Transparency by Design

Softwaredesign für einen transparenten Umgang mit Daten am Arbeitsplatz

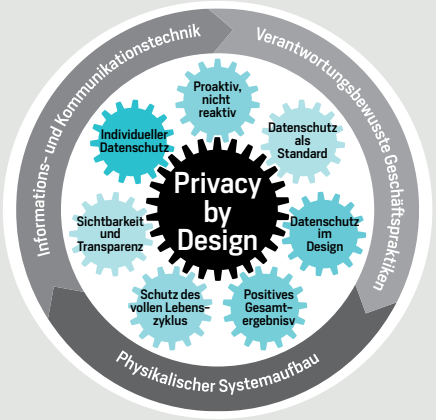
ALEXANDER PRETSCHNER,
VALENTIN ZIEGLMEIER,
LEHRSTUHL FÜR SOFTWARE UND SYSTEMS ENGINEERING AN DER TU MÜNCHEN

Eine immer transparentere Arbeitswelt erfordert, dass sich Softwaredesign weiterentwickelt, um sinnvolle Anwendungsfälle für Daten zu ermöglichen und zugleich Datenmissbrauch vorzubeugen. Wir beschreiben deshalb Transparency by Design als möglichen Weg für Software am Arbeitsplatz. Dabei blicken wir auf die fundamentalen Veränderungen, die diesen neuen Ansatz notwendig machen, diskutieren mögliche Implikationen für Softwaredesign und erläutern unsere Umsetzung des Konzepts im Rahmen des Forschungsprojekts.

1_ Software im Kontext neuer Transparenz

Daten und die Software, die sie verarbeitet, durchdringen heute alles. Mit der fortschreitenden Digitalisierung wird insbesondere die Arbeitswelt immer transparenter. Diese Realität kreiert zwei zentrale Themen für die Informatik: Auf der einen Seite ergeben sich immense Möglichkeiten, auf Basis von Daten effektivere Software zu gestalten. Prädiktive Analysen, maschinelles Lernen und Big Data ermöglichen komplett neue Anwendungsfälle für Daten. Auf der anderen Seite trägt unsere Disziplin eine klare Verantwortung. Datentransparenz steht erst einmal im Gegensatz zu Datenschutz, welcher einen hohen Stellenwert einnimmt.

Schon im Jahr 1995 schlug Ann Cavoukian, die ehemalige Information and Privacy Commissioner der kanadischen Provinz Ontario, deshalb den Ansatz „Privacy by Design" vor, welcher im Jahr 2009 in einer Veröffentlichung von sieben fundamentalen Prinzipien konkretisiert wurde (siehe Cavoukian[2]). Im Gegensatz zur üblichen Vorgehensweise, nach Fertigstellung eines Softwareprojekts den Datenschutz zu prüfen und gegebenenfalls zu überarbeiten, sieht Privacy by Design vor, Datenschutzgrundsätze von vornherein in die Konzeptionierung und das Design von Software zu integrieren. Dem liegt die Annahme zugrunde, dass sich Softwaredesign grundlegend und inkompatibel verändert, wenn Datenschutz



Privacy by Design (nach Ann Cavoukian)

bewusst einbezogen wird; eine Anpassung, die im Nachhinein nur sehr umständlich oder gar nicht mehr möglich ist.

Cavoukians Sorgen betrafen dabei die legitime Nutzung von Daten innerhalb kontrollierter Systeme. Wie sollte jedoch mit illegitimer Nutzung umgegangen werden? Dazu entwickelte sich zur selben Zeit die Idee der „Information Accountability", die Daniel Weitzner und Koautoren im Jahr 2007 vorstellten (siehe Weitzner et al.[7]). Ihre informatische Sichtweise bezieht neben der bewusst angedachten Nutzung von Daten deren Missbrauchspotenzial ein. Schon wenn Daten nicht ausreichend gesichert sind, besteht Missbrauchsgefahr, vor allem aber war schon damals bekannt, dass automatisiert abgeleitete Daten und Zusammenführung über mehrere Quellen Risiken bergen. Zugleich betonen die Autoren aber auch ein weiteres Problem des damaligen Status quo: Zu rigide Schutzmaßnahmen schränken legitime Nutzung ein und ersticken Innovation im Keim. Transparenz, als Kernbestandteil von Accountability, kann dann eine eher ermöglichende als problematische Rolle im Datenschutz einnehmen, wenn die Verwendung von Daten beobachtet werden kann und illegitime Nutzung sanktioniert wird.

Gerade am Arbeitsplatz kann die illegitime Nutzung von Daten besondere Schlagkraft gewinnen. Zum Fehlen der Transparenz der Systeme und zur Überforderung der Nutzer kommen die erzwungene Nutzung von Software ohne Wahlmöglichkeit sowie die natürliche Machtasymmetrie zwischen Führungskraft und Mitarbeitenden als Brandbeschleuniger hinzu. Es reicht hier also nicht mehr, die Entscheidung für oder gegen eine Software als probates Gegenmittel für Datenschutzsorgen zuzulassen. Zugleich gewinnen Datenschutzsorgen durch die Gefahr von beruflichen Nachteilen oder sogar Jobverlust an Gewicht.

