (c); GDPR Art. 5(1)(b) and (c).

PbD,[33] and with the view that technical responses to data protection regulation should focus first on data minimisation, given the present difficulty, discussed above, of implementing more sophisticated forms of automated regulatory compliance (and see Gürses, Troncoso, and Diaz 2011).

The second option will usually involve a greater investment of development resources, but when coupled with further rounds of modelling and testing, the incorporation of a PET or perhaps a privacy design pattern (Hafiz 2006) at this early stage of the design process can fulfil Cavoukian's requirement that 'privacy [become] an essential part of the core functionality being delivered' (Cavoukian 2011, Principle 3).

The third option will be unattractive to most businesses, particularly start-up companies or those on the cusp of developing a disruptive new product or service who will not want creative energy and momentum to go to waste. If a product is ultimately unworkable, however, it is self-evidently preferable to halt its development in the early stages of the design process rather than later on when further creative and financial resources have been invested and the resulting loss is all the greater.

3.5 Changing duties under the GDPR

3.5.1 Accountability

The formal proofing capabilities of the Petri net could also assist in the fulfilment of the new assessment requirements contained in the GDPR. These represent a shift away from the current regime's increasingly inadequate focus on notification to a system based around the principle of accountability,[34] with a focus on internal record-keeping

---

33 GDPR, Recital 78; Art. 25(1).
34 GDPR, Art. 5(2).

and auditability. Article 24(1) requires data controllers to be able to demonstrate that the processing they carry out is 'performed in accordance with this Regulation', while Article 30 creates an obligation to 'maintain a record of processing activities under [the controller's] responsibility'. Notably, Article 35 requires controllers to perform a 'data protection impact assessment' in cases where it considers that the processing it seeks to carry out 'is likely to result in a high risk to the rights and freedoms of natural persons'.[35] The impact assessment should contain *inter alia* 'a systematic description of the envisaged processing operations'[36] and

> [a description of] the measures envisaged to address the risks, including safeguards, security measures and mechanisms to ensure the protection of personal data and to demonstrate compliance with this Regulation...[37]

Although much depends on the eventual interpretation of these provisions, it is conceivable that the approach described here could contribute much to the meeting of these new documentation requirements. Formal verification can demonstrate that particular states in the model are definitively avoided or reached (and therefore that particular legal tests are passed or failed, as in Figure 5 above), while the graphical model itself can serve as accessible documentation of the system being implemented.

Appropriately-trained staff of the data protection supervisory authority might readily understand such 'documentation', and its generation as part of the design process might allay fears that the GDPR's new controller duties will create disproportionate expense for SMEs and start-ups (de Hert and Papakonstantinou 2016, 192; London Economics 2010).

---

35 *Ibid.* Art. 35(1).
36 *Ibid.* Art. 35(7)(a).
37 *Ibid.* Art. 35(7)(d).

3.5.2 Certification and promulgation

Related to the documentation requirements are the provisions regarding certification. Recital 60 obliges data controllers to 'be able to demonstrate the compliance of processing activities', while Recital 60(c) refers to 'approved certifications'. Article 23(2a) states that '[a]n approved certification mechanism pursuant to Article 39 may be used as an element to demonstrate compliance with the requirements in paragraphs 1 and 2'. Under Article 39(5), an appropriate regulatory body (perhaps including the European Data Protection Board, proposed under the GDPR[38]) could develop, maintain and promulgate certified models of the relevant provisions, using their access to in-house legal expertise to maintain their accuracy. Alternatively, there may be scope for the development of such models on a 'crowd-sourced' basis, where the task of creating and updating models of the relevant provisions is opened out to the legal community, with the final decision on whether a given model is 'release-quality' remaining with the relevant regulatory authority.[39]

Once a Petri net of the relevant legal provision has been created and made publicly available, software designers would 'plug in' Petri nets of their proposed designs using the pre-determined inputs and outputs of the legal model as hooks. This would enable them to leverage the regulators' legal expertise, embodied in the legal model, without the need to attempt to perform or commission an expensive and time-consuming legal assessment. If the proposed digital system passes the verified legal model this might be a material contribution to fulfilling the GDPR's certification

---

38 See Art. 68. The EDPB is intended to replace the Article 29 Data Protection Working Party established under the DPD.

39 This is very much akin to the development process of major open source projects such as the Linux kernel, where anyone can contribute code but the final determination as to which contributions are of sufficient quality to be included in a public release is made by a small board of core maintainers. See Moody (2002, 179).

requirements, and could in turn be commercial selling point for the data controller, perhaps promoted by, for example, a new consumer symbol.[40]

4. Conclusions

The approach described in this paper gives systems designers a powerful opportunity to test the compliance of their designs with data protection law, at a stage in the product development process where wasted investment can be kept to a minimum.

