Software als Institution und ihre Gestaltbarkeit

Carsten Orwat · Oliver Raabe
Erik Buchmann · Arun Anandasivam
Johan-Christoph Freytag
Natali Helberger · Kei Ishii
Bernd Lutterbeck · Dirk Neumann
Thomas Otter · Frank Pallas
Ralf Reussner · Peter Sester
Karsten Weber · Raymund Werle

Software greift immer mehr regulierend in unser Leben ein, d. h. sie wirkt ähnlich wie herkömmliche Institutionen wie z. B. Recht, Verträge oder Verhaltensnormen. Die Anwendung der interdisziplinären Institutionenforschung scheint daher zur Erforschung und Gestaltung dieses Sachverhalts vielversprechend zu sein.

Einleitung

"… a newly salient regulator." [14, S. 5] und "We are all regulated by software now."[4, S. 1758]: Diese Ausdrücke verdeutlichen die Thematik dieses Artikels. Mit dem zunehmenden Einsatz von Softwaresystemen entwickelt sich Software zu einem Regelungsmechanismus nicht nur für individuelles Verhalten, sondern auch für zwischenmenschliche Interaktionen, z. B. in E-Commerce-Systemen, Online-Kooperationstools oder Systemen des digitalen Rechtemanagements. Software regelt Zugang sowie Austausch- und Nutzungsformen von Daten und Informationen in Forschung und Privatleben, oder sie bestimmt Transaktionsformen zwischen Unternehmen und in unternehmensinternen Abläufen. Ambient Intelligence oder Ubiquitous Computing werden Softwareregelungen in nahezu jeden Lebensbereich transportieren.

Da Entwicklung und Gestaltung von Software eine verantwortliche Rolle in der Gesellschaft erlangt haben, ist die Erforschung der Wirkungen einer weitgehenden Durchdringung der Alltags- und Berufswelt mit Softwareanwendungen sowie deren Gestaltungsoptionen dringend geboten. Mit diesem Artikel soll ein neuartiger, interdisziplinärer Forschungsansatz vorgestellt werden: die Betrachtung von *Software-Regelungsmechanismen als Institution*. Diese Perspektive ermöglicht die Einbindung von Erkenntnissen und Forschungsmethoden aus unterschiedlichsten Teildisziplinen

DOI 10.1007/s00287-009-0404-z

Carsten Orwat
Karlsruher Institut für Technologie, Institut für Technikfolgenabschätzung und Systemanalyse, Postfach 3640, 76021 Karlsruhe, Deutschland
E-Mail: orwat@kit.edu

Oliver Raabe · Peter Sester
Karlsruher Institut für Technologie, Institut für Informationsrecht, Karlsruhe, Deutschland
E-Mail: oliver.raabe@kit.edu

Erik Buchmann · Ralf Reussner
Karlsruher Institut für Technologie, Institut für Programmstrukturen und Datenorganisation, Karlsruhe, Deutschland
E-Mail: erik.buchmann@kit.edu

Arun Anandasivam
Karlsruher Institut für Technologie, Institut für Informationswirtschaft und -Management, Karlsruhe, Deutschland

Johan-Christoph Freytag
Humboldt-Universität zu Berlin, Institut für Informatik, Berlin, Deutschland

Natali Helberger
Universität Amsterdam, Institut für Informationsrecht, Amsterdam, Niederlande

Kei Ishii · Bernd Lutterbeck · Frank Pallas
Technische Universität Berlin, Lehrstuhl Informatik und Gesellschaft, Berlin, Deutschland

Dirk Neumann
Universität Freiburg, Lehrstuhl für Wirtschaftsinformatik, Freiburg, Deutschland

Thomas Otter
Gartner Deutschland GmbH, München, Deutschland

Karsten Weber
Universität Opole, Opole, Polen

Raymund Werle
Max-Planck-Institut für Gesellschaftsforschung, Köln, Deutschland

Published online: 16 December 2009

Zusammenfassung
Software regelt immer mehr zwischenmenschliche Interaktionen. Üblicherweise werden die Funktionsmechanismen, Wirkungen und Gestaltungsoptionen von Regeln in der Institutionenforschung behandelt. In diesem Artikel soll beleuchtet werden, inwieweit sich Ansätze der Institutionenforschung auf Software anwenden lassen und was sich aus dieser Forschungsperspektive zu den Regelungswirkungen und Gestaltungsoptionen von Software ableiten lässt.

der Institutionenforschung, z. B. aus Wirtschafts-, Rechts-, Organisations- und Politikwissenschaften sowie aus Soziologie und (politischer) Philosophie, wobei der Neuen Institutionenökonomie [26, 27] eine besondere Rolle zukommen kann. Im Allgemeinen befasst sich die Institutionenforschung mit der Analyse der Wirkungsweisen von Regelsystemen, um ihre Effekte in Organisationen, bei wirtschaftlichen Transaktionen oder auf die Gesellschaft zu verstehen und sie möglichst optimal zu gestalten. Um das Potenzial dieser Perspektive auszuloten und zukünftige Forschungsschwerpunkte zu umreißen, wurde im Dezember 2008 ein interdisziplinärer Workshop mit 16 Experten aus Informatik, Recht, Wirtschaft, Philosophie und Soziologie veranstaltet (http://www.kit.edu/sai2008/). Dieser Artikel fasst die dabei gewonnenen Erkenntnisse zusammen und zeigt einige Potenziale auf, die eine interdisziplinäre Institutionenforschung für die Gestaltung von Softwaresystemen haben kann, um einen Diskurs zum Thema „Software als Institution“ in Gang zu setzen.