Die im Jahr 2018 in Kraft getretene Datenschutz-Grundverordnung (DSGVO) verankerte wegen solcher Risiken Werte und Datenschutzprinzipien in der Gesetzgebung und definierte eindeutige Rechte für Individuen in Bezug auf ihre persönlichen Daten. Das Prinzip Privacy by Design findet sich hier zum Beispiel in Artikel 25 („Datenschutz durch Technikgestaltung und durch datenschutz-freundliche Voreinstellungen") wieder. Dabei war ein wichtiges Ziel, die Datensouveränität zu stärken und den Betroffenen Kontrolle über die Verarbeitung ihrer Daten zu geben. Während dieser zusätzliche Schutz wichtig war und die Situation deutlich verbessert hat, reicht er doch nicht aus, um Missbrauch von Daten zu vermeiden. Zugleich kann die präventive Natur des Gesetzes legitime und sinnvolle Datennutzung einschränken. Verarbeitungsprozesse persönlicher Daten können nun die vorherige Einwilligung von Individuen erfordern, wenn keine anderen Rechtfertigungsgründe bestehen. In diesem Fall müssen Individuen zum Zeitpunkt der Einwilligung verstehen, was ihre Datenfreigabe bewirken wird. Das kann bereits bei legitimer Datennutzung schwer sein, vor allem wenn technisches Wissen über die Systeme fehlt, aber eine unvorhergesehene Nutzung oder ein Missbrauch von Daten sind nicht kontrollierbar. Das erzeugt Sorgen (siehe Teebken & Hess[5]), die ein risikovermeidendes Verhalten begünstigen. Wo es für Nutzerinnen und Nutzer also möglich ist, werden Datenfreigaben eingeschränkt und Dateneingaben minimiert. Auf der anderen Seite führt erzwungene oder betrieblich notwendige Transparenz aufgrund dieser Sorgen möglicherweise zu Reaktanz (siehe Feng et al.[3]), Unzufriedenheit mit dem Arbeitgeber oder psychischer Belastung (siehe Bhave et al.[1]).

Wir sehen das Problem in der fehlenden Einbeziehung der betroffenen Personen im Softwaredesign. Statt sich, wie bisher, nur auf die Nutzer der Software zu fokussieren, sollte Software bereits mit dem Gedanken an die Transparenzinteressen von Datensouveränen entwickelt werden, um Inverse Transparenz zu realisieren. Dafür schlagen wir einen neuen Ansatz vor: „Transparency by Design".

2_ Ein neuer Weg für Software am Arbeitsplatz – Transparency by Design

Klassischerweise richten sich die Methoden des Software-Engineering auf die Nutzerinnen und Nutzer der Software aus. Mit seinem Buch „Usability Engineering" (Software-Ergonomie) führte Jakob Nielsen 1993 (siehe Nielsen[4]) hier einen neuen Standard ein, dem die Idee zugrunde lag, die Nutzbarkeit von Software für ihre Anwender zu verbessern. Der Fokus lag erstmals nicht mehr rein auf der Funktionalität oder den Kosten, sondern auf individuellen Faktoren in der Interaktion von Mensch und Software. Wie oben beschrieben glauben wir im Sinn des Design Thinking, dass als neue Interessenvertreter bei der Softwareentwicklung die Datensouveräne primär in den Blick genommen werden sollten.

Wie wir beschrieben haben, können die heute übliche fehlende Transparenz über die Datenverarbeitung sowie das

Ignorieren der Datenschutzsorgen von Mitarbeitenden zu Problemen im Betrieb führen. Führungskräfte sollten daher ein natürliches Interesse an einem verbesserten Umgang haben. Jedoch glauben wir, dass auch Softwarehersteller profitieren können. In vielen Betrieben in Deutschland ist es üblich, dass Mitarbeitende über die Mitbestimmung eine indirekte Einkaufsmacht ausüben können, was neue Software betrifft. Durch die Mitbestimmungspflicht zum Beispiel bei Einrichtungen zur Leistungs- und Verhaltenskontrolle sind solche Rechte auch gesetzlich verankert. Wenn ein Tool die Datenschutzsorgen nicht ausreichend adressiert, könnte die Konsequenz sein, dass es gar nicht gekauft wird oder ein Unternehmen sich zumindest nach Alternativen umsehen wird.

Dabei begegnet der Datenschutz so, wie er heute konzipiert ist, im Softwaredesign der Komplexität der entwickelten Tools nicht adäquat. Wir identifizieren diesbezüglich drei wichtige Treiber:

1. Anwendungsfälle werden komplexer und die Datennutzung schwerer überschaubar, vor allem für technisch nicht versierte Nutzerinnen und Nutzer.
2. Eine pauschale Entscheidung für oder gegen eine Datenfreigabe ist oft nicht sinnvoll treffbar, da viel vom Kontext der Nutzung abhängt.
3. Software entwickelt sich kontinuierlich weiter (agile Entwicklung, Software-as-a-service), weshalb Wissen über die Datenverarbeitung schnell veraltet.

Doch was ist die Alternative? Wir schlagen Transparency by Design vor (siehe Zieglmeier & Pretschner[10]). Grundprinzip ist eine Weiterentwicklung von Privacy by Design: Bereits bei der Konzeption von Software sollte Datennutzungstransparenz mitgedacht werden, um Softwaredesign für empowerte Datensouveräne zu ermöglichen.

