Eingereicht und angenommen für den Konferenzband „Die Fortentwicklung des Datenschutzes" (http://dx.doi.org/10.1007/978-3-658-23727-1), dort jedoch aufgrund unannehmbarer Forderungen und mangelnder Verhandlungsbereitschaft des Verlags nicht veröffentlicht.

Erfolgsfaktoren für den Datenschutz durch Technikgestaltung

Sven Türpe, unabhängiger Wissenschaftler, <tuerpe@acm.org>
Andreas Poller, Fraunhofer SIT, <andreas.poller@sit.fraunhofer.de>

Keywords

Privacy by Design, Softwareentwicklung, Entwicklungsprozesse

Abstract

Datenschutz als Gestaltungsziel für IT-Systeme hat es schwer. Der kreative Prozess der Entwicklung definiert sein eigenes Ziel erst im Wechsel von Erkunden und Entscheiden. Agile Methoden bieten dafür einen Rahmen, ohne jedoch die Entwicklung im Detail zu steuern. Das Querschnittsthema Datenschutz geht dabei leicht unter. Zum einen können ihm ökonomische Anreize entgegenwirken, woran Eingriffe in Entwicklungsprozesse wenig ändern. Zum anderen kann er an geringer Sichtbarkeit und fehlendem Stakeholder-Interesse scheitern.

Einleitung

Datenschutz durch Technikgestaltung oder „by Design" erscheint vordergründig als eine Forderung an die Technik oder im weiteren Sinne an soziotechnische Systeme. Im Fokus dieser Perspektive stehen IT-Systeme als Erzeugnisse, die mit ihrer Architektur, ihren Funktionen und sonstigen Merkmalen den Zielen des Datenschutzes mehr oder minder gut gerecht werden. Die Eigenschaften eines Systems sind nicht willkürlich festgelegt, sondern Ergebnis seines Gestaltungs- und Entwicklungsprozesses. Das Verlangen nach Datenschutz „by Design" richtet sich auf dieses Ergebnis, doch verwirklicht wird es indirekt durch Beeinflussung der Entwicklungsarbeit.

Defizite im Entwicklungsergebnis erscheinen rückblickend oft offensichtlich. So attraktiv jedoch die Vorstellung wirkt, Datenschutzbelange systematisch bereits von Beginn der Entwicklung an zu berücksichtigen, so schwer ist sie umzusetzen. Entwicklungsprozesse haben häufig kein klar und detailliert festgelegtes Ziel, sondern sie erkunden einen großen, vieldimensionalen Entwurfsraum auf der Suche nach möglichst guten Lösungen für anfangs nur grob umrissene Probleme.

Dies wirft die Frage auf, wie sich die Gestaltung von IT-Systemen beeinflussen lässt und welche Faktoren den Umgang mit dem Thema Datenschutz im Entwicklungsprozess bestimmen. Welche Schwierigkeiten sich hinter dieser Frage verbergen, zeigen Beispiele wie die

Telematikinfrastruktur für das Gesundheitswesen und die eID-Funktion des Personalausweises. Die Gesundheitstelematik wird seit Anbeginn um Sicherheitsmechanismen herum konstruiert – mit spärlichen Ergebnissen und inzwischen zum Teil museumsreifen Konzepten. Im Gegensatz dazu ist die datenschutzfreundliche eID-Funktion des Personalausweises seit Jahren verfügbar, wird jedoch kaum genutzt.

In beiden Fällen lag das Augenmerk vor allem auf der Technik und weniger auf den Vorgängen ihrer Gestaltung. Weithin üblich sind heute agile Entwicklungspraktiken mit kurzen Entwicklungszyklen, welche die Charakteristik von Gestaltungsprozessen aufnehmen. Mit bürokratischen Mitteln wie rigiden Maßnahmenkatalogen oder Prozessschablonen lässt sich der Datenschutz darin nicht fördern. Stattdessen kommt es darauf an, ihn fortlaufend als Entwicklungsziel zu vertreten, das sichtbar und genügend wichtig ist. Am besten gelingt dies, wenn die Ziele des Datenschutzes ohne Nachhilfe von den Stakeholdern eines Systems ausgehen und gleichzeitig mit verfügbarer Technik zu erreichen sind.

Verlockung und Herausforderung

In abstrakter Form ist die Forderung nach Datenschutz durch Technikgestaltung oder Privacy by Design leicht in den Raum gestellt. Die Schutzziele[1] des Datenschutzes sollen nicht nur formal, oberflächlich und durch nachträgliche Eingriffe erfüllt werden, sondern sich inhärent in der Gestaltung (sozio-)technischer Systeme niederschlagen. Diese Idee begleitet den Datenschutz schon länger, etwa in Gestalt der organisatorisch-technischen Maßnahmen aus dem alten BDSG oder der Privacy-Enhancing Technologies (PET), mit denen sich vor allem die Forschung beschäftigt.

In der IT-Sicherheit, dem in vieler Hinsicht engsten Verwandten des Datenschutzes mit teils denselben Schutzzielen und Mechanismen, findet sich unter der Bezeichnung „Security by Design“ ein ähnlicher Gedanke. Der Hintergrund dort ist, dass sich die Sicherheit eines Systems nicht durch die Beseitigung einzelner Schwachstelleninstanzen verbessern lässt, sondern nur durch grundlegende Maßnahmen, die ganze Klassen von Problemen entschärfen. Fragen der Sicherheit müssen deshalb durch den ganzen Entwicklungsprozess hindurch beachtet und behandelt werden. Infolgedessen entstanden Praktiken und Werkzeuge für die Entwicklung sicherer Software und Systeme[2]. Die damit gesammelten Erfahrungen lassen sich auf den Datenschutz übertragen.

Schnell zeigen sich in der Praxis allerdings Schwierigkeiten und Grenzen. Zwei Beispiele mögen dies verdeutlichen: die 2010 eingeführten eID-Funktionen des Personalausweises sowie die seit langem in der Entwicklung befindliche Telematikinfrastruktur für das Gesundheitswesen.

---

[1] M. Rost und K. Bock: Privacy by Design und die neuen Schutzziele, in: DuD 35, 1 (2011), S. 30–35.
[2] z.B. Building Security In Maturity Model (BSIMM), https://www.bsimm.com/;
Software Assurance Maturity Model (OpenSAMM), http://www.opensamm.org/;
Microsoft Security Development Lifecycle, https://www.microsoft.com/en-us/sdl.

