Simple Regeln für komplexe Strukturen: Was die Informatik von der NIÖ lernen kann

Diskussionspapier zum Workshop "Software als Institution", Karlsruhe Institute of Technology – Version 0.9 –

Frank Pallas

12. Dezember 2008

Unter der zumindest plausibel erscheinenden Annahme, dass Software unter bestimmten Umständen als Institution anzusehen ist, stellt der vorliegende Beitrag die Frage nach der Entstehung eben solcher Institutionen. Aktuelle Entwicklungen hin zu serviceorientierten Architekturen (SOA), Mashups und ähnlichem legen die Vermutung nahe, dass neben den etablierten, grundsätzlich "top-down" funktionierenden Verfahren zukünftig "bottom-up"-Ansätze der Softwareerstellung immer mehr an Bedeutung gewinnen werden. Es stellt sich dann allerdings die Frage, wie sich erreichen lässt, dass die "richtigen" (Software-)Institutionen entstehen. Zur Beantwortung dieser Frage kann sich die "Neue Institutionenökonomik (NIÖ)" möglicherweise als hilfreich erweisen.

1 Einführung

Die Sinnhaftigkeit eines Workshops zu Thema "Software als Institution" erschließt sich nahezu umgehend. Software kann in bestimmten Fällen kodifizierte Verhaltensregeln beinhalten, die zudem durch die Software selbst durchgesetzt werden. Bestimmte Arten von Software entsprechen damit fraglos der üblichen Institutionendefinition, wie sie sich bspw. bei Voigt (2002, S. 34) findet: Institutionen sind hiernach *"allgemein bekannte Regeln, mit deren Hilfe wiederkehrende Interaktionssituationen strukturiert werden und die mit einem Durchsetzungsmechanismus bewehrt sind."*

Die Informatik und mit ihr "die Informatiker" befassen sich vornehmlich mit der Erstellung von Software und damit letztendlich auch mit der Gestaltung

von Institutionen im hier betrachteten Sinne. Es erscheint daher durchaus angebracht, den Entstehungsprozess eben dieser Institutionen zumindest auch aus *informatischer* Sicht zu betrachten, um daraus Rückschlüsse für institutionentheoretische Überlegungen zu ziehen. Der folgende Abschnitt stellt dazu ausgewählte Aspekte der etablierten Sicht auf die Softwareentwicklung kurz vor und setzt diese in Relation zur Neuen Institutionenökonomik. Abschnitt 3 stellt dem einige derzeit an Bedeutung gewinnende, alternative informatische Entwicklungs- und Strukturparadigmen gegenüber und arbeitet die Unterschiede zum etablierten Modell heraus. In Abschnitt 4 werden (mögliche) Implikationen für die institutionentheoretische Sicht auf Software und durch Software geprägte Systeme herausgearbeitet.

Angesichts des noch frühen Stadiums der Forschungen in diesem Bereich müssen die Überlegungen gezwungenermaßen von gewissen Unsicherheiten und Unklarheiten geprägt sein. Dennoch lassen sich diverse Ansatzpunkte für zukünftige Forschungen im Gebiet der "Software-Institutionen" identifizieren.

2 Software-Engineering, Software-Institutionen und der Leviathan

Grundsätzlich stellt Softwareentwicklung einen Kooperationsprozess zwischen unterschiedlichen Individuen dar. Für die notwendige Kooperation wiederum kennt die Neue Institutionenökonomik zwei *grundsätzliche* Modelle, zwischen denen jedoch Abstufungen unterschiedlichen Grades möglich sind: Märkte und Hierarchien.[1] Beide Modelle haben jeweils Vor- und Nachteile, die bereits Gegenstand ausführlicher ökonomischer Diskussionen sind.

In dieser Zweiteilung entsprechen die klassischen informatischen Methoden des Software-Engineering primär der hierarchischen, auf bewusster und strukturierter Planung basierenden Vorgehensweise. So steht nach herrschender Lehrmeinung am Anfang einer jeden Softwareentwicklung die Anforderungsanalyse, gefolgt von Systementwurf, tatsächlicher Implementierung, Evaluierung und schließlich Pflege der erstellten Software. Auf die besonderen Spezifika der unterschiedlichen Vorgehensmodelle soll hier nicht genauer eingegangen werden. Auch wenn sich Wasserfallmodell, Spiralmodell, V-Modell usw. zum Teil gravierend unterscheiden, verfolgen sie dennoch allesamt einen im Wesentlichen planerischen Ansatz und gehen grundsätzlich davon aus, dass Software erstellt wird, um vergleichsweise wohldefinierte Ziele zu erreichen.[2] Unerheblich ist dabei, ob es sich beim zu erstellenden System um eine E-Government-Anwendung, eine Plattform für eine Online-Community oder einen Kopierschutzmechanismus handelt. Die Ziele sind bekannt und mehr

[1] So bspw. Williamson (1975, 1985). Aus etwas anderer Perspektive auch Malone (2004).