Im Folgenden wird untersucht, was es bedeutet, Software als Institution aufzufassen, wie das Verhältnis zu bestehenden Institutionen und insbesondere zum Recht zu sehen ist, wie Softwareanwendungen effektive Regelungsmechanismen darstellen können und welche Erkenntnisse aus der Institutionenforschung für die Gestaltung von Softwareinstitutionen genutzt werden können.

Warum „Software als Institution“?

Zum Institutionenbegriff

Unter Institutionen werden anerkannte Systeme von Regeln verstanden, die zwischenmenschliche Interaktionen ermöglichen, strukturieren oder beschränken [7]. Institutionen umfassen förmliche Regeln (staatlich gesetztes Recht, niedergeschriebene Verträge) und ungeschriebene Verhaltensnormen oder soziale Konventionen [7, 16, 19, 20, 22]. Wesentliche Elemente sind die Regeln selbst, aber auch die Mechanismen zur Durchsetzung des Gesollten [35]. Oft spielen verschiedene Institutionstypen zusammen, z. B. werden Verträge nur durchsetzbar, wenn ein funktionierendes Rechtssystem zur Verfügung steht [18, 26].

Werden Institutionen angewandt, kontrolliert und durchgesetzt, so bringen sie Ordnung in soziale Tätigkeiten und vermindern Unsicherheit, die z. B. aus willkürlichem oder opportunistischem Verhalten in einer Interaktion resultiert. So werden z. B. bei Handelstätigkeit die Kosten der Anbahnung, Durchführung, Kontrolle und Durchsetzung von Austauschbeziehungen, d. h. die Transaktionskosten, dadurch bestimmt, wie sicher eine Leistung gehandelt werden kann. Entscheidend ist, welche Institutionen beteiligt sind, die opportunistisches Verhalten ausschließen oder verringern [34]. Die Gestaltung von Institutionen sorgt also dafür, wie effizient Handel stattfindet und hat Einfluss auf Arbeitsteilung, Spezialisierung und Produktivität. Ausgestaltung und Wirksamkeit von Institutionen gehören zu den bestimmenden Faktoren für wirtschaftliche Leistungsfähigkeit, gesellschaftlichen Wohlstand und soziale Entwicklung, sowohl hemmend als auch fördernd [18, 19].

Verständnis von Softwareinstitutionen

Software kann eine doppelte Funktion einnehmen. Einerseits wird sie als Transmissionsmedium und Durchsetzungsmechanismus von anderen Institutionentypen eingesetzt, z. B. bei DRM-Systemen, die Verträge durchsetzen. Hier entspricht Software einer technischen Vorkehrung zur Durchsetzung von Rechten, ähnlich wie beispielsweise Architektur zur Abgrenzung von Eigentumsrechten dient [11]. Andererseits kann Software neue Regeln definieren und durchsetzen, z. B. durch die Festlegung eines Workflows in Unternehmen. Daher kann Software auch den Charakter einer eigenständigen Institution haben [4]. Die letztere Funktion ist bisher nur wenig untersucht worden [15, S. 4].

Reguliert Software individuelles Verhalten oder soziale Interaktionen, so kann man von

Abstract
Software increasingly establishes the rules of human interactions. Usually, functional mechanisms, implications, and options of shaping such rules are the subject of institutional research. Thus, we describe in this paper how institutional research can be applied to software matters and which insights about implications and options of shaping can be obtained from this research field.

„programmierten Institutionen“ oder Softwareinstitutionen sprechen. Softwareanwendungen beinhalten ein System von formalen Regeln, die a) vom Softwareentwickler implementiert werden oder b) die Berechtigte durch Systemeinstellungen anderen Nutzern auferlegen. Softwareinstitutionen wirken oft mit andere Regeln zusammen, z. B. mit Recht [32] (Urheberrecht, Vertragsrecht) oder sozialen Normen.