As will have been evident, however, the technique described need not be limited to data protection law; it is agnostic with regard to the legal fields that can be modelled and assessed, and thus the approach has the potential to assist those designing digital systems whose functionality has regulatory implications (that is, almost all of them).

One might therefore imagine a more holistic approach to 'by design', where code is verified against models of provisions from heterogeneous fields of law. By enabling the deep integration of regulatory norms early on in the design process, we can balance on the one hand the need to retain a democratic connection between the creation of regulation and the locus of its operation, and on the other the desire to invent and develop new digital products and services.

4.1 Future research

Further inter-disciplinary research is required to develop the approach outlined above and to consider in greater detail questions such as the extent to which Petri nets can be derived from existing source code, the appropriate level of technical and legal isomorphism that can and should be represented in a Petri net (and therefore what normative content is left out of it) (Bench-Capon and Coenen 1992), and the most

---

[40] For a relevant discussion of data protection and the existing CE mark, used in Europe to signify compliance with certain minimum safety, health and environmental standards, see Lachaud (2016).

appropriate Petri net variant to use (for example, Raskin et al (1996) extend the basic net to include deontic states, which may be suitable for modelling more normative permission-based systems, including legislative provisions).

References

Article 29 Data Protection Working Party. 2012. 'Opinion 01/2012 on the Data Protection Reform Proposals' WP 191

Bamberger, Kenneth A. 2010. 'Technologies of Compliance: Risk and Regulation in a Digital Age' *Texas Law Review* 88 (4): 669.

Bench-Capon T.J., and F.P. Coenen. 1992. 'Isomorphism and Legal Knowledge Based Systems' *Artificial Intelligence and Law* 1:65

Breaux, Travis D., Matthew W. Vail, and Annie Antón. 2006. 'Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations'. In *Requirements Engineering, 14th IEEE International Conference*, 49–58

Brownsword, Roger. 2008. 'So What Does the World Need Now? Reflections on Regulating Technologies' in *Regulating Technologies: Legal Futures, Regulatory Frames and Technological Fixes*, edited by Karen Yeung and Roger Brownsword, pp. 23–49. (Oxford: Hart)

Calo R. 2013. 'Code, Nudge, or Notice' *Iowa Law Review* 99:773

Carolan E. 2016. 'The Continuing Problems with Online Consent under the EU's Emerging Data Protection Principles' *Computer Law & Security Review* 32:462

Cavoukian, Ann. 2012. 'Privacy by Design: Origins, Meaning, and Prospects for Assuring Privacy and Trust in the Information Era' in *Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards: Aspects and Standards*, edited by George O.M. Yee. (IGI Global)

Cavoukian, Ann. 2011. 'Privacy by Design: The 7 Foundational Principles'. https://www.iab.org/wp-content/IAB-uploads/2011/03/fred_carter.pdf

Cisco Systems. 2011. 'The Internet of Things: How the Next Evolution of the Internet Is Changing Everything' http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

Citron, Danielle Keats. 2007. 'Technological Due Process' *Washington University Law Review* 85: 1249–1313.

Citron, Danielle Keats. 2008. 'Open Code Governance' *University of Chicago Legal Forum* 355

Cohen, Julie E. 2016. 'The Regulatory State in the Information Age' *Theoretical Inquiries in Law* 17(2).