[2] Dies gilt im Übrigen auch für modernere "agile" Methoden. Auch wenn die Vorgehensweise hier weniger formal und wohlstrukturiert ist, sind die zu verfolgenden Ziele üblicherweise auch hier vorgegeben und vergleichsweise klar formuliert.

oder weniger klar definiert und die Software dient "lediglich" der Erreichung eben dieser Ziele.

Aus institutionentheoretischer Sicht heißt dies, dass Software-Institutionen in einem grundsätzlich planerischem Prozess *erstellt* werden, bei dem das zu erreichende Verhalten der von der Software-Institution beeinflussten Individuen als *Ziel* vorgegeben ist. Eine E-Government-Anwendung, die ausschließlich vorab definierte Handlungen erlaubt oder den Handlungsspielraum der Nutzer auf einen vorab spezifizierten Bereich beschränkt, ist also eine Institution, die mit dem bewussten Ziel *erstellt* wird, das Verhalten von Individuen in *vorbestimmter* Art und Weise zu beeinflussen. Ein Kopierschutzverfahren wird ebenso mit der Maßgabe entwickelt, bestimmte, klar formulierte Regeln (was ist erlaubt, was nicht?) durchzusetzen und stellt somit – in Verbindung mit den Regeln selbst – ebenfalls eine bewusst und mit einem definierten Ziel *geschaffene* Institution dar. Um ein bestimmtes Verhalten zu erreichen, werden also neue Institutionen geschaffen oder bestehende verändert. Es werden, um mit Richter und Furubotn (2003, S. 10) zu sprechen, die *"Spielregeln"* modifiziert, was letztendlich zu veränderten Nutzenfunktionen[3] der einzelnen Beteiligten und damit zu verändertem Verhalten führt.

Genau diese Sichtweise spricht auch aus der Einladung zu diesem Workshop: Versteckt in einer geradezu überwältigenden Menge zu klärender Aspekte findet sich dort auf Seite 5 die Anmerkung, dass *"Programmierte Institutionen [...] meist von einem Technologieanbieter oder von einer Entwicklergemeinschaft geschaffen"* werden. Die Entwickler der jeweiligen Software-Institution werden damit zu einer maßgeblichen Instanz für das Verhalten der betroffenen Individuen. Sie verfolgen bestimmte Ziele, gestalten die Institution unter Verwendung oben genannter Methoden des Software-*Engineering* gemäß diesen Zielen und sind dadurch in der Lage, individuelles Verhalten zu erreichen, das den verfolgten Zielen entspricht. Man könnte dann gewissermaßen von *"Institutions-Engineering"* sprechen. Der Software-Gestalter als Leviathan, der an der Spitze eines hierarchischen Prozesses steht, mit den ihm zur Verfügung stehenden Mitteln Macht über individuelles Verhalten ausübt und so – hoffentlich – "alles zum Guten wendet".[4]

[3] Genau genommen ist auch das Nichtbeachten einer mit technischen Mitteln durchgesetzten Regel nur in den allerseltensten Fällen vollkommen unmöglich. Ein mittels eines DRM-Systems durchgesetztes Kopierverbot lässt sich bspw. durch das Umgehen des jeweiligen Systems (bspw. durch Entschlüsselung oder durch Ausnutzung der analogen Lücke) prinzipiell sehr wohl missachten. Allerdings verursacht ein solches Umgehen Kosten, die den daraus entstehenden erwarteten Nutzen oftmals übersteigen, wodurch die Nichtbeachtung ökonomisch "unvernünftig" würde. So wäre es im DRM-Beispiel oftmals günstiger, den gesicherten Inhalt legal zu erwerben anstatt den für die Umgehung notwendigen Aufwand auf sich zu nehmen.

[4] Man beachte hierbei das aktuelle Buch von Joel Spolsky (2008). Dessen Titel ziert *tatsächlich* das allseits bekannte Leviathan-Frontispiz von Hobbes (1651). Vermutlich ist aber weder Spolsky noch seinem Verleger gänzlich bewusst, was sie damit zum Ausdruck bringen. Sie selbst begründen die Wahl des Leviathan-Titels folgendermaßen: *"[W]e pay hommage to Hobbes's* The Leviathan *[...] where the giant is made up of lots of individuals, because [we] felt this was not a bad metaphor*