Wie regelt Software? Gegenüber konventionellen Institutionen wie sozialen Normen, Verträgen oder Recht, kann Software einige Besonderheiten aufweisen [4, 13, 25, 31]:

- Software funktioniert automatisch. Regeln werden automatisch und ohne menschliche Auslegungsspielräume durchgesetzt. Es wird vorab definiert, welche Handlungsmöglichkeiten Nutzer haben bzw. anderen Nutzern gewähren können.
- Software regelt unmittelbar. Im Gegensatz zu z. B. Recht, das erst ex post durch Gerichte durchgesetzt wird, regelt Software ex ante effektiv die Handlungsspielräume der Nutzer.
- Software regelt teilweise unbemerkt. Die Nebenfunktionen von Softwareanwendungen können wenig oder gar nicht sichtbar für die Betroffenen sein.
- Software regelt zunehmend kontextorientiert. Regelungen können kontextabhängig erfolgen, z. B. entsprechend der Umgebung oder der Situation des Nutzers.
- Softwareregeln sind dynamisch. Über die Weiterentwicklung der Software können sich Regeln ändern. Die Regelentwicklung verläuft nach softwarespezifischen Mustern, die oft durch ökonomische oder technische Motive geprägt sind.
- Software ist vielgestaltig und präzise formbar. Software ist detailliert für komplexeste Interaktionen gestaltbar. Teilweise werden diese dadurch überhaupt erst regelbar.

Softwareinstitutionen und Recht

In den Rechtswissenschaften wurde die Steuerungsfähigkeit von Software bereits untersucht (vgl. „Lex Informatica“ [25], „Code as Law“ [13, 14], „Regulation by Software“ [4] oder „Regulation by Machine“ [24]). Ein prominentes Beispiel sind Systeme des digitalen Rechtemanagements (DRM), welche die im Urheberrecht definierten Nutzungsmöglichkeiten durchsetzen aber auch verändern bzw. einschränken können. Dabei bestehen Bedenken, dass „neues Recht“ durch private Akteure geschaffen wird, welches mit staatlich gesetztem Recht kollidieren kann [6, 13, 24, 29].

Allerdings ist Software nicht mit Recht gleichzusetzen, d. h. Software entsteht in der Regel außerhalb der etablierten, demokratischen Gesetzgebungsverfahren und die ihr zugrunde liegenden Regeln können nicht durch den Richter durchgesetzt werden. Der Ausdruck „Code as Law“ [13] verdeutlicht daher lediglich deren regulierende Wirkung jenseits des staatlichen Rechts. So kann die Institution „Software“ ähnliche Wirkungen erzeugen wie die Institution „staatliches Recht“. Dies gilt insbesondere für Regeln, die von privaten Akteuren als Verträge oder Selbstregulierung gesetzt und mit Software durchgesetzt werden.

Dabei ist es problematisch, dass staatlich gesetztes Recht, das unter Abwägung der Interessen verschiedenster Interessengruppen entsteht, durch private technisch durchgesetzte Regelungen „überschrieben“ oder modifiziert werden könnte [6, 24]. So ist beispielsweise die Institution Eigentumsrecht das Ergebnis der staatlichen Abwägung von Rechten der Eigentümer und den Interessen der Allgemeinheit. Auch das geistige Eigentumsrecht bzw. Urheberrecht ist das Ergebnis von Abwägungen zwischen Verwertungsinteressen der Urheberrechtsinhaber und Schrankenbestimmungen im Interesse der Allgemeinheit (z. B. Zitiermöglichkeiten für die Wissenschaft). Software wird nicht in einem Prozess staatlicher Güterabwägung entwickelt und etabliert, sondern folgt in der Regel technischen Möglichkeiten und Marktmechanismen, insbesondere der Marktstellung von Anbieter und Nachfrager.

Regelungswirkungen von Softwareinstitutionen

Softwareinstitutionen können positive Wirkungen haben, z. B. im außerrechtlichen Bereich. Sie können sogar effektiver als konventionelle Institutionen wie soziale Normen oder Recht sein. So wurde in frühen File-Sharing-Netzwerken beobachtet, dass Nutzer Dateien zur Verfügung stellen, weil sie der sozialen Norm des Teilens folgten und erwarteten, dass andere sich ebenfalls daran orientieren [5]. Mit Wachsen der P2P-Netzwerke stieg jedoch die Zahl der sogenannten „free rider", die sich nicht an diese Norm hielten. Viele aktuelle P2P-Systeme knüpfen nun den Zugang zu Ressourcen an die Bereitstellung eigener Ressourcen. Auf technische Weise erzwingt Software reziprokes bzw. kooperatives Verhalten und verhindert Regelabweichungen [3].

Auch das Beispiel Datenschutz zeigt, dass Softwareinstitutionen gegenüber dem ordnungsrechtlichen Ansatz – mit seinen allseits bekannten Defiziten [28] – wirkungsvolle Regelungsmechanismen darstellen können. Ein Beispiel dafür sind hippokratische Datenbanken [1]. Klassische Datenbanken gestatten mindestens dem Administrator den Vollzugriff auf die Daten. Hippokratische Datenbanken hingegen unterliegen spezifischen Grundsätzen, nach denen personenbezogene Datensätze ausschließlich gemäß vorab definierter Regeln kontext- oder präferenzbezogen verarbeitet und dem Anwender zur Verfügung gestellt werden. Diese Regeln können vom Systembetreiber oder von den betreffenden (d. h. beschriebenen) Personen festgelegt werden. Es handelt sich also um eine automatisierte, explizit definierte, datenbanküberwachte und -gesicherte Einhaltung organisatorischer Regeln und Benutzerpräferenzen. Bei der Umsetzung dieser Regeln und Präferenzen ist der Ersteller jedoch nicht festgelegt. Ebenso wie beim Beispiel DRM und Urheberrecht kann auch bei Hippokratischen Datenbanken und Datenschutzrecht ein Spannungsverhältnis zu staatlich gesetztem Recht entstehen. Zudem ist nicht klar, welche Datenschutzregeln [2, 21] in welcher Form und mit welchem technischen Ansatz umzusetzen sind.