Die eID-Funktionen des Personalausweises stellen ein Verfahren zur Nutzeridentifikation im Internet zur Verfügung, bei dessen Entwicklung der Datenschutz von Anfang an berücksichtigt wurde[3]. Der Ausweis lässt seinem Inhaber die Kontrolle über die Datenübermittlung, er unterstützt Funktionen wie die Altersverifikation sowie die pseudonyme Nutzung und seine kryptografischen Protokolle verhindern die unerlaubte Identifikation und Verknüpfung von Nutzungsvorgängen. Dieser aus Sicht des Datenschutzes nahezu perfekte Baustein hat nur einen Fehler: Er kann sich gegen andere Mechanismen nicht durchsetzen und wird kaum unterstützt und verwendet.

Während die eID-Funktion wenigstens einsatzbereit ist, kämpft die Gesundheitstelematik seit anderthalb Dekaden darum, diesen Zustand überhaupt zu erreichen. Die Ursachen dafür sind in einem Großprojekt mit verschiedensten Stakeholdern naturgemäß vielfältig. Auffällig ist jedoch der Versuch, alle Sicherheits- und Datenschutzfragen bis ins Detail im Vorhinein zu regeln und eine komplexe Infrastruktur um Kryptografie und Smartcards als Sicherheitsmechanismen herum zu konstruieren.

Ein Symptom dafür sind vielhundertseitige Dokumente wie das Sicherheitskonzept[4] mit sehr detaillierten Festlegungen. Daraus spricht der Versuch, von Anfang an alles richtig und auf gar keinen Fall irgendeinen Fehler zu machen, sind doch Gesundheitsdaten unbestritten sensibel und schützenswert. Jedoch gelten in der Softwareentwicklung sowohl die übertriebene Dokumentation im Vorfeld der Implementierung („Softwarebürokratie“) als auch die Arbeit nach dem Wasserfallmodell („Big Design Up Front“) als möglichst zu vermeidende Antipatterns.

Unabhängig davon, welche Ursachen der schleppende Projektverlauf in diesem speziellen Fall hat, ist zu konstatieren, dass die einst sorgfältig ausgearbeiteten Konzepte inzwischen sichtbar veralten. Zu ihrer Entstehungszeit war von Technologien wie Cloud Computing, Internet of Things, Smartphones und künstlicher Intelligenz so wenig die Rede wie von Cyberwaffen oder modernen Sicherheitskonzepten wie Googles BeyondCorp[5]. Während Supermarktkassen heute das kontaktlose Zahlen mit dem Smartphone erlauben, kämpft das Gesundheitswesen mit einem Virtual Private Network (VPN) und der Basisanwendung Stammdatenmanagement. Ungewiss bleibt darüber hinaus, ob die alten Konzepte heute überhaupt noch angemessene Sicherheit bieten.

Gestaltungsvorgänge

Die Eigenschaften fertiger Produkte und Artefakte lassen sich verhältnismäßig leicht analysieren, bewerten und kritisieren. Den Luxus der Rückschau genießen Entwickler nicht, die ein Produkt erst formen und gestalten. Sie müssen sich nach und nach an die genaue

---

[3] A. Poller, U. Waldmann, S. Vowé und S. Türpe: Electronic Identity Cards for User Authentication – Promise and Practice, IEEE Security & Privacy Mag., 10, 1 (2012), S. 46–54.
[4] gematik: Einführung der Gesundheitskarte – Übergreifendes Sicherheitskonzept der Telematikinfrastruktur, Version 2.2.0, 10.3.2008, https://fachportal.gematik.de/spezifikationen/basis-rollout/datenschutz-und-datensicherheit/uebergreifendes-sicherheitskonzept-der-telematikinfrastruktur/.
[5] Google: BeyondCorp – A new approach to enterprise security, https://cloud.google.com/beyondcorp/.

Problemdefinition und eine brauchbare Lösung herantasten. Dazu gehört die Erkundung es Entwurfsraums ebenso wie Entscheidungen und Abwägungen. Das Ergebnis schließlich wird meist ein von verschiedenen Zielen geprägter Kompromiss sein.

Vertrackte Probleme

Die Entwicklung von Produkten, insbesondere von Software und IT-Systemen, entpuppt sich häufig als ein schwer zu fassendes Problem, das sich einfachen linearen Lösungsstrategien entzieht („Wicked Problem“[6]). Kennzeichnend dafür sind unter anderem folgende Eigenschaften[7]: Ein „Wicked Problem“ lässt sich a priori nicht klar und vollständig formulieren. Es hat keine richtigen oder falschen Lösungen, sondern einen großen Raum unterschiedlich guter Lösungsmöglichkeiten. Die Tauglichkeit von Lösungen lässt sich nicht schnell und einfach testen. Lösungsstrategien für „Wicked Problems“ lassen sich auch nicht durch Versuch und Irrtum entwickeln, da jedes Problem einzigartig ist und nicht einfach eine weitere Instanz eines bekannten Problemtyps. Designaufgaben wie die Software- und Systementwicklung fallen häufig in die Klasse der „Wicked Problems“[8], besonders wenn sie auf innovative Produkte zielen und nicht nur auf Variationen des Bekannten.

Exploration und Entscheidung

Dementsprechend zeigt sich in der Entwicklungsarbeit ein Wechselspiel von Exploration und Entscheidung. Ausgehend von einer anfangs oft unscharfen Problem- oder Zieldefinition beginnt zunächst eine Erkundung des Problem- und Entwurfsraums. Dabei werden verschiedene Lösungskandidaten skizziert, bewertet und verglichen, womit eine Verfeinerung der Ziele und eine Vertiefung des Problemverständnisses, der festen Randbedingungen und der Lösungsmöglichkeiten einhergeht. Durch Auswahl werden die wesentlichen Entwurfsentscheidungen getroffen und so nach und nach das Ergebnis geformt. Dieser generische Prozess zeigt sich in verschiedenen Designdisziplinen von der Architektur[9] bis zur Softwareentwicklung[10].

Die Ergebnisse hängen neben der Kreativität im Finden von Lösungskandidaten wesentlich von deren Bewertung und Auswahl ab. Hier schlagen sich abstrakte Entwurfsziele und Prioritäten in Entscheidungen nieder. Neben den eigentlichen Bewertungskriterien ist auch die Qualität und Gründlichkeit der Bewertung bedeutsam. So ist beispielsweise in Fragen der

---

[6] H. W. J. Rittel und M. M. Webber: Dilemmas in a General Theory of Planning, in: Policy Sci., 4, 2 (1973), S. 155–169;
P. DeGrace und L. H. Stahl: Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms, Yourdon Press, 1990;
M. Poppendieck: Wicked Projects, Softw. Dev. Mag., 10, 5 (2002), S. 72–76;
J. Sutherland: Agile Development: Lessons Learned from the First Scrum, in: Cutter Agile Project Management Advisory Service: Executive Update, 5, 20 (2004), S. 1-4.
[7] Rittel und Webber (Fn 6)
[8] DeGrace und Stahl (Fn 6), Poppendieck (Fn 6)
[9] E. Do und M. D. Gross: Thinking with Diagrams in Architectural Design, in: Thinking with Diagrams, A. F. Blackwell (Ed.), Springer Netherlands, 2001, S. 135–149.
[10] S. Berkun: Making Things Happen: Mastering Project Management, O’Reilly, 2008.