1.1_ Das Konzept von Transparency by Design

Die grundlegende Idee von Transparency by Design basiert direkt auf dem Konzept der Inversen Transparenz: Wenn Datennutzende wie zum Beispiel Führungskräfte im Unternehmen auf Daten zugreifen (diese nutzen), sollen Datensouveräne gemäß dem Prinzip „Watch the Watcher" darüber Transparenz erhalten. Konzeptuell erfordert das drei Schritte:

1. Überwachung aller Datenzugriffe
2. Verifizierung der Authentizität der Zugriffe und Speicherung
3. Verfügbarmachen der gespeicherten Zugriffe

Wir können diese drei Schritte, wie im Softwaredesign üblich, auf jeweils getrennte konzeptionelle Komponenten übertragen. Jede Komponente bekommt eine von den anderen Komponenten getrennte Aufgabe. Das ermöglicht uns bei der konkreten Umsetzung in die Praxis, die Implementierungen auszutauschen oder zu variieren, ohne das grundlegende Konzept zu verändern. Entsprechend ergeben sich aus dieser Liste drei konzeptuelle Komponenten: Ein *Monitor* überwacht Datenzugriffe (1), eine *Überwachungsinstanz* prüft und speichert das Nutzungsprotokoll (2) und das *Anzeigentool* macht die Information für den Datensouverän verfügbar (3). Schematisch ist das in Abbildung 1 zu sehen.

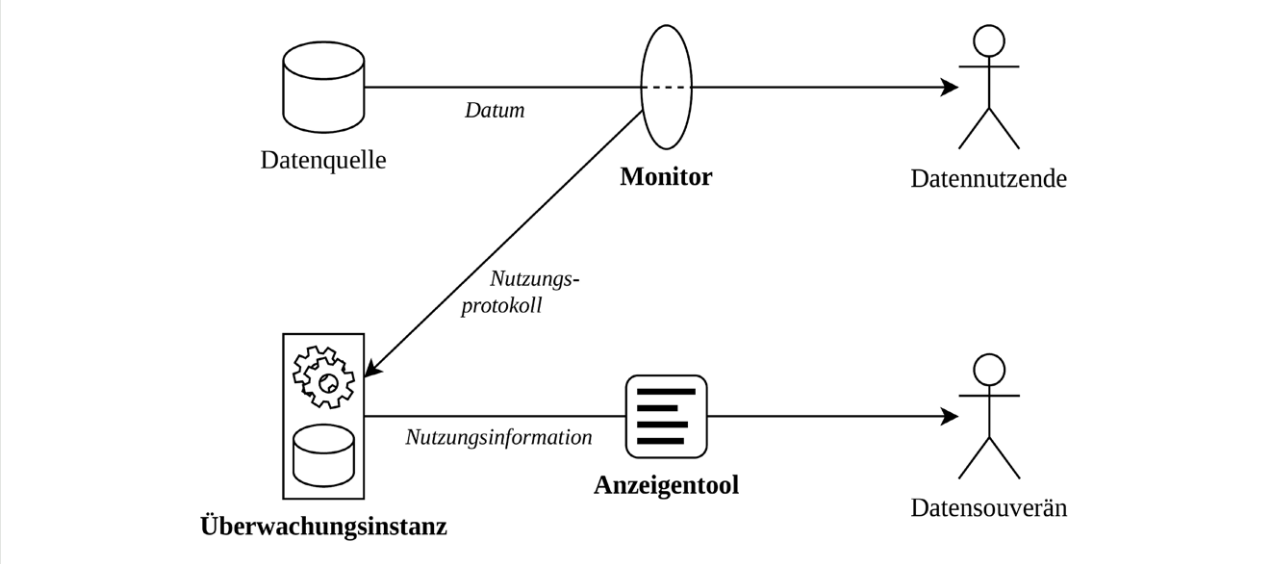


Abbildung 1: Transparentmachen von Datenzugriffen durch ein konzeptionelles Framework für Transparency by Design (Zieglmeier & Pretschner[10])

1.2_ Implikationen für Softwaredesign

Nun stellt sich die Frage, wie sich das abstrakte Konzept auf konkretes Softwaredesign auswirkt. Daten werden üblicherweise nicht über kontrollierte Kanäle an Datennutzende „geschickt", sondern bewegen sich ständig zwischen (Teil-) Systemen hin und her, werden aggregiert, kopiert, konvertiert oder verschoben. Ein „Zugriff" könnte hier einfach bedeuten, dass ein datenbasiertes Tool gestartet wird – allein für die Anzeige normaler Startseiten werden Datenzugriffe nötig. Was ist also ein Datenzugriff und wie können wir sicherstellen, dass wir ihn protokollieren?

Um diese Frage zu beantworten, gehen wir zurück auf den Zusatz „by Design" des Konzepts. Wir setzen beim Softwaredesign an, denn hier werden die Anwendungsfälle definiert und die Datenflüsse modelliert. Dabei erwartet Transparency by Design, dass bereits in der Konzeptphase einer Software, die mit persönlichen Daten arbeitet, die Transparenz für Datensouveräne über die Verarbeitung ihrer Daten mitgedacht wird. Solange dabei Daten nur im System verschoben oder manipuliert werden, ohne Datennutzenden zur Verfügung gestellt zu werden, besteht kein Änderungsbedarf. Dieser wichtige Aspekt ist auch der Grund, wieso die Transparenz nicht einfach im Nachhinein ermöglicht werden kann: Jegliche Überwachung einer existierenden Software läuft Gefahr, entweder relevante Zugriffe zu übersehen oder irrelevante Zugriffe unnötig zu protokollieren und dadurch Datensouveräne zu überfordern. Ein Nutzungsprotokoll soll genau ab dem Moment angelegt werden, in dem Datennutzende letztlich Zugriff auf die Daten erhalten.