Gestaltungsfragen von Softwareinstitutionen

Wenn Software menschliches Verhalten und Interaktionen reguliert, stellt sich die Frage der Legitimation solcher Institutionen. Legitimation kann vor allem durch die Beteiligung der Betroffenen erreicht werden, z. B mit demokratischen Prozessen. Viele Regelungen der Software werden jedoch über Marktprozesse anerkannt, indem Software nachgefragt und angewandt wird. So kann Softwareinstitutionen die Legitimation fehlen. Je mehr Softwareinstitutionen Zwang auf Individuen ausüben, desto mehr ist zu fragen, inwieweit der Einzelne oder die Gesellschaft Einfluss auf die Entwicklung und Ausgestaltung von Software haben sollten. Da die Schaffung und Durchsetzung von Softwareinstitutionen oft bei einem Akteur oder bei wenigen Akteuren liegt, die keiner demokratischen Kontrolle unterzogen sind, sind die Entwicklungsprozesse von Software zu untersuchen, um diese Akteure und deren Motive aufzudecken sowie Möglichkeiten zur politischen oder staatlichen Partizipation bei der Softwaregestaltung [12] und im Allgemeinen Möglichkeiten zur Verbesserung der Legitimität zu etablieren.

Allerdings ist dabei zu berücksichtigen, dass Software nicht gleich Software ist. Stattdessen sind im Folgenden verschiedene Softwaretypen und einzelne Anwendungsfälle zu unterscheiden, z. B. proprietäre versus Open-Source-Software, Software als Marktstandards versus Komiteestandards sowie Software erstellt nach Software-Engineering-Prinzipien versus Software erstellt durch „handwerkliche" Entwicklung.

Partizipation durch Open-Source-Softwareentwicklung?

Im Design und Code der Software werden üblicherweise Nutzer- und Entwickleranforderungen umgesetzt. Oft sind nur wenige Akteure involviert, wie etwa ein Kraftwerksbetreiber bei der Entwicklung einer Kraftwerkssteuerung. Wenn Software nun zunehmend Einfluss auf die Gesellschaft ausübt, kann zur besseren Legitimierung gefordert werden, mehr Akteure in die Entwicklung einzubinden. Allerdings stellt sich die Frage, wie dies organisiert werden kann. Eine Möglichkeit besteht im Open-Source-Entwicklungsmodell, in dem verschiedene Formen einer Beteiligung zu erkennen sind. Einerseits können Nutzeranforderungen unkontrolliert, d. h. ohne zwischengeschaltete Regelungsinstanz über die Weiterentwicklung eigener, unabhängiger Softwarelösungen umgesetzt werden (forking) (siehe das Beispiel des Internet Relay Chat unten).

Andererseits können Anforderungen kontrolliert über Plattformen zum Fehler- und Anforderungsmanagement in den Entwicklungsprozess eingebracht werden. Dies kann entweder relativ wenig formalisiert oder in formalisierten Prozessen, wie beispielsweise beim „Java Community Process", geschehen. Dort finden sich detaillierte Systeme und Regeln zur Einreichung von Modifikationswünschen oder zu Entscheidungsverfahren über die Umsetzung. Auffallend ist, dass das Entscheidungsgremium von Unternehmen dominiert ist (z. B. IBM, Intel, SAP, Nokia), welche über die zeitlichen und finanziellen Ressourcen verfügen, um sich an den Verfahren zu beteiligen.

Da die Gruppe der Open-Source-Entwickler nicht unbedingt deckungsgleich mit der Gruppe der Nutzer ist, müssen die Bedürfnisse der letzteren Gruppe nicht zwangsläufig abgebildet sein. Damit Nutzer sich als Entwickler beteiligen und ihre Anforderungen unmittelbar einbringen können, müssen Programmierkenntnisse sowie zeitliche und finanzielle Ressourcen vorhanden sein. Ob die Entwicklung von Open-Source-Software im unmittelbaren Sinne der Nutzer erfolgt, ist daher an Voraussetzungen geknüpft.