Benutzerinteraktion Feedback aus der Beobachtung realer Benutzern meist wertvoller als eigene Einschätzungen der Entwickler.

Demgegenüber erweist sich die Vorstellung von detaillierten Anforderungen, die a priori vorliegen und nach und nach zu einem Systementwurf verfeinert werden, als trügerisch. Klare Anforderungen als im Sinne des Wasserfallmodells dem Entwurf vorausgehende detaillierte Problemspezifikation entpuppen sich in der Praxis oft als Illusion[11]. Auch sind Statements von Nutzern und anderen Stakeholdern über ihre Bedürfnisse mit Vorsicht zu genießen da beispielsweise eine vorgeschlagene Funktion zwar auf ein Problem oder Interesse hindeutet, der Vorschlag aber nicht unbedingt die beste Lösung dafür ist.

Gestaltungsdimensionen, Prioritäten und Abwägungen

Die Gestaltung eines Gegenstands erfolgt parallel in vielen Dimensionen, zwischen denen es häufig zu Wechselwirkungen kommt. Manche Dimensionen lassen sich direkt formen, wie das Material und der Aufbau eines physischen Gegenstands. Andere Dimensionen, wie zum Beispiel die Haltbarkeit, die Gebrauchstauglichkeit oder die Herstellungskosten, werden nur indirekt von den eigentlichen Gestaltungsentscheidungen beeinflusst, sind aber für die Tauglichkeit eines Entwurfs wichtig und oft Gegenstand von Entwurfszielen.

Aufgrund von Abhängigkeiten, Wechselwirkungen und indirekten Einflüssen erstrecken sich die Auswirkungen einer Entscheidung meist über mehrere Gestaltungsdimensionen. Treten dabei Konflikte auf, können diese entweder durch Kompromisse oder durch die Priorisierung der Entwurfsziele gelöst werden. Extremfälle der Priorisierung einzelner Ziele sind sogenannte Designerprodukte wie die Saftpresse „Juicy Salif“ von Philippe Starck, die aufgrund einseitiger Fokussierung auf die Ästhetik unter Vernachlässigung der Funktionalität eher eine Skulptur als einen Gebrauchsgegenstand darstellt[12]. Besser für den Einsatz in der Küche eignen sich Produkte, die auf Gebrauchstauglichkeit optimiert sind. Dies bedeutet jedoch nicht, dass die Fokussierung auf den dekorativen Wert falsch wäre – sie führt lediglich zu einem anderen Produkt, das sich für andere Zwecke eignet und andere Käufer anspricht.

Weniger bedeutsame Gestaltungsaspekte können auch dem Zufall beziehungsweise den Nebenwirkungen wichtiger Entscheidungen überlassen bleiben, solange sie innerhalb großzügiger Toleranzbereiche bleiben. So wird man, um beim Beispiel der Saftpressen zu bleiben, ohne Not keine Anstrengungen unternehmen, etwa deren Feuerbeständigkeit zu optimieren. Mithin hat das fertige Produkt gestaltbare Eigenschaften, die jedoch bei der Gestaltung nicht besonders berücksichtigt wurden.

---

[11] P. Ralph: The Illusion of Requirements in Software Development, in: Requirements Engineering, 18, 3 (2013), S. 293–296.

[12] B. Russo und A. De Moraes: The Usability of Iconic Designs: A Case Study of Juicy Salif, in: Proc. Hum. Factors Ergon. Soc. Annu. Meet., 47, 5 (2003), S. 844–847;
G. Muratovski: Research for Designers: A Guide to Methods and Practice, Sage Publications, 2015.

Feedback und empirische Bewertung

Voraussetzung zielgerichteter Entwurfsentscheidungen ist die Bewertung der Alternativen. Manche Aspekte können die Gestalter selbst gut einschätzen oder sie müssen es sogar, etwa die Ästhetik bei den bereits angeführten Designerprodukten oder technische Fragen bei der Entwicklung von IT-Systemen. Andere Dimensionen erlauben selbst Experten nur spekulative Bewertungen. So sind die Benutzerinteraktion und Nutzerakzeptanz immer für Überraschungen gut, sobald ein System auf reale Benutzer trifft.

Um nicht zu lange ins Blaue hinein zu entwickeln, versucht man frühzeitig mit vertretbarem Aufwand empirische Erkenntnisse zu gewinnen. In der nutzerorientierten Gestaltung (User-centered Design) nutzt man Methoden wie Beobachtungen und Befragungen, die Entwicklung prototypischer Benutzersteckbriefe (User Personas) sowie frühe Benutzertests mit Mock-ups und Papierprototypen. Die Vorgehensweise des Design Thinking als aktuelle Fortschreibung dieser Ideen geht sogar noch weiter und wendet den Ansatz der schnellen, leichtgewichtigen Empirie auch auf die Suche nach den überhaupt zu lösenden Problemen und erfolgversprechenden Projektzielen aus.

Entwicklungspraktiken in der IT

IT-Systeme sind komplexe Artefakte, die in aller Regel in Teamarbeit entwickelt werden. Um Probleme wie eine geringe Akzeptanz des fertigen Produkts (Beispiel: eID im Personalausweis) oder das Festhalten an vorläufigen Anforderungen und veraltenden Technologien über einen zu langen Zeitraum (Beispiel: Gesundheitskarte) zu vermeiden, bedient man sich heute überwiegend agiler Arbeitsweisen. Die agile Entwicklung orientiert sich an der Charakteristik von Gestaltungsprozessen und erlaubt die Reaktion auf Veränderungen.

Agile Methoden

IT-Entwicklungsprojekte scheitern leicht, das heißt sie enden ohne befriedigendes Ergebnis. Die Ursachen liegen teils in der Natur der „Wicked Problems“, teils in der Schwierigkeit, Gestaltungsprozesse zu organisieren und zu steuern. Aufgrund ihrer dynamischen, erst im Projektverlauf nach und nach konkretisierten und verfeinerten Zielvorstellung entzieht sich die Entwicklungsarbeit der detaillierten Vorausplanung, etwa nach dem Wasserfallmodell mit seinen aufeinanderfolgenden Phasen Anforderungsanalyse, Entwurf, Implementierung, Test und Wartung. Das gegenteilige Extrem, die Gestaltung nur der ungesteuerten Kreativität der Entwickler zu überlassen, ist mangels Koordination und Priorisierung ebenso riskant[13].