Sofort ergeben sich Folgefragen. Viele Tools erlauben Exportfunktionen oder arbeiten direkt mit Dateien auf dem Speicher, die auch extern zugreifbar sind. Wie sollen diese Daten effektiv überwacht werden? Gar nicht. Denn während es heute noch gängig ist, Nutzern und Nutzerinnen Zugriff auf die unterliegenden Dateien zu geben, zum Beispiel über Exportfunktionen oder Netzwerkspeicher, ist ein klarer Trend hin zu Cloud-Software zu erkennen, die nur noch als „Service" angeboten wird. Daten sind hier nicht mehr Dateien, über die die Nutzenden verfügen, sondern abstrakte Konzepte, die Funktionen ermöglichen. Dadurch verlieren Nutzer so viel Kontrolle, dass in Artikel 20 der DSGVO sogar bereits wieder Datenportabilität und ein Recht auf Export gefordert wird. Auf den ersten Blick scheint dieses Recht sogar in Konflikt mit unserem Konzept zu stehen, aber hier lohnt ein genauerer Blick: Es bezieht sich ausschließlich auf die eigenen Daten, bei denen natürlich kein Risiko für Missbrauch besteht, wenn sie dem Datensouverän selbst zur Verfügung stehen.

Schwerer wiegt die Frage nach dem Umgang mit Daten-

Daten sind hier nicht mehr Dateien, über die die Nutzenden verfügen, sondern abstrakte Konzepte, die Funktionen ermöglichen.

zugriffen, die wir als „Umgebungsnutzung von Daten" (ambient usage) bezeichnen wollen. Ein Beispiel soll zur Illustration helfen: Das Tool Jira Software von Atlassian, das die Planung und Verwaltung von Softwareprojekten ermöglicht, dient uns im Projekt als Anwendungsbeispiel. Wenn hier die Startseite geöffnet wird, was automatisch beim Aufruf des Tools passiert, öffnet sich ein sogenanntes „Dashboard". Das ist nichts weiter als eine Ansammlung von kleinen Fenstern, die jeweils eine eigene Analyse, Funktion oder Sicht auf Daten darstellen (siehe Abbildung 2). In einem üblichen Dashboard, wie dem unten dargestellten, ist dabei meist nur ein Teil der Fenster zeitgleich sichtbar. Wenn eine Datennutzerin nun also Jira Software öffnet, um einfach eine Suche durchzuführen, wird sie trotzdem automatisch das Dashboard laden. Auch wenn sie dieses nicht beachtet und sofort auf die Suche wechselt, finden hier unzählige Datenzugriffe statt.

Wenn diese Zugriffe nun alle protokolliert werden, könnte das Datensouveräne verunsichern und verwirren. Zugleich wäre es wohl unmöglich, zu erkennen, ob die Datennutzerin die auf dem Dashboard sichtbaren Analysen vielleicht doch beachtet hat. Eye-Tracking mag theoretisch eine Lösung sein, praktisch wäre es wohl zu invasiv und nicht umsetzbar.

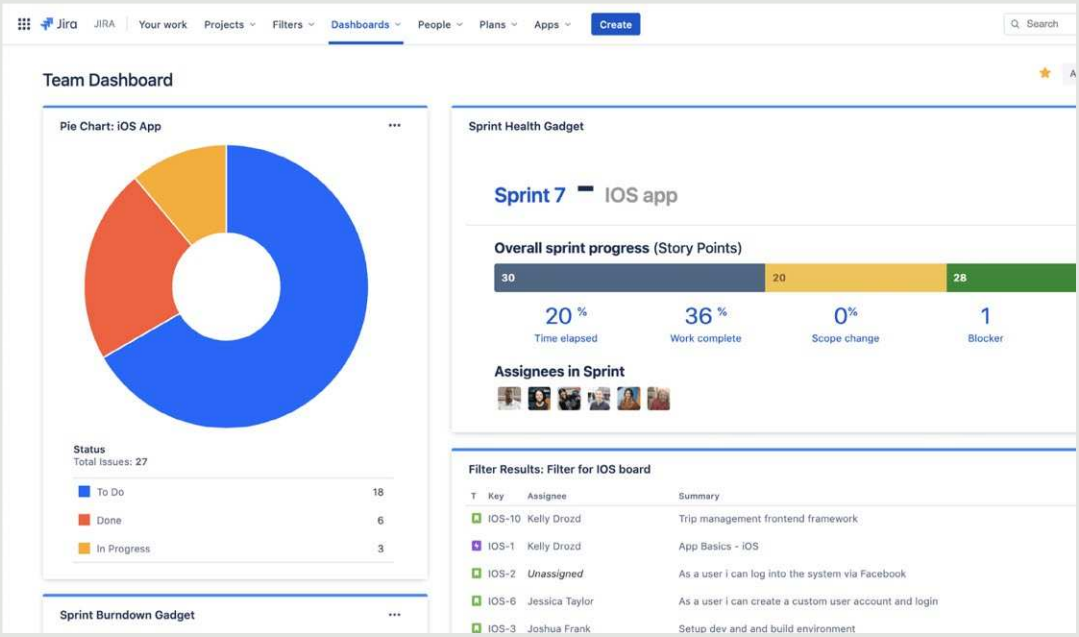


Abbildung 2: Ein beispielhaftes Dashboard von Jira Software (Quelle: www.atlassian.com)