Ein Beispiel für die Gestaltbarkeit von Softwareinstitutionen ist die Open-Source-Software „Internet Relay Chat" (IRC). Dabei stellen Privatpersonen mit IRC Software ausgerüstete Server für andere Nutzer zur Verfügung. Bei frühen Versionen wurde mit der Serversoftware die Regel implementiert, dass Nutzer nach dem Ausloggen ihre Nutzerkennungen (Nickname) und Diskussionsgruppen (Channel) verloren. Da mit Wachsen des IRC-Netzes auch die Konflikte um Kennungen stiegen, setzen Nutzer zusätzliche Software ein, die ein Einloggen simulierte und dadurch ein „Eigentum" an bestimmten Nutzer- und Gruppenkennungen ermöglichte. Dadurch wurde allerdings die Stabilität des IRC gefährdet. Die Serverbetreiber haben daher ihrerseits Software angeboten, mit der ebenfalls Kennungen gesichert werden konnten. Verbleibende Konflikte wurden durch das Aufsetzen einer großen Zahl von weiteren IRC-Netzen aufgelöst, die unterschiedliche Regelsysteme implementierten. Das IRC-Beispiel zeigt, dass Technik eine soziale Ordnung kreieren kann, die nahezu unabhängig vom bestehenden, äußeren Regelungsgefüge ist. Es ist ferner ein Beispiel für institutionelle Dynamik, da die Software und ihre Regeln ständig weiterentwickelt wurden. Es zeigt, dass Softwaresysteme – und damit Softwareinstitutionen – grundsätzlich auch durch Anwender oder Nutzer veränderbar sein können [8].

Legitimierung durch Standardisierung von Software?

Können Softwareinstitutionen einen höheren Grad an Legitimität erlangen, wenn sie in Standardisierungsverfahren gestaltet werden? Hierbei wird normalerweise zwischen privaten und staatlichen Verfahren unterschieden, wobei es aber auch Zwischenformen gibt.

Im nichtstaatlichen Sektor wird zwischen Marktstandards und Komiteestandards unterschieden. Marktstandards werden von Individuen oder Firmen entwickelt, gelangen implementiert in Produkten auf den Markt und werden durch Marktdurchdringung zum Standard. Marktstandards können einen Zwang zur Anpassung oder Implementierung für andere Akteure herstellen. Ihre Legitimierungsprobleme sind noch weitgehend ungelöst. Komiteestandards werden hingegen durch Standardisierungsorganisationen (z. B. DIN, ETSI, IETF, W3C) nach festen Verfahrensregeln entwickelt und beschlossen. Die Standardisierungsorganisationen bemühen sich, eine Input- oder Output-Legitimierung zu erreichen [33]. Input-Legitimierung zielt dabei auf die Prozesse der Standardentwicklung ab, d. h. auf Konsens bei Entscheidungen und durch eine ausgewogene Beteiligung aller Akteure, die Interessen am Standard haben bzw. von ihm betroffen sind. Problematisch ist, dass mit der Beteiligung in der Regel Kosten verbunden sind, auch durch den Aufbau der erforderlichen Expertise. Solche Komitees sind daher oft durch Industrieinteressen und technisch-ökonomische Aspekte dominiert, und andere Interessen wie z. B. Datenschutzfragen werden oft nur wenig berücksichtigt.

Output-Legitimierung richtet sich hingegen auf die Legitimierung des Ergebnisses, d. h. des Standards. Dabei wird weniger auf die Standardentwicklung in einem offenen Diskurs geachtet. Stattdessen bestimmen Verfahrensregeln die Einflussnahme von Interessengruppen von außen auf den Standardisierungsprozess (z. B. durch Kommentierung), um eine Berücksichtigung von Konsumenten-, Arbeitnehmer- oder sonstigen öffentlichen Interessen zu ermöglichen [33]. Insofern

sind für den nichtstaatlichen Standardisierungsbereich die Output-orientierten Verfahren eine Möglichkeit der Legitimierung.

Im Bereich der staatlich, d. h. durch ausdrückliche gesetzliche Gestaltungsanforderungen gesetzten Standards, stellt sich nicht mehr die Frage der Legitimität, sondern die der Durchsetzung. Bei neuen Verfahren, wie z. B. bei elektronischen Geschäftsprozessen im Energiebereich, wird die inhaltliche Ausgestaltung des Standards von den betroffenen Marktakteuren entwickelt, das Endergebnis wird aber dann als verbindlicher Rechtsakt durchgesetzt. Im Energiesektor werden so die für die elektronische Marktkommunikation zu verwendenden Datenprotokolle festgelegt und damit auch die Möglichkeiten der daran anknüpfenden Softwareimplementierungen. Dadurch wird zwar Rechtssicherheit für die beteiligten Akteure erzeugt. Allerdings birgt die faktische Beschränkung der Beteiligung auf wenige interessierte Marktakteure die Gefahr der Konservierung der von diesen präferierten Techniken in sich. Innovative Alternativvorschläge von nicht beteiligten Akteuren haben kaum eine Chance, bei der Standardisierung berücksichtigt zu werden. Würde man im Nachhinein Änderungen des Standards vornehmen – was durch den Änderungsvorbehalt des Gesetzes vorgesehen ist – entstünden für die ursprünglich Beteiligten unkalkulierbare Investitionsrisiken.