Als praktikabler Ansatz zwischen diesen Extremen entstanden in den 1990er Jahren agile Vorgehensweisen, die dann nach der Veröffentlichung des *Manifests für Agile*

[13] B. W. Boehm: Software Risk Management: Principles and Practices, in: IEEE Software, 8, 1 (19919, S. 32–41.

Softwareentwicklung[14] im Jahr 2001 schnell populär wurden. Zusammen mit dem Manifest erschienen um die Jahrtausendwende die ersten Beschreibungen agiler Methoden, etwa *Extreme Programming*[15] *(XP)* und *Scrum*[16]. Um diese Kristallisationspunkte herum entwickelte sich in der Folge ein reichhaltiges Ökosystem agiler Praktiken.

Das Agile Manifest postuliert vier Grundwerte und zwölf Prinzipien. Die agile Entwicklung konzentriert sich danach auf die Arbeit an der Software selbst statt an begleitenden Dokumenten, sie setzt auf die enge und kontinuierliche Zusammenarbeit qualifizierter Entwickler untereinander sowie mit den Abnehmern ihres Produkts statt auf detaillierte Vorgaben und Prozesse, und sie nimmt eine positive Haltung zu Veränderungen ein, etwa in den Anforderungen an ein System.

Aus diesen Werten ergeben sich typische Merkmale agiler Arbeitsweisen:

- Arbeitsorganisation in kurzen Iterationen, die jeweils ein überschaubares Inkrement der Software produzieren,
- der Verzicht auf spekulative Entwicklungsarbeit, der keine konkreten aktuellen Anforderungen zugrunde liegen, zugunsten derjenigen Arbeiten, die den höchsten Kunden- oder geschäftlichen Wert versprechen,
- die empirische Steuerung der Entwicklungsarbeit anhand von Selbstbeobachtung und regelmäßigem Kundenfeedback sowie damit verbunden
- die weitgehende Selbstorganisation der Entwicklerteams und der Verzicht auf Softwarebürokratie.

Zu diesen Merkmalen kommen spezifische Entwicklungspraktiken, die das agile Arbeiten unterstützen[17]. Nicht alle dieser Praktiken finden universelle Zustimmung[18], dennoch bilden sie den gegenwärtigen Stand der Technik für weite Bereiche der Softwareentwicklung.

Beispiel: Scrum

Das heute wohl verbreitetste agile Vorgehensmodell ist *Scrum*[19]. Dabei handelt es sich um ein Management-Framework zur Organisation der Entwicklungsarbeit. Scrum definiert die Rollen mehrerer Akteure – Scrum Master, Product Owner und Entwicklerteam – sowie deren Zusammenarbeit in einem iterativen Prozess.

Als Prozess betrachtet erfasst Scrum alle Anforderungen und Ideen im sogenannten *Product Backlog*. Die eigentliche Entwicklungsarbeit erfolgt in Zyklen von einigen Wochen Dauer, den *Sprints*. Zu Beginn jedes Sprints werden in einem Meeting, dem *Sprint Planning*, Elemente aus dem Product Backlog ausgewählt und in das *Sprint Backlog* übernommen. Das Ziel des Sprints ist, die ausgewählten Anforderungen bis zum Ende des Sprints vollständig umzusetzen, das

---

[14] K. Beck et al.: Manifesto for Agile Software Development, 2001, http://agilemanifesto.org/.
[15] K. Beck: Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000.
[16] K. Schwaber und M. Beedle: Agile Software Development with Scrum, Prentice Hall, 2002;
K. Schwaber und J. Sutherland: The Scrum Guide, 2017, https://www.scrumguides.org/.
[17] L. Williams: What Agile Teams Think of Agile Principles, in: Commun. ACM, 55, 4 (2012), S. 71–76.
[18] B. Meyer: Making Sense of Agile Methods, in: IEEE Software, 35, 2 (2018), S. 91–94.
[19] Schwaber und Beedle (Fn 16), Schwaber und Sutherland (Fn 16)

heißt zu implementieren, zu testen, zu dokumentieren und so weiter. Am Ende des Sprints werden die Ergebnisse in einem weiteren Meeting, dem *Sprint Review*, demonstriert und anhand einer Checkliste, der *Definition of „Done“*, abgenommen.

Getragen wird dieser Prozess von den definierten Rollen der Mitwirkenden und dem Wechselspiel ihrer Aufgaben. Die Steuerung der Arbeit erfolgt zwischen dem Product Owner und dem Entwicklerteam. Der Product Owner ist für die Priorisierung des Product Backlogs und für die Abnahme der Sprint-Ergebnisse verantwortlich. Damit steuert er – ggf. unter Beteiligung weiterer Stakeholder – die Richtung der Entwicklung sowie die Qualität des Produkts. Das funktionsübergreifende Entwicklerteam organisiert seine Arbeit im Rahmen seiner Rolle selbst und ist für die eigentliche Entwicklungsarbeit verantwortlich. Der Scrum Master unterstützt und moderiert die Zusammenarbeit dieser beiden Parteien.

Mit dieser Rollenverteilung sorgt Scrum dafür, dass die Entwicklungsarbeit einerseits gesteuert, koordiniert und an den Bedürfnissen der Kunden orientiert wird, den Entwicklern andererseits jedoch die erforderlichen Freiräume für ihre Gestaltungsarbeit bleiben. Gleichzeitig ermöglichen kurze Iterationen die schnelle Anpassung an sich verändernde Anforderungen und neue Erkenntnisse.

Kontinuierliche Arbeit im laufenden Betrieb

Parallel zum Siegeszug der agilen Methoden wanderten große Teile der Anwendungen vom Endgerät in die Cloud. An die Stelle langer, oft mehrjähriger Entwicklungs- und Release-Zyklen trat dabei die Möglichkeit, direkt am Produktivsystem zu arbeiten. Damit verbunden sind Weiterentwicklungen der agilen Entwicklungsmethoden wie DevOps, Continuous Delivery und Testen im Produktivbetrieb.

Entwicklung und Systembetrieb rücken näher zusammen und verschmelzen schließlich zu einer Einheit („DevOps“). Dabei werden nicht nur organisatorische Grenzen zwischen Entwicklung und Systemadministration beseitigt, sondern auch Arbeitsmethoden angeglichen. So übernimmt DevOps die agile Idee der konsequenten Automatisierung von Routineaufgaben und überträgt sie in das Infrastruktur- und Anwendungsmanagement.