Danaher J. 2016. 'The Threat of Algocracy: Reality, Resistance and Accommodation' *Philosophy & Technology* 29:245

EDPS. 2015. ‘EDPS Recommendations on the EU’s Options for Data Protection Reform’. Opinion 3/2015

European Commission. 2010a. ‘A Digital Agenda for Europe’. COM/2010/0245 final

European Commission. 2010b. ‘A Comprehensive Approach on Personal Data Protection in the European Union’. COM(2010) 609 final

European Commission. 2012. ‘Big Data at your Service’ https://ec.europa.eu/digital-single-market/en/news/big-data-your-service

European Commission. 2015. ‘Data Protection Eurobarometer’ http://ec.europa.eu/justice/newsroom/data-protection/news/240615_en.htm

Feigenbaum, Joan, Michael J. Freedman, Tomas Sander, and Adam Shostack. 2002. ‘Privacy Engineering for Digital Rights Management Systems’ in *Security and Privacy in Digital Rights Management*, edited by Tomas Sander, pp. 76–105. Lecture Notes in Computer Science 232. (Springer Berlin Heidelberg)

Freiheit, Jorn, Marc Luuk, Susanne Munch, and Grozdana Sijanski. 2006. ‘Lexecute: Visualisation and Representation of Legal Procedures’. *Digital Evidence & Elec. Signature L. Rev.* 3:19

Grimmelmann J. 2005. ‘Regulation by Software’ *The Yale Law Journal* 114:1719

Gürses, Seda, Carmela Troncoso, and Claudia Diaz. 2011. ‘Engineering Privacy by Design’. *Computers, Privacy & Data Protection* 14(3)

Hafiz, Munawar. 2006. ‘A Collection of Privacy Design Patterns’ *ACM Proceedings of the 2006 Conference on Pattern Languages of Programs* 7

Hamadi, Rachid, and Boualem Benatallah. 2003. ‘A Petri Net-Based Model for Web Service Composition’. In *ADC Proceedings of the 14th Australasian Database Conference* 17:191–200

Hammer, Volker, Ulrich Pordesch, and Alexander Roßnagel. 1993. ‘KORA – Eine Methode Zur Konkretisierung Rechtlicher Anforderungen Zu Technischen Gestaltungsvorschlägen Für Informations-Und Kommunikationssysteme’. *Infotech/I+ G* 21

de Hert, Paul, and Vagelis Papakonstantinou. 2016. ‘The New General Data Protection Regulation: Still a Sound System for the Protection of Individuals?’ *Computer Law & Security Review* 32(2):179–94. doi:10.1016/j.clsr.2016.02.006

Hoekstra, Rinke, Joost Breuker, Marcello Di Bello, and Alexander Boer. 2007. ‘The LKIF Core Ontology of Basic Legal Concepts’. *LOAIT* 321:43

Hoepman, Jaap-Henk. 2014. ‘Privacy Design Strategies’ in *ICT Systems Security and Privacy Protection*, pp. 446–459 (Springer)

Hoffmann, Axel, Thomas Schulz, Holger Hoffmann, Jandt, Silke, Alexander Roßnagel, and J. M. Leimeister. 2012. 'Towards the Use of Software Requirement Patterns for Legal Requirements'. SSRN Scholarly Paper ID 2484455. Rochester, NY: Social Science Research Network. http://papers.ssrn.com/abstract=2484455

ISO. 2011. 'Information Technology - Security Techniques - Privacy Framework' BS ISO/IEC 29100:2011. ISO/IEC

ISO. 2004. 'Systems and Software Engineering. High-Level Petri Nets. Concepts, Definitions and Graphical Notation' Standard 15909–1:2004+A1:2010

Kendall G. and Wickham G. 1999. *Using Foucault's Methods* (Sage Publications)

Kitchin R. 2016. 'Thinking Critically About and Researching Algorithms' *Information, Communication & Society* 1

Koops, Bert-Jaap, and Ronald Leenes. 2014. 'Privacy Regulation Cannot Be Hardcoded. A Critical Comment on the "Privacy by Design" Provision in Data-Protection Law'. *International Review of Law, Computers & Technology* 28(2):159–71. doi:10.1080/13600869.2013.801589

Lachaud, Eric. 2016. 'Could the CE Marking Be Relevant to Enforce Privacy by Design in the Internet of Things?' in *Data Protection on the Move*, edited by Serge Gutwirth, Ronald Leenes, and Paul De Hert, pp. 135–62. Law, Governance and Technology Series 24 (Springer Netherlands)

Latour, Bruno. 1987. *Science in Action: How to Follow Scientists and Engineers through Society* (Cambridge, Massachusetts: Harvard University Press)