„Bottom-up"-Gestaltung von Software?

Gegenwärtig ist in der Softwareentwicklung ein Veränderungsprozess zu beobachten: Traditionell steuert eine zentrale Instanz den Entwicklungsprozess. Softwareanwendungen werden dabei „top down" durch Software-Engineering entwickelt, also durch Analyse der Ziele und Wirkungen, auf die die Software zugeschnitten wird. Im Gegensatz dazu verändert sich mit dem Aufkommen von Mashups oder serviceorientierten Architekturen (SOAs) die Art der Softwareentwicklung; es kommt zu Softwareentwicklung durch dezentrale, unabhängige Parteien. Statt der hierarchischen Ordnung finden sich eher markt- und netzwerkartige Strukturen, in denen Software „bottom up" entwickelt wird. Basisdienste (z. B. Google Maps) können durch vorab nicht bestimmbare Entwickler zu neuen Diensten verknüpft werden. Die Ergebnisse derartiger Softwareentwicklungsprozesse sind nicht mehr genau planbar, und das gilt auch für die so entstehenden Regelungsmechanismen. Beispielsweise kann Google die entstehenden Google-Maps-Mashups nicht planen und nur begrenzt durch die Gestaltung der Basisdienste beeinflussen [23].

Ein „bottom-up"-Vorgehen ist allerdings nur auf der Ebene der Anwendungsentwicklung möglich. Die Basisdienste oder Basistechnologien entstehen eher nach dem „top-down"-Ansatz. Zudem ist der „bottom-up"-Ansatz von der Offenheit und Kompatibilität der Softwaremodule abhängig, da sich sonst neue Abhängigkeiten ergeben. Künftige Forschungen können sich auf Institutionen zur Gestaltung des anbieterübergreifenden Zusammenspiels einzelner Mashup-Dienste konzentrieren oder z. B. Gewährleistung oder Haftung der Diensteanbieter regeln. Die bisher ungerichtete Entwicklung von Mashup-Diensten könnte z. B. verstärkt durch monetäre oder nichtmonetäre Anreize gelenkt werden.

Legitimierung durch Zertifizierung?

Softwareinstitutionen können auch dadurch legitimiert werden, dass die Einhaltung von Regelkatalogen – die einen gesellschaftlichen Konsens widerspiegeln – überprüft und zertifiziert wird. Bisher wurde die Zertifizierung in erster Linie zur Überwindung von Qualitätsproblemen diskutiert. Hier sind besonders Fehler bei der Abfrage der Nutzeranforderungen relevant, die bei der Umsetzung in Software mitgeführt werden. Die Lösung des Problems wird vor allem im systematischen Software-Engineering und im Einbau von Verifikationsschritten gesehen. Zu den für die Legitimierung relevanten Einzelfragen der Gestaltung von Zertifizierungssystemen kann die Institutionenforschung Aussagen liefern, beispielsweise zur Unabhängigkeit des Prüfers, zu Gebührenregelung, zu Prüfungs- und Sanktionseffektivität und -kosten oder zur Vermeidung opportunistischen Verhaltens der Prüfer [10].

Rechtskenntnis und -umsetzung bei der Softwareentwicklung

Viele gesellschaftliche Werte, die menschliche Interaktionen bestimmen (z. B. der Datenschutz), werden in formales Recht gegossen. Daher ist nach der Umsetzung rechtlicher Vorgaben in Software und den Rechtskenntnissen der Entwickler zu fragen. Dazu hat eine auf dem Workshop vorgestellte Studie ca. 100 Softwareentwickler befragt. Die Umfrage kommt zu einem ernüchternden Ergebnis hin-

sichtlich der Kenntnisse in den Rechtsbereichen Datenschutz, Barrierefreiheit und rechtliche Risiken von Webservices. Allerdings wurde auch der Wunsch vieler Softwareentwickler nach einer besseren rechtlichen Ausbildung und einer stärkeren Einbeziehung rechtlicher Fragen in den Entwicklungsprozess deutlich.

Grundsätzlich hat man erkannt, dass ein dichtes Netz einzelfallbezogener Normen die Vorhersagemöglichkeiten des Gesetzgebers überstrapazieren würde, denn nicht alle Regelungstatbestände können vorhergesehen werden. Regelt man nicht einzelfallbezogen, sondern schafft Regelungen mit Interpretationsspielraum, so stellt sich die Frage, ob die Werkzeuge der Interpretation bzw. Auslegung (juristische Kommentare, Lehrbücher etc.) für Softwareentwickler auch geeignet sind. Es ist eher zu vermuten, dass rechtliche Normen einerseits und Normen zum Entwurf und zum Funktionieren von Softwaresystemen andererseits nicht einfach ineinander überführbar sind. Daraus kann der Bedarf nach einem „generischen" Recht für Softwareentwickler abgeleitet werden. Das wäre ein Aggregat von Regularien, welches Softwaregestalter handhaben könnten, um damit gesellschaftliche Werte im Systementwurf umzusetzen. Ein Beispiel sind die „Creative-Commons"-Lizenzen, die aus den Lizenzierungsdetails diverser nationaler Urheberrechte allgemein verständliche Formulierungen extrahieren und juristischen Laiennutzern zur Verfügung stellen [9, 15]. Fehlen einfache rechtliche Vorgaben, besteht die Gefahr, dass Entwickler rechtliche Vorschriften ignorieren oder eigene Interpretationen implementieren.