Infolgedessen entsteht eine hochautomatisierte Prozesskette vom Entwickler bis ins laufende System, über die Änderungen der Software jederzeit kurzfristig in den Produktivbetrieb übernommen werden können. Umgekehrt gelangt Feedback aus dem Betrieb, etwa Fehlerberichte und Protokolle, ebenso leicht zurück zu den Entwicklern. An die Stelle kurzer Iterationen kann damit die kontinuierliche Weiterentwicklung des laufenden Dienstes treten.

Daraus resultiert die Möglichkeit, Änderungen schnell und unkompliziert im Produktivbetrieb zu testen. So erhält etwa beim A/B-Testing ein Teil der Nutzer ein neues Feature oder eine modifizierte Variante. Neben der allgemeinen Beobachtung und Protokollierung zum Finden und Analysieren von Problemen werden dabei insbesondere die Wirkungen im Hinblick auf angestrebte Ziele gemessen und verglichen; Ziele könnte zum Beispiel eine Erhöhung der

Klickrate, eine verringerte Zahl von Abbrüchen eines Vorgangs oder eine längere Verweildauer der Nutzer sein. Im jeweiligen Sinne erfolgreiche Änderungen werden dann nach und nach auf alle Nutzer ausgedehnt, erfolglose oder fehlerhafte hingegen zurückgezogen oder überarbeitet.

Experimente mit Nutzern werfen selbst Fragen des Datenschutzes und der Ethik[20] auf. So ernteten Facebook und die Cornell University vor einigen Jahren heftige Kritik angesichts eines Experiments[21], das sich auf Eingriffe in die Newsfeeds nichtsahnender Facebook-Nutzer stützte. Technisch unterscheidet sich dieses Experiment freilich kaum von verbreitet angewandten Praktiken wie dem A/B-Testing.

Komponenten und Plattformen

Software entsteht selten von Grund auf neu. Neuentwicklungen stützen sich auf eine breite Palette an Plattformen, Komponenten und Entwurfsmustern, auf die Entwickler zurückgreifen können. Mit der Wahl einer Plattform – zum Beispiel eines mobilen Betriebssystems wie Android oder iOS als Plattform für eine App – sind implizite Entscheidungen sowie Vorgaben und Empfehlungen an die Entwickler verbunden. Daraus resultieren einerseits Einschränkungen des Entwurfsraums, andererseits jedoch auch große Arbeitserleichterungen.

Als Folge der Verwendung vorhandener Plattformen und Komponenten sowie bekannter Entwurfsmuster wird ein Teil der Eigenschaften eines IT-Systems außerhalb von dessen eigentlicher Entwicklung geprägt und folgt dem Stand der Technik zur Entwicklungszeit. Zwar sind Abweichungen vom Üblichen grundsätzlich möglich, doch müssen sie nicht nur technisch, sondern auch ökonomisch praktikabel sein, um sich gegenüber den Alternativen durchzusetzen.

Datenschutz als Gestaltungsziel

Datenschutz gehört zu den nichtfunktionalen Anforderungen[22], auch wenn er möglicherweise funktionale Konsequenzen hat. Gesetzliche Vorgaben wirken als Einschränkungen (Constraints), die den Entwurfsraum beschneiden. In der Forderung nach Datenschutz durch Technikgestaltung steckt damit zunächst die Bekräftigung, dass die Vorschriften des Datenschutzes auch praktisch und nicht nur formal auf dem Papier umzusetzen sind. In einer weiter reichenden Interpretation kann sie auch so verstanden werden, dass die Ziele des Datenschutzes in der Systemgestaltung im praktikablen und zumutbaren Rahmen als Qualitätsattribute zu verfolgen seien, ohne dass es im Einzelnen ausdrücklicher Vorgaben bedürfe. Gemeinsam sind beiden Perspektiven Wechselwirkungen mit anderen Entwurfsaspekten sowie der Umstand, dass die Berücksichtigung einer weiteren Anforderungsklasse mit zusätzlichem Aufwand verbunden ist.

---

[20] E. W. Felten: Privacy and A/B Experiments, in: J. Telecomm. High Tech. L., 13 (2015), S. 193.
[21] A. D. I. Kramer, J. E. Guillory und J. T. Hancock: Experimental Evidence of Massive-Scale Emotional Contagion Through Social Networks., in: Proc. Natl. Acad. Sci. U. S. A., 111, 24 (2014), S. 8788–8790.
[22] M. Glinz: On Non-Functional Requirements, in: Proc. 15th IEEE Int. Requirements Engineering Conf., 2007, S. 21–26.

Von Anfang an einbauen?

Bereits die bloße Umsetzung konkreter gesetzlicher Vorgaben erhöht die Komplexität der Entwicklungsarbeit und stellt Entwickler vor nichttriviale Entscheidungen. Dies sei hier am Beispiel der Rechtmäßigkeit der Datenverarbeitung durch einen Webdienst illustriert. Vorgaben dazu macht Kapitel II der DS-GVO. Danach kann die Verarbeitung aufgrund einer Einwilligung oder eines anderen Erlaubnistatbestands, zum Beispiel einer Notwendigkeit zur Diensterbringung, erfolgen. Einwilligungen müssen freiwillig sein; sie können jederzeit widerrufen werden und der Widerruf muss so einfach wie die Einwilligung sein. Für Kinder und Jugendliche ist die Einwilligung der Sorgeberechtigten einzuholen.

Bereits die Unterscheidung zwischen personenbezogenen und anderen Daten sowie erst recht unterschiedliche Verarbeitungszwecke und Erlaubnisgrundlagen erweitern das Datenmodell des Dienstes um eine aus technischer Sicht willkürliche Dimension, die in der Entwicklung zu beachten ist. Relativ einfach umzusetzen sind dabei globale Regeln (z.B. „Alle Nutzerstammdaten mit Ausnahme der pseudonymen Benutzerkennung sind optional und lassen sich jederzeit ändern oder löschen.“) sowie Festlegungen entlang technischer und architektureller Kriterien (z.B. „Protokolldateien werden unabhängig vom Inhalt nach sieben Tagen automatisch gelöscht.“). Komplizierter wird die Umsetzung, wenn explizite Fallunterscheidungen notwendig sind (z.B. „Die Angabe der Gewerkschaftszugehörigkeit darf nur mit Einwilligung der/des Betroffenen protokolliert werden.“).