Stattdessen könnte eine Konsequenz im Umgang mit solcher Umgebungsnutzung von Daten sein, dass sich Interaktionsparadigmen grundlegend verändern. Eine Möglichkeit wäre, dass nicht automatisch unzählige Datenzugriffe gemacht werden, ohne dass die Datennutzenden sich bewusst dafür entscheiden. Am Beispiel des Dashboards könnte das bedeuten, Analysen nur auf explizite Anfrage freizuschalten und sonst zu verbergen (siehe Abbildung 3).

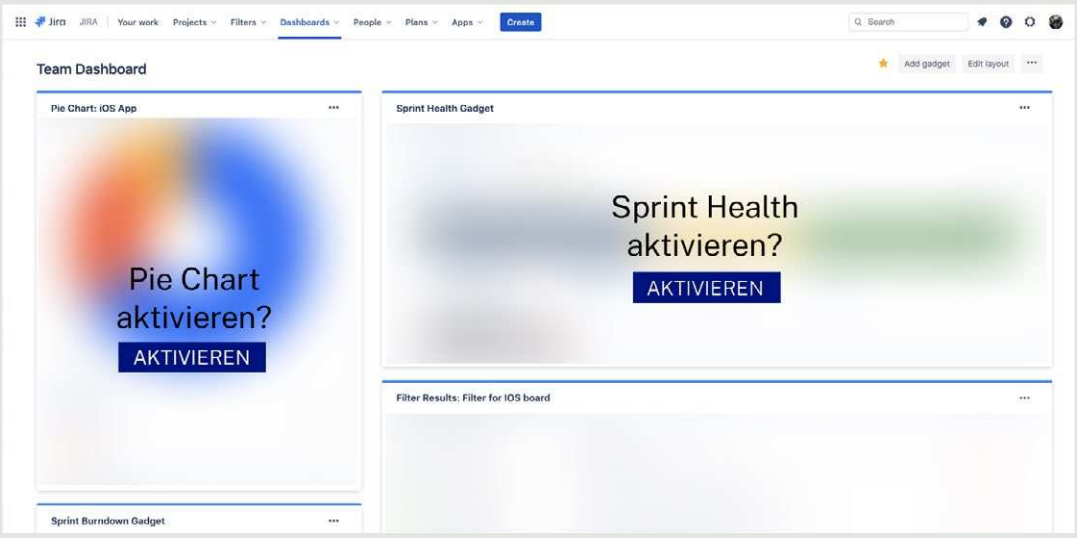


Abbildung 3: Modifiziertes Dashboard von Jira Software mit Transparency by Design (Anpassungen durch die Autoren)

Langfristig könnte sich durch Transparency by Design Software also weg von den aktuellen Prinzipien der freien Datennutzung und „Beliebigkeit“ von Analysen hin zu einem bewussteren Umgang mit Daten verändern.

1.3_ Fälschungssichere und vollständige Protokollierung

Bis hierhin haben wir uns mit der Überwachung von Datenzugriffen befasst. Tendenziell liegt in diesem Schritt auch die wichtigste Grundlage für Inverse Transparenz; doch ohne fälschungssicheres Protokoll (Überwachungsinstanz, siehe Abbildung 1) entfaltet sie nicht ihr volles Potenzial.

Dabei steht die Vertrauenswürdigkeit der gesammelten Protokolle im Fokus. Vertrauen in die Nutzungsprotokolle ist eine zentrale Voraussetzung, um das Potenzial von Inverser Transparenz zu realisieren. Es wird schnell klar, dass ein unvollständiges oder manipuliertes Protokoll keinen Wert darstellt. Wenn Mitarbeitende sich nicht auf eine korrekte und fälschungssichere Protokollierung verlassen können, verlieren die Daten ihre Macht. Wir müssen dabei vor allem für die Integrität der Daten garantieren. Das bedeutet: Wir benötigen die Sicherheit, dass jede Datennutzung in einem Protokolleintrag gespeichert wird, aber genauso wichtig ist, dass jedem Eintrag wiederum eine tatsächlich stattgefundene Datennutzung zugrunde liegt.

Hierbei können wir verschiedene Ansätze identifizieren, die diese Integrität sicherstellen können. Dabei lohnt es sich, aus der Perspektive der Nutzer von Inverser Transparenz, also der Datensouveräne und Datennutzenden, zu denken. Beide haben ein Interesse daran, dass die Nutzungsprotokolle vollständig und korrekt sind. Datensouveräne, weil sie die Sicherheit wollen, dass kein Datenmissbrauch unerkannt bleibt. Datennutzende, weil sie nicht aufgrund von manipulierten Nutzungsdaten in Schwierigkeiten geraten wollen. Für unsere Überlegungen im Folgenden ist es dabei nicht relevant, ob die Überwachungsinstanz, beziehungsweise ihre Funktion, als separates Programm oder als Teil der Software selbst realisiert wird; konzeptionell ändert sich nichts.