Anreize zur Durchsetzung von gewolltem Verhalten

Damit Endnutzer Systeme im Sinne des Systementwicklers bzw. -betreibers nutzen, müssen oft passende Anreizmechanismen eingebaut werden. Deren Analyse liegt im Feld der Institutionenforschung. Dies ist insbesondere bei Plattformen zur Kooperation der Fall, z. B. bei sozialen Netzwerken oder auf elektronischen Märkten. Häufig enthalten diese Systeme suboptimale Anreizmechanismen, wie dies die vielen „Trittbrettfahrer" in P2P-Netzwerken ebenso wie die Zurückhaltung von Informationen in Supply-Chain-Managementsystemen verdeutlichen. So können selbst technisch ausgereifte Systeme durch fehlerhafte Anreizausrichtung die Organisationsziele verfehlen. Als Lösung bieten sich einerseits monetäre Anreize zur Ressourcennutzung und -bereitstellung an (z. B. beim Grid Computing [17]). Andererseits sieht man am Beispiel von Open-Source-Projekten, offenen Teilnahmeplattformen (z. B. Wikis) oder sozialen Netzwerken, dass der Leistungsanreiz oft durch eine Rangliste der Leistungen bzw. Beiträge gesteuert wird anstatt durch monetäre Anreize [30].

Während der Entwicklungs- und Nutzungsphase einer Software können Anreizformen im Sinne des „Incentive Engineering" in die Software integriert werden, um die intendierte Nutzung oder gesellschaftlich wünschenswerte Ergebnisse zu erreichen. Ziel ist es, die Softwarenutzung mit der Anreizgestaltung zu koppeln. Voraussetzung für ein anreizkompatibles Software-Engineering ist allerdings ein Verständnis menschlichen Verhaltens, das sich aus den Verhaltenstheorien der Ökonomie, Psychologie, Soziologie oder Politikwissenschaften ableiten lässt.

Zusammenfassung

Fasst man Softwareanwendungen als Institution auf, bietet sich die Institutionenforschung zur Analyse des Sachverhalts und zur Ableitung von Gestaltungsoptionen an. Die Institutionenforschung liefert den Technikwissenschaften dazu vielfältige Methoden und Erkenntnisse z. B. aus den Wirtschafts-, Rechts- und Politikwissenschaft und eröffnet eine mehrdimensionale Perspektive auf Anforderungen an die Softwareentwicklung und die Verfahren hierzu. Der Artikel bietet eine Reihe von Beispielen dafür an, dass Softwaregestaltung verschiedenste gesellschaftliche Akteure und Ebenen betrifft und von ihnen beeinflusst wird. Daher kann sie im Hinblick auf ihren institutionellen Charakter auch nicht ausschließlich mit den Methoden und Erkenntnissen einer Leitdisziplin allein untersucht werden.

Literatur

1. Agrawal R, Kiernan J, Srikant R, Xu Y (2002) Hippocratic databases. In: Very Large Data Base. Proceedings of the 28th International Conference on Very Large Data Bases, August 20–23, 2002, pp 143–154, Hong Kong, China
2. Bizer J (2007) Sieben Goldene Regeln des Datenschutzes. Datenschutz und Datensicherheit – DuD 31(5):350–356
3. Böhm K, Buchmann E (2007) Free riding-aware forwarding in Content-Addressable Networks. VLDB J 16(4):463–482
4. Grimmelmann J (2005) Regulation by Software. Yale Law J 114:1721–1758
5. Haug S, Weber K (2002) Kaufen, Tauschen, Teilen. Musik im Internet. Peter Lang, Frankfurt am Main
6. Helberger N (2006) Code and (intellectual) property. In: Dommering E et al. (eds) Coding Regulation. Essays on the Normative Role of Information Technology. In-