Dabei implizieren verschiedene Funktionen und Teilsysteme jeweils eigene Sichten auf verarbeitete Daten, die häufig an technischen statt an Datenschutzkategorien ausgerichtet sind. So kann zum Beispiel eine Backup-Funktion ohne Schwierigkeiten zwischen Dateien oder Datenbanktabellen unterscheiden, jedoch kaum zwischen einzelnen Datensätzen oder gar –feldern; Protokolle werden als Dateien („Protokoll vom 25. Mai 2018“) und in technischen Kategorien („wichtige sicherheitsrelevante Ereignisse“) behandelt, aber nicht nach Betroffenen oder Datenarten organisiert.

Damit prägt letztlich die verwendete Technik den praktikablen Umgang mit Daten, statt sich Vorgaben anzupassen. Anders als etwa im Wasserfall-Modell der Gesundheitstelematik vorausgesetzt, richtet sich die Entwicklung und Gestaltung nicht einfach nach Vorgaben. Vielmehr kommt es zu Wechselwirkungen – Entwurfsentscheidungen müssen nicht nur Datenschutzvorgaben beachten, sondern können ihrerseits zum Beispiel eine Notwendigkeit begründen. Das Datenschutzkonzept eines Systems muss folglich im Zuge seiner Entwicklung mitwachsen.

Erfolgt ein Teil der Datenverarbeitung aufgrund von Einwilligungen der Betroffenen, so stellt sich darüber hinaus Frage, wie diese Einwilligungen eingeholt und dokumentiert werden. Eine Herausforderung ist dabei die Gestaltung der Benutzerinteraktion. So ist es einerseits gewiss legitim, dezent um eine Einwilligung zu bitten; andererseits können aufdringliche Bitten, etwa durch Inhalte verdeckende Boxen, die Freiwilligkeit in Frage stellen.

Solche Schwierigkeiten sprechen nicht grundsätzlich gegen die Forderung, den Datenschutz in der Entwicklung möglichst frühzeitig und durchgängig zu berücksichtigen. Sie verdeutlichen jedoch, dass damit erstens Arbeitsaufwand verbunden ist und dass es zweitens nicht damit getan ist, einmal eine Richtlinie aufzustellen und sie dann nebenbei umzusetzen.

Komplizierte Charakteristik

Datenschutzanforderungen lassen sich selten a priori als eine Liste benötigter Funktionen oder Maßnahmen formulieren und später abarbeiten. Sie sind vielmehr als zusätzlicher Aspekt mit den übrigen Gestaltungsdimensionen verwoben. Während die abstrakten Ziele und Vorgaben feststehen, unterscheiden sich ihre Konkretisierungen von Fall zu Fall und stehen in Wechselwirkungen mit anderen Zielen und Anforderungen. Die Notwendigkeit, Wirkung und Eignung von Maßnahmen oder Entwurfsentscheidungen hängt vom Kontext ab und lässt sich sinnvoll nur im Rahmen eines spezifischen Systems und Systementwurfs diskutieren.

Querschnittsthema

Datenschutz ist ein Querschnittsthema, das potenziell alle Teile und Aspekte eines Systems betrifft. Folglich lässt sich der Datenschutz nur eingeschränkt zentralisieren und einer Systemkomponente konzentrieren. Für die Entwicklung bedeutet dies, dass Datenschutzfragen immer und überall auftauchen können. Effektiver Datenschutz ist keine Systemfunktion, sondern die Konsequenz vieler Gestaltungsentscheidungen in ihrem Zusammenwirken.

Indirekte Erfüllung

Von Interesse sind letztlich nicht einzelne Maßnahmen oder Merkmale, sondern die damit erzielte Wirkung. Anders als viele funktionale Ziele lassen sich Datenschutz und Sicherheit deshalb nicht einfach ausgehend von Anwendungsfällen (Use Cases) oder User Stories durch Verfeinerung bis zur Implementierung erreichen. Vielmehr handelt es sich zunächst um eine Bewertungsdimension und bei Schwächen und Defiziten muss man nach Lösungen für das jeweilige Problem suchen.

Erhöhte Komplexität

Die Berücksichtigung einer zusätzlichen Gestaltungsdimension erhöht die Komplexität der Entwicklungsarbeit. Sie bringt zusätzliche Anforderungen, Entwurfsaspekte und Beurteilungskriterien mit sich und verlangt Abwägungen und Priorisierungen gegen andere Entwurfsziele. Vielleicht am einfachsten zu handhaben sind die Extreme: die Unterordnung aller anderen Entwurfsziele unter den Datenschutz sowie umgekehrt seine konsequente Vernachlässigung.

Externer Nutzen und Anreizkollision

Die Umsetzung von Datenschutzanforderungen kann im genuinen Interesse der Betreiber und Entwickler eines Systems liegen, etwa wenn sie befürchten, Datenskandale oder mangelndes Vertrauen könnten das Geschäft gefährden. In solchen Fällen besteht ein natürlicher Anreiz,

Maßnahmen zu ergreifen. Häufig haben Datenschutzmaßnahmen jedoch keinen solchen direkten Nutzen oder sie laufen sogar wirtschaftlichen Interessen zuwider. Das heißt nicht, dass der Datenschutz in diesen Fällen nutzlos wäre, aber dieser Nutzen tritt nicht dort ein, wo die Kosten anfallen. Für Betreiber und Entwickler wird der Datenschutz in solchen Fällen zu einem Compliance-Thema – gesucht ist der günstigste Weg, die gesetzlichen Vorgaben ausreichend einzuhalten.

Schwierige Bewertung und Kontrolle

Die Bewertung von Datenschutzeigenschaften ist aufwändig und kaum zu automatisieren. Das macht es schwer, schnelles und kontinuierliches Feedback bereitzustellen, wie es agile Methoden voraussetzen. Wie gut oder schlecht ein System die Ziele des Datenschutzes umsetzt, zeigt sich weder in automatisierten Tests noch in der Benutzerinteraktion, sondern nur in der Bewertung durch Fachleute. Hinzu kommt, dass Datenschutz und Sicherheit als Ziele auf die Reduktion künftiger Risiken gerichtet sind[23]. Die Diskussion solcher Risiken bewegt sich jedoch schnell ins Reich der Spekulation, wo Risiken mangels Datengrundlage beliebig über- oder unterschätzt werden können. Das macht es schwer, solide zu argumentieren, warum der Schutz in einer konkreten Situation ausreicht oder auch nicht.