Lessig, Lawrence. 2006. *Code: Version 2.0* (New York: Basic Books)

Lin, Bill. 1998. 'Software Synthesis of Process-Based Concurrent Programs' in *ACM Proceedings of the 35th Annual Design Automation Conference*, 502–505

London Economics. 2010. 'Study on the Economic Benefits of Privacy-Enhancing Technologies (PETs), Final Report to the European Commission, DG Justice, Freedom and Security'

London, Ralph L. 1972. 'The Current State of Proving Programs Correct' in *Proceedings of the ACM Annual Conference*, 1:39–46

London R.L., 1972. 'The Current State of Proving Programs Correct' in *Proceedings of the ACM annual conference*

Meldman, Jeffrey A. 1977. 'A Petri-Net Representation of Civil Procedure' *Idea* 19:123

Meldman, Jeffrey A., and Anatol W. Holt. 1971. 'Petri Nets and Legal Systems' *Jurimetrics Journal* 12(2):65–75

Moody, Glyn. 2002. *Rebel Code* (New York: Basic Books)

Murata, Tadao. 1989. 'Petri Nets: Properties, Analysis and Applications' in *Proceedings of the IEEE* 77(4):541–580

Oberle, Daniel, Felix Drefs, Richard Wacker, Christian Baumann, and Oliver Raabe. 2012. 'Engineering Compliant Software: Advising Developers by Automating Legal Reasoning' *SCRIPTed* 9(2): 280–313

Otto, P.N., and A.I. Anton. 2007. 'Addressing Legal Requirements in Requirements Engineering' in *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*, 5–14. doi:10.1109/RE.2007.65

Pasquale, Frank. 2015. *The Black Box Society: The Secret Algorithms That Control Money and Information* (Harvard University Press, Cambridge)

Petri, Carl Adam. 1962. 'Kommunikation Mit Automaten'. PhD diss., University of Bonn. http://epub.sub.uni-hamburg.de/informatik/volltexte/2011/160/

Pocs, Matthias. 2012. 'Will the European Commission Be Able to Standardise Legal Technology Design without a Legal Method?' *Computer Law & Security Review* 28(6):641–50. doi:10.1016/j.clsr.2012.09.008

Raskin J.-F., Tan Y.-H. and van der Torre L.W. 1996. 'Modeling Deontic States in Petri Nets' (Erasmus University)

Reidenberg, Joel R. 1997. 'Lex Informatica: The Formulation of Information Policy Rules through Technology' *Texas Law Review* 76:553

Reisig, Wolfgang. 1992. *A Primer in Petri Net Design* (Springer)

Schwartz, Paul M., and Daniel J. Solove. 2014. 'Reconciling Personal Information in the United States and European Union' *California Law Review* 102:877

Shatz, S. M., and W. K. Cheng. 1988. 'A Petri Net Framework for Automated Static Analysis of Ada Tasking Behavior' *Journal of Systems and Software* 8(5):343–59. doi:10.1016/0164-1212(88)90027-1

Spiekermann, S., and L.F. Cranor. 2009. 'Engineering Privacy' in *IEEE Transactions on Software Engineering* 35(1):67–82. doi:10.1109/TSE.2008.88

Verheij B. 2007. 'Argumentation Support Software: Boxes-and-Arrows and Beyond' *Law, Probability and Risk* 6:187

Voyatzis P. and Schäfer B. 2012. 'The Battle of the Precedents: Reforming Legal Education in Mexico Using Computer-Assisted Visualization' in *The Arts and the Legal Academy: Beyond Text in Legal Education*, edited by Zenon Bankowski, Paul Maharg and Maksymilian Del Mar, pp. 149-68 (Routledge)

Whitten, Alma, and J. Doug Tygar. 1999. 'Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0' in *Usenix Security* 1999

Yeung, Karen. 2008. 'Towards an Understanding of Regulation by Design' in *Regulating Technologies: Legal Futures, Regulatory Frames and Technological Fixes*, edited by Karen Yeung and Roger Brownsword, pp. 79–108 (Oxford: Hart)

Zittrain, Jonathan. 2008. *The Future of the Internet and How to Stop It* (New Haven: Yale University Press)