formation Technology and Law Series, No. 12, pp 205–248. T.M.C. Asser Press, The Hague
7. Hodgson G (2006) What Are Institutions? J Econ Issues 40(1):1–25
8. Ishii K (2005) Code Governance. "Code" as Regulation in a Self-governed Internet Application from a Computer Science Perspective. Dissertation. Fakultät IV – Elektrotechnik und Informatik der Technischen Universität Berlin, Technische Universität Berlin, Berlin. http://ishii.de/kei/codegovernance/Ishii2005-CodeGovernance.pdf (Abgerufen am 4. Dezember 2009)
9. Ishii K, Lutterbeck B, Pallas F (2008) Forking, Scratching und Re-Merging. Ein informatischer Blick auf die Rechtsinformatik. Technische Universität Berlin, Lehrstuhl Informatik und Gesellschaft, Berlin. http://ig.cs.tu-berlin.de/ma/bl/ap/2008/IshiiLutterbeckPallas-ForkingItch-scratchingUndRe-merging.EinInformatischerBlickAufDieRechtsinformatik-2008-03-03.pdf (Abgerufen am 4. Dezember 2009)
10. Jahn G, Schramm M, Spiller A (2005) The Reliability of Certification: Quality Labels as a Consumer Policy Tool. J Consumer Policy 28(1):53–73
11. Katyal NK (2002) Architecture as Crime Control. Yale Law J 111:1039–1139
12. Kesan JP, Shah RC (2005) Shaping Code. Harv J Law Technol 18(2):319–399
13. Lessig L (1999) Code and other laws of cyberspace. Basic Books, New York
14. Lessig L (2006) Code Version 2.0. Basic Books, New York
15. Lutterbeck B (2008) Vom „empirischen" zum „generischen" Recht – der Beitrag der Institutionenökonomik. Beitrag für den Workshop „Software als Institution", veranstaltet vom Karlsruhe Institute of Technology (KIT), 12. Dezember 2008, Karlsruhe. Technische Universität Berlin, Berlin. http://ig.cs.tu-berlin.de/ma/bl/ap/2008/BL-VomempirischenZumgenerischenRechtDerBeitragDerInstitutionenoekonomik-2008-12-30.pdf (Abgerufen am 4. Dezember 2009)
16. Ménard C, Shirley MM (2005) Introduction. In: Ménard C et al. (eds) Handbook of New Institutional Economics. Springer, Berlin Heidelberg, pp 1–18
17. Neumann D, Holtmann C, Orwat C (2006) Grid-Economics. Wirtschaftsinformatik 48(3):206–209
18. North DC (1991) Institutions. J Econ Perspectives 5(1):97–112
19. North DC (1992) Institutionen, institutioneller Wandel und Wirtschaftsleistung. Die Einheit der Gesellschaftswissenschaften, Bd. 76. Mohr, Tübingen
20. North DC (2005) Institutions and the Performance of Economies Over Time. In: Ménard C et al. (eds) Handbook of New Institutional Economics. Springer, Berlin Heidelberg, pp 21–30
21. OECD (1980) OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. Organisation for Economic Co-operation and Development, Paris
22. Ostrom E (2005) Doing Institutional Analysis: Digging Deeper than Markets and Hierarchies. In: Ménard C et al. (eds) Handbook of New Institutional Economics. Springer, Berlin Heidelberg, pp 819–848
23. Pallas F (2008) Simple Regeln für komplexe Strukturen: Was die Informatik von der NIÖ lernen kann. Diskussionspapier zum Workshop „Software als Institution", Karlsruhe Institute of Technology. Technische Universität Berlin, Lehrstuhl Informatik und Gesellschaft, Berlin. http://ig.cs.tu-berlin.de/ma/fp/ap/2008/Pallas-SimpleRegelnFuerKomplexeStrukturenWasDieInformatikVonDerNiOeLernenKanntext-2008-12-12.pdf (Abgerufen am 4. Dezember 2009)
24. Radin MJ (2004) Regulation by Contract, Regulation by Machine. J Institut Theoret Econ – Z gesamte Staatswiss 160(1):142–156
25. Reidenberg JR (1998) Lex Informatica: The Formulation of Information Policy Rules Through Technology. Texas Law Rev 76(3):553–584
26. Richter R, Furubotn E (1996) Neue Institutionenökonomik: eine Einführung und kritische Würdigung. Neue ökonomische Grundrisse. Mohr, Tübingen
27. Richter R (2005) The New Institutional Economics: Its Start, its Meaning, its Prospects. Eur Bus Organ Law Rev 6(2):161–200
28. Roßnagel A (2005) Modernisierung des Datenschutzrechts für einen Welt allgegenwärtiger Datenverarbeitung. Multimedia Recht 8(2):71–75
29. Samuelson P (2003) DRM {and, or, vs.} the law. Commun ACM 46(4):41–45
30. Sauer RM (2007) Why develop open-source software? The role of non-pecuniary benefits, monetary rewards, and open-source licence type. Oxford Rev Econ Policy 23(4):605–619
31. Shah RC, Kesan JP (2003) Manipulating the governance characteristics of code. Info 5(4):3–9
32. Wagner RP (2005) On Software Regulation. Southern California Law Rev 78:457–520
33. Werle R, Iversen EJ (2006) Promoting Legitimacy in Technical Standardization. Sci Technol Innovation Stud 2(2):19–39. http://www.sti-studies.de/fileadmin/articles/iversen-werle-230306.pdf (Abgerufen am 4. Dezember 2009)
34. Williamson OE (1987) The Economic Institutions of Capitalism. Free Press, New York
35. Zippelius R (1971) Einführung in die juristische Methodenlehre, 2. Aufl. Beck, München