Erfolgsfaktoren

Die effektive Berücksichtigung von Datenschutzaspekten im Entwicklungsprozess ist möglich, aber kein Selbstläufer. Scheitern kann sie auf drei Ebenen: erstens an einer geringen Gewichtung des Datenschutzes gegenüber anderen Entwurfszielen aufgrund wirtschaftlicher Anreize, zweitens an der Charakteristik des Datenschutzes als Gestaltungsziel und drittens an Schwächen in der Organisation der Entwicklungsarbeit.

Wirtschaftlichkeit

Entwurfsentscheidungen werden immer auch von wirtschaftlichen Erwägungen beeinflusst. Die agile Entwicklung macht dies explizit, indem sie Arbeiten nach dem erwarteten Wert der Ergebnisse priorisiert. Übersteigen die Kosten von Entwicklungsarbeiten den Nutzen oder Gewinn, so lohnen sie sich überhaupt nicht und unterbleiben; Arbeiten mit höherem Gewinn bei gleichem Aufwand werden systematisch bevorzugt. Scheitern kann der Datenschutz als Gestaltungsziel folglich an einem schlechten Kosten-Nutzen-Verhältnis.

Auf der Kostenseite erhöht die Berücksichtigung von Datenschutzaspekten den Entwurfs- und Implementierungsaufwand sowie die Komplexität der Entwicklung. Davon betroffen sind neben der Planung und Umsetzung spezifischer Datenschutzfunktionen auch die übrige Entwicklungsarbeit, soweit dabei Datenschutzfragen beachtet werden müssen.

---

[23] S. Xiao, J. Witschey und E. Murphy-Hill: Social Influences on Secure Development Tool Adoption: Why Security Tools Spread, in: Proc. 17th ACM Conf. on Computer Supported Cooperative Work & Social Computing – CSCW ’14, 2014, S. 1095–1106.

Der erzielte genuine und direkte Nutzen für Entwickler und Betreiber bleibt jedoch möglicherweise gering und kann sogar negativ ausfallen. Zum einen unterstützt der Datenschutz nicht per se andere Entwurfsziele wie Funktionalität, Usability, Performance und so weiter. Zum anderen sind Zielkonflikte immer dann unvermeidlich, wenn besserer Datenschutz zu geringeren Gewinnen oder zu Funktionseinschränkungen führt.

Die besten Aussichten hat der Datenschutz durch Technikgestaltung mithin dort, wo seine Ziele mit denen von Verantwortlichen, Verarbeitern und Entwicklern übereinstimmen oder wenigstens dazu kompatibel bleiben. Dies ist beispielsweise der Fall, wenn ein Geschäftsmodell vom Vertrauen in einen Anbieter abhängt und bei Problemen schnell zusammenbricht. Umgekehrt tendiert er zur bloßen Compliance-Übung, wo dieser echte Anreiz fehlt, und bei Zielkonflikten kann manipulative Gestaltung[24] hinzukommen, die formale Vorgaben oberflächlich einhält, tatsächlich jedoch unterläuft.

Sichtbarkeit, Evaluation, Feedback

Um wirksam zu werden, müssen die abstrakten Ziele des Datenschutzes in der Entwicklung in konkrete Anforderungen, Entscheidungen und Arbeiten umgesetzt werden. Dies geschieht nicht auf magische Weise als Folge einer Awareness-Kampagne, sondern das Thema Datenschutz muss in den maßgeblichen Artefakten und Aktivitäten präsent sein, damit es bearbeitet wird. Im Fall von Scrum bedeutet das beispielsweise: Datenschutzaufgaben müssen im Product Backlog und in der Definition of „Done“ auftauchen und Gegenstand der Planungs- und Review-Meetings sein. Andernfalls kann und wird sich niemand darum kümmern.

Im Weg stehen dem die geringe inhärente Sichtbarkeit des Themas sowie die Schwierigkeit, im Takt der Entwicklung Entscheidungen zu evaluieren und Feedback zu erhalten. Während relevante Funktionsfehler, Usability-Probleme, Leistungsschwächen usw. häufig in Tests oder spätestens im Produktivbetrieb auffallen, weil sie die Benutzung oder den Betrieb merklich beeinträchtigen, ist dies bei Datenschutzmängeln kaum der Fall. Vielmehr muss man ein System explizit auf solche Mängel hin analysieren.

Anders als viele technische Softwaretests lassen sich diese Analysen kaum automatisieren, so dass sie sich insbesondere in die agile Entwicklung schlecht integrieren. Entwicklern fehlt damit das ständige, schnelle Feedback zu ihren Arbeitsergebnissen und Entscheidungen.

Vertretung und Priorisierung

Auch sichtbar gemachte und wirtschaftlich erfüllbare Datenschutzanforderungen können unberücksichtigt bleiben, wenn sie in der Entwicklung systematisch niedrigere Priorität erhalten als andere Ziele und Aufgaben. In Scrum zum Beispiel ist der Product Owner dafür zuständig, das Product Backlog in Abstimmung mit den Stakeholdern zu priorisieren, und zwar nach dem

[24] Einsatz sogenannter Dark Patterns, https://darkpatterns.org/.

erwarteten Kundennutzen der jeweiligen Tätigkeit bzw. ihrer Ergebnisse, und Qualitätskriterien werden im Sprint Review anhand der Definition of „Done“ geprüft.

Damit das Thema tatsächlich Gewicht erhält, muss es nicht nur pro forma in diesen Artefakten vorkommen, sondern auch fortlaufend von maßgeblichen Stakeholdern – z.B. Anwendern oder dem Produktmangement – vertreten werden. Andernfalls tritt es schnell in den Hintergrund gegenüber anderen Anforderungen mit aktiveren Fürsprechern.

Mittel zur Förderung

Datenschutz durch Technikgestaltung zeigt sich zuerst in der Entwicklungsarbeit und erst mittelbar in ihren Resultaten. Versuche, das Pferd von hinten aufzuzäumen und die Diskussion auf Maßnahmen wie Pseudonymisierung oder Verschlüsselung zu fokussieren, haben deshalb nur geringe Erfolgsaussichten. Stattdessen kommt es darauf an, dass das Entwurfsziel Datenschutz in der Entwicklung ausreichendes Gewicht erhält und deshalb in Gestaltungsentscheidungen berücksichtigt wird.

Ein mit Sicherheits- und Datenschutzbedenken begründeter schwerfälliger Wasserfallprozess wie im Fall der Gesundheitstelematik leistet dies nur scheinbar, denn er unterwirft die Gestaltung einem ungeeigneten Prozess. Sinnvoller erscheint die Unterstützung gängiger Vorgehensweisen und damit verbunden die Frage, wie sich der Datenschutz dort fördern lässt. Das bedeutet heute: Datenschutz durch Technikgestaltung muss sich in agile und hyperagile Vorgehensweisen einfügen.