Langfristig könnte sich durch Transparency by Design Software also weg von den aktuellen Prinzipien der freien Datennutzung und „Beliebigkeit“ von Analysen hin zu einem bewussteren Umgang mit Daten verändern.

Wir müssen davon ausgehen, dass jede Mitarbeiterin und jeder Mitarbeiter im Unternehmen sowohl als Datensouverän als auch als Datennutzer und -nutzerin agieren kann, inklusive der höchsten Entscheiderinnen und der Systemadministration. Das bedeutet, dass potenziell jede Partei ein Manipulationsinteresse haben kann. Insofern gehen wir von einer „Umgebung ohne Vertrauen“ (trust-free environment) aus. In den letzten Jahren entwickelt sich eine Sicherheitstechnologie von Intel, die sogenannten Software Guard Extensions (SGX), zur logischen Wahl für solche Umgebungen. Ohne zu tief ins Detail zu gehen, wird hier durch kryptographische Methoden und eine abgeschottete Umgebung im Computerprozessor sichergestellt, dass Programme und Daten auch auf nicht kontrollierbaren Systemen manipulationsfrei bleiben. Die Technologie hat Potenzial und kann in unserem Fall als Lösung dienen, doch erfordert sie weiterhin Vertrauen – vor allem in Intel als Hersteller des Prozessors.

Deshalb haben wir auch alternative Wege erforscht, mit dem Ziel, in einer Umgebung ohne Vertrauen komplett auf Basis kryptographischer Methoden die Integrität der Nutzungsprotokolle sicherzustellen: mit Blockchain-Technologie (siehe Zieglmeier & Loyola Daiqui[9]). Zugrunde liegt hierbei die Idee, dass alle, die am System teilnehmen. eine Kopie der Nutzungsprotokolle in verschlüsselter Form als Blockchain speichern. Bei einer Zugriffsanfrage wird kein Zentralrechner angefragt, sondern ein sogenannter Knoten, der dem Datensouverän gehört, bekommt die Anfrage zugestellt. Wenn der Zugriff zugelassen wird, erstellt dieser Knoten automatisiert einen Protokolleintrag und verteilt diesen im Netzwerk.

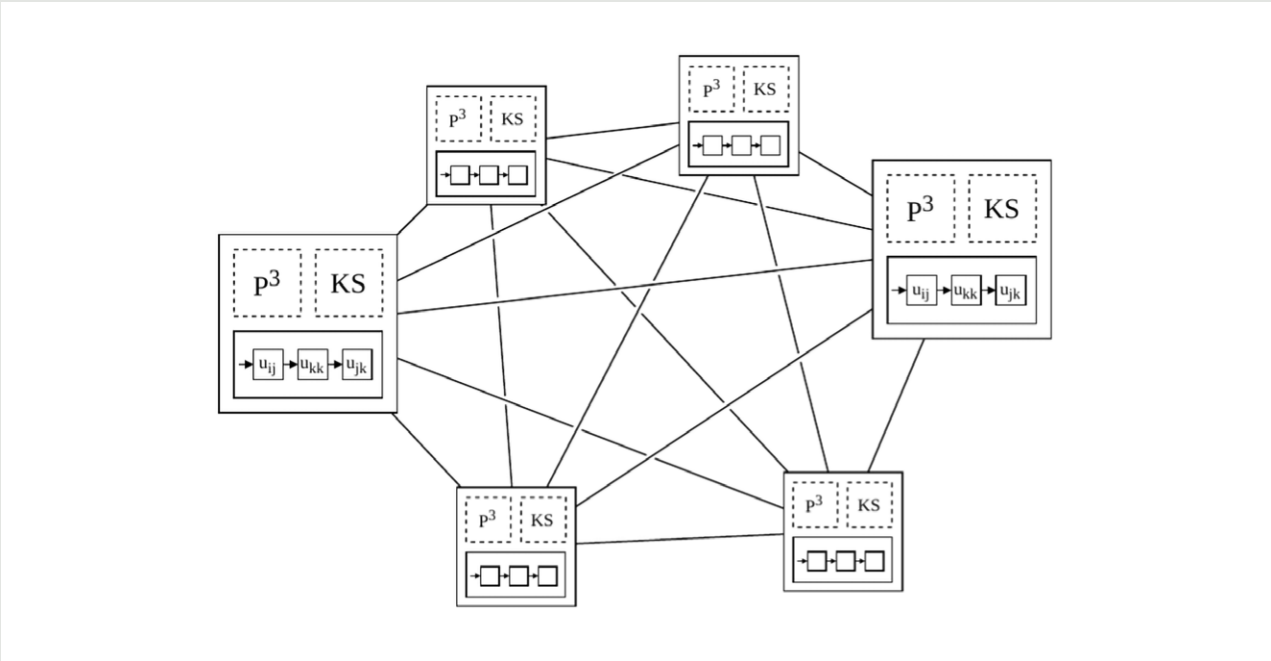


Abbildung 4: Peer-to-peer-Architektur des Blockchain-Netzwerks (Zieglmeier & Loyola Daiqui[9])

Blockchainlösungen sind oft in der Kritik, vor allem wegen der hohen Energieverbräuche, aber auch wegen der Frage nach ihrer Notwendigkeit. In unserem Konzept stellt zumindest ersteres kein Problem dar: Statt auf die üblichen energieintensiven Mechanismen zur Verifizierung zu setzen, können wir mithilfe der echten Identitäten der Netzwerkteilnehmenden und des Proof-of-Authority-Prinzips (siehe Wood[8]) ähnliche Garantien geben, ohne große Energieverbräuche zu erfordern. Auf der anderen Seite ist aber die Frage nach der Notwendigkeit einer Blockchain weniger eindeutig zu beantworten.