Die Erfolgsfaktoren aus dem vorigen Abschnitt bieten Ansätze zur Unterstützung und Förderung unter der Annahme, dass die ökonomischen Anreize nicht grundsätzlich in die falsche Richtung zeigen. Wo sie dies doch tun, wird man allein durch Unterstützung der Gestaltungsvorgänge nur wenig erreichen. Die Liste der skizzierten Ansätze erhebt keinen Anspruch auf Vollständigkeit, sondern sie deutet nur Richtungen für weitere Arbeiten an. Vorbild ist die IT-Sicherheit, die, abgesehen von etwas anders gelagerten ökonomischen Anreizen, viele Gemeinsamkeiten und Überschneidungen mit dem Datenschutz zeigt.

Organisation und Werkzeuge

Unterstützungsteams

Um ihre Entwickler in puncto IT-Sicherheit zu unterstützen, haben viele Software– und Internetfirmen eigene Sicherheitsteams aufgestellt. Analoge Strukturen oder die thematische Erweiterung der bestehenden können auch den Datenschutz als Entwicklungsziel fördern. Die Aufgaben eines solchen Teams sind vielfältig. Es fungiert als Ansprechpartner, Berater und Stakeholder, gibt Entwicklern Feedback, stellt in Zusammenarbeit mit den Entwicklern Werkzeuge und Hilfsmittel bereit und gibt dem Management eine Sicht auf das Thema. Diese

Aufgaben gehen weit über die formalen Pflichten eines Datenschutzbeauftragten hinaus. Dennoch bleibt die eigentliche Gestaltungsarbeit Aufgabe der jeweiligen Entwicklerteams.

Methoden und Werkzeuge

In der IT-Sicherheit entstand im Lauf der Zeit eine Reihe von Werkzeugen, die Entwickler bei der Analyse und Gestaltung helfen. Dazu gehören zum Beispiel Methoden zur Bedrohungsmodellierung[25]. Diese Methoden nehmen Entwicklern keine Arbeit ab, sondern helfen ihnen beim Erfassen, Analysieren und Diskutieren relevanter Entwurfsmerkmale. Von analogen Werkzeugen könnte auch der Datenschutz profitieren, zumal mit ihrem Einsatz spezifische Aktivitäten im Entwicklungsprozess verbunden sind, die Aufmerksamkeit und Sichtbarkeit schaffen.

Umgebung und Ökosystem

Communities

In der IT-Sicherheit haben sich Communities wie das Open Web Application Security Project[26] (OWASP) gebildet. Das OWASP vernetzt Akteure aus verschiedenen Unternehmen, die mit ähnlichen Sicherheitsfragen konfrontiert sind, da sie dieselben Technologien und Plattformen einsetzen. Diese Community trägt Open-Source-Projekte, die Arbeitsmaterialien, Werkzeuge und Musterlösungen bereitstellen. Gerade das OWASP ist damit zur ersten Anlaufstelle für viele Sicherheitsfragen geworden und seine Arbeit repräsentiert den Stand der Technik. Der Datenschutz durch Technikgestaltung würde von vergleichbaren Strukturen profitieren, deren Wachstum sich jedoch nicht erzwingen lässt.

Lösungsbausteine

Die Kostenseite im Kosten-Nutzen-Verhältnis hängt bei aufwändigen Maßnahmen wesentlich von der Verfügbarkeit einsetzbarer Lösungsbausteine ab. Die Bereitstellung entsprechender Komponenten kann deshalb dazu beitragen, dass Datenschutzbelange wirksam berücksichtigt werden. Zum Beispiel ist die Verschlüsselung von Internet-Verbindungen in vielen Anwendungen praktikabel, da fertige Implementierungen des TLS-Protokolls[27] ebenso zur Verfügung stehen wie die Infrastruktur zur Schlüsselverwaltung.

Allerdings müssen diese Komponenten nicht nur einen guten Datenschutz bieten, sondern auch in jeder anderen Hinsicht für den praktischen Einsatz geeignet sein. Wie leicht daran selbst ausgefeilte Entwürfe scheitern, zeigt das Beispiel der eID-Funktion im Personalausweis, die bis heute kaum praktische Bedeutung hat. Chancenreicher wären Bausteine, deren Gestaltung und Weiterentwicklung von der Community ihrer Anwender getragen wird.

---

[25] A. Shostack: Threat Modeling: Designing for Security, Wiley, 2014.
26 https://owasp.org/
[27] Transport Layer Security, auch noch unter dem früheren Namen Secure Socket Layer (SSL) bekannt.

Fazit

So nachvollziehbar der Wunsch nach Datenschutz durch Technikgestaltung ist, so schwer lässt er sich konkretisieren und durch Einflussnahme auf Entwicklungsprozesse fördern. Gestaltungsprozesse suchen im Entwurfsraum nach Lösungen und wägen dabei Vor- und Nachteile in verschiedenen Dimensionen ab. Die Gestaltung verfolgt Ziele, aber sie strebt nicht linear einem vorbestimmten Endzustand entgegen. Eine klare Vorstellung vom Produkt entsteht vielmehr erst im Gestaltungsprozess.

Detaillierte Vorgaben für Produkteigenschaften, etwa in Form von Maßnahmenkatalogen, setzen wesentliche Entwurfsentscheidungen bereits voraus. Außerhalb eng umgrenzter Bereiche mit geringer Innovationsdynamik eignen sie sich deshalb schlecht zur Steuerung, sondern hinken der technischen Entwicklung hinterher. Weniger starr und deshalb passender sind Ansätze, die Entwicklungs- und Gestaltungsvorgänge als probabilistische Prozesse betrachten und mit dem Ziel beeinflussen, die Wahrscheinlichkeit guter, d.h. im Sinne des Datenschutzes wünschenswerter Ergebnisse zu erhöhen.

Die ideale Grundlage für Datenschutz durch Technikgestaltung bildet eine echte Nachfrage nach Datenschutz. Sie gibt dem Datenschutz einen ökonomischen Wert, der sich in Gestaltungszielen niederschlägt. Diese Situation tritt allerdings nur selten ein. Das liegt zum einen an der Schwierigkeit der Risikobewertung – welchen Nutzen Maßnahmen haben oder welche tatsächlichen Risiken ihr Fehlen mit sich bringt, lässt sich schwer quantifizieren und die Folgen ungenügenden Datenschutzes gehören nur selten zur Alltagserfahrung. Zum anderen gerät der Datenschutz leicht in einen Zielkonflikt mit den möglichen Gewinnen aus der Datenverarbeitung.