Idealerweise handelt es sich am Arbeitsplatz nicht um eine Umgebung ohne Vertrauen, weshalb eine Technologie wie Blockchain vielleicht nicht notwendig ist. Allerdings stellt sich die Frage, in welchen Fällen das nicht mehr zutrifft und ob nicht gerade dann den Nutzungsprotokollen besonderer Wert zukommt. Speziell der Fall eines Streits mit einer Führungskraft scheint hier relevant zu sein. Nehmen wir an, dass ein Missbrauch von Daten durch die Führungskraft erfolgt ist, um eine unerlaubte Leistungskontrolle durchzuführen. Nach diesen Analysen wird dem Mitarbeiter gekündigt, weshalb er sich zur Wehr setzen will. Er prüft das Protokoll der Datenzugriffe, um mögliches Fehlverhalten zu erkennen. Kann er in dieser Situation noch auf die Systeme der Firma vertrauen? Die Führungskraft könnte gut vernetzt sein und hat möglicherweise bereits die Anweisung zum Löschen der belastenden Daten gegeben. Es liegt nahe, dass nur eine Lösung komplett ohne Notwendigkeit von Vertrauen hier verlässlich als Datengrundlage dienen kann. Trotzdem wird es wichtig bleiben, als Maxime nicht absolute Sicherheit anzustreben, sondern auch die Nutzbarkeit im Auge zu behalten (siehe dazu auch Tøndel et al.[6]).

Es liegt nahe, dass nur eine Lösung komplett ohne Notwendigkeit von Vertrauen hier verlässlich als Datengrundlage dienen kann.

3_ Die technische Umsetzung Inverser Transparenz in der Praxis – das Konzept der Toolchain

Im Rahmen des Forschungsprojekts haben wir uns zum Ziel gesetzt, die Idee der Inversen Transparenz in die Praxis zu bringen und im Rahmen eines betrieblichen Praxislaboratoriums bei der Software AG zu erproben. Von zentraler Bedeutung für dieses Vorhaben war, die abstrakten Ideen auf konkrete und adressierbare Probleme zu reduzieren. Im ersten Schritt identifizierten wir deshalb einen relevanten Anwendungsfall bei unserem Praxispartner Software AG: das Tool Jira Software. Nun sollte es im zweiten Schritt darum gehen, die theoretisch konzeptionierte Transparenzlösung (siehe Abbildung 1) in die Praxis zu bringen. Da wir mit einer vorhandenen Software-Suite umgingen, entwickelten wir in einem mehrschrittigen Verfahren eine realisierbare Umsetzung der Inversen Transparenz in der Praxis: die Inverse-Transparenz-Toolchain.

Den Prinzipien „Trennung der Belange" (separation of concerns) und Software-as-a-service folgend, realisierten wir jede konzeptionelle Komponente als selbstständigen Web-Service. Dabei erfüllt der „Overseer" die Aufgabe der Überwachungsinstanz und die Web-App „Watch the Watcher" ist das Anzeigentool (siehe Abbildung 5).

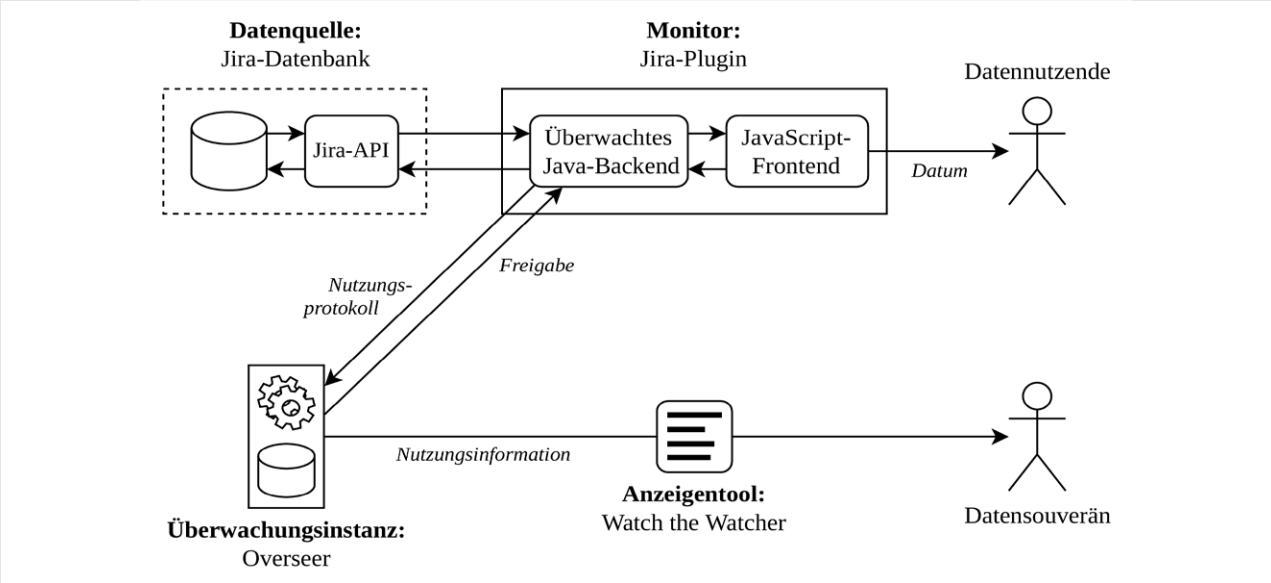


Abbildung 5: Instantiierung des Konzepts für Transparency by Design mit Web-Services am Anwendungsbeispiel Jira Software (Zieglmeier & Pretschner[10])

Für den Monitor mussten wir eine Lösung finden, um Transparenz im laufenden Betrieb möglichst wenig invasiv und ohne tiefgehende Administrationsrechte auf dem Zielsystem zu realisieren. Unsere Lösung war deshalb die Entwicklung von vollintegrierten Plugins. Anhand von Abbildung 5 lässt sich unsere Umsetzung leicht erklären: Wir entwickelten Plugins, die Monitor-Funktionalität bereits in einem überwachten Java-Backend integrieren. Jede Anfrage an das JavaScript-Frontend (die Fenster des Dashboards, siehe auch Abbildung 2) löst dann automatisch eine Protokollierung in der Überwachungsinstanz „Overseer" aus.

Nun ist es spannend zu beobachten, welche Anwendungsfälle sich auf Basis der Toolchain realisieren lassen und welche Weiterentwicklungen und Anpassungen an den Tools sich aus dem Einsatz in der betrieblichen Praxis ergeben.

4_ Zusammenfassung

Wir argumentieren, dass Softwaredesign sich weiterentwickeln muss. Als nächsten Schritt nach Privacy by Design, einer wichtigen Grundlage für den Datenschutz, schlagen wir deshalb Transparency by Design vor: das Mitdenken der Transparenzinteressen der Datensouveräne bereits bei der Entwicklung, um Inverse Transparenz zu realisieren. Das bedeutet einen Wandel vom stark präventiven Umgang mit Daten (Privacy by Design) hin zu mehr Offenheit, flankiert durch vollständige Transparenz über die Verarbeitungsprozesse (Transparency by Design). Drei Treiber begründen dies hauptsächlich: Komplexere Anwendungsfälle erschweren die Überschaubarkeit von Systemen (1), pauschale Entscheidungen für oder gegen Datenfreigaben sind ohne Kontext nicht treffbar (2) und Software ist nicht mehr statisch, sondern entwickelt sich laufend weiter, was regelmäßige Überprüfbarkeit erforderlich macht (3). Dabei glauben wir, dass diese Veränderung im Softwaredesign fundamentale Änderungen an Interaktionsparadigmen zur Folge haben kann und das Potenzial hat, die Datensouveränität von Individuen zu stärken. Um das Konzept der Inversen Transparenz unmittelbar erlebbar zu machen und unsere Hypothesen auf den Prüfstand stellen zu können, haben wir deshalb eine konkrete Umsetzung von Inverser Transparenz in Form der Toolchain entwickelt, die uns im Praxislaboratorium als technische Grundlage diente. Im Zusammenspiel zwischen Toolchain und Praxislaboratorium liegt ein zentraler Wert unseres Forschungsansatzes.

5_ Referenzen

1 Bhave, Devasheesh P., Teo, L., Dalai, R. Privacy at work: A review and a research agenda for a contested terrain. Journal of Management, 46, 1 (2020), 127–164. Supplementals.

2 Cavoukian, A. Privacy by design: The 7 foundational principles. Infor-mation and privacy commissioner of Ontario, Canada, 2009.

3 Feng, W., Tu, R., Lu, T., Zhou, Z. Understanding forced adoption of self-service technology: The impacts of users psychological reactance. Behaviour & Information Technology, 38, 8 (2019), 820–832.

4 Nielsen, J. Usability Engineering. Academic Press, 1993.

5 Teebken, M., Hess, T. Privacy in a Digitized Workplace: Towards an Understanding of Employee Privacy Concerns. In Proceedings of the 54th Hawaii International Conference on System Sciences. 2021, 6661–6670.

6 Tøndel, I. A., Soares Cruzes, D., Jaatun, M. G. Achieving ‚Good Enough' Software Security: The Role of Objectivity. In Proceedings of the Evaluation and Assessment in Software Engineering. April 2020, 360–365.

7 Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., Sussman, G. J. Information Accountability. Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2007-034. June 2007.

8 Wood, G. Proof-of-authority private chains. 2015. https://github.com/ethereum/guide/blob/master/poa.md.

9 Zieglmeier, V., Loyola Daiqui, G. (2021). GDPR-Compliant Use of Block-chain for Secure Usage Logs. In Proceedings of the Evaluation and As-sessment in Software Engineering. June 2021, 313–320.

10 Zieglmeier, V., Pretschner, A. Trustworthy transparency by design. Vorveröffentlichung auf arXiv (2021). arXiv:2103.10769.