(Staatliche) Regulierung wiederum dient in diesem Kontext dazu, Einfluss auf die Art und Weise der so geschaffenen Institutionen zu nehmen, um ein für die Allgemeinheit "wünschenswertes" Ergebnis zu erreichen. Diese Regulierung kann dabei auf unterschiedlichsten Wegen realisiert werden. In Bereichen wie "E-Government" oder "E-Health" beispielsweise tritt der Staat selbst als Initiator oder Urheber von Software-Institutionen (JobCard bzw. ELENA, A2LL , eGK-Infrastruktur, etc.) in Erscheinung und spezifiziert das durch die Institution zu erreichende Verhalten explizit. In anderen Bereichen wie dem Datenschutz in sozialen Netzwerken oder dem Spannungsfeld zwischen Urheberrecht und wissenschaftlicher Lehre wiederum setzt staatliche Regulierung lediglich mehr oder weniger enge Grenzen für die mittels Software-Institutionen realisierte Beeinflussung individuellen Verhaltens. In beiden Fällen jedoch setzt die Regulierung *am Beginn* des hierarchischen, zielgerichteten Prozesses des *"Institutions-Engineering"* an: Bei den verfolgten Zielen. Entweder sie definiert die von der Institution zu erreichenden Ziele (nahezu) vollständig selbst oder aber sie beeinflusst die Zielsetzungen nichtstaatlicher Institutionen-Gestalter durch Ver- und Gebote oder andere Anreizmechanismen.

Für den klassischen, hierarchisch-planerischen und zielgerichteten Prozess der Gestaltung von Software-Institutionen erscheint ein solches Vorgehen durchaus plausibel. Wie im folgenden Abschnitt dargelegt wird, stehen dem aber zunehmend alternative Entwicklungsmodi gegenüber, in denen Software-Institutionen auf andere Art und Weise entstehen. Neben weiteren Aspekten wird dies auch Auswirkungen für die (staatliche?) Beeinflussung solcher Institutionen haben müssen.

Die Erstellung von Software wird bisher vor allem als "top-down" gerichteter, hierarchisch-planerischer Prozess betrachtet. Dies gilt selbstverständlich auch dann, wenn man Software als Institution begreift.

3 SOA, Mashups und andere Phänomene

Neben den genannten, sowohl in Erstellung als auch in Wirkweise dem "top-down"-Prinzip folgenden (Software-)Institutionen aus den Bereichen E-Government, E-Health und DRM erwähnt die Einladung allerdings auch den Bereich des "Ubiquitous Computing". Zwei weitere, besonders aktuelle informatische Trends erwähnt die Einladung jedoch nicht: serviceorientierte Architekturen (SOA) und Mashups, also "Webseiten" die durch Verknüpfung bereits bestehender Webseiten und -dienste und/oder durch deren Anreicherung entstehen. Alle drei Fälle unterscheiden sich in einem Punkt signifikant von der klassischen Informatik-Sicht auf die Erstellung von Software (-Institutionen):

for how programming is done: individuals building something gigantic—but individuals are the key." (Spolsky, 2008, S. xi)

Die Entwicklung erfolgt nicht von oben nach unten und dient zudem in weiten Teilen auch keinem vorab bekannten und wohlspezifizierten Ziel. Vielmehr beginnt in den genannten Fällen die "Entwicklung" von Software "unten", bei den Basisdiensten und entwickelt sich durch Verknüpfung bereits existierender Basisdienste "aufwärts".

So folgt die Idee, alle einmal erstellten Objekte mit einem eindeutigen, per RFID auslesbaren Identifizierer zu versehen, keinem explizit formulierten Ziel im Sinne einer bewusst angestrebten Verhaltensbeeinflussung. Der Gedanke ist vielmehr, eine *Infrastruktur* zu schaffen, auf Basis derer dann eine Vielzahl unterschiedlichster und vorab noch *nicht* bekannter Dienste und Anwendungen realisiert werden kann.[5] Der ebenfalls vorgesehene Object Naming Service (ONS) wiederum basiert auf einer solchen, grundlegenden Nummeriertheit von Dingen, reichert diese um zusätzliche Funktionalität an und stellt seinerseits ebenfalls einen Basisdienst zur Erstellung von Diensten und Anwendungen auf "höheren" Ebenen dar. Anders als im oben erwähnten, klassischen Software-Engineering erfolgt die Entwicklung hier also nicht top-down sondern bottom-up: Zuerst werden grundlegende Basisdienste erstellt, die dann zur Erstellung "höherwertiger", also komplexerer und näher zum Individuum befindlicher Dienste und Anwendungen verwendet werden. Die Funktionalität dieser höheren Dienste und Anwendungen wird dabei zwar maßgeblich von den existierenden und verwendbaren Basisdiensten beeinflusst, sie wird aber nicht von diesen *bestimmt*. Allein aus der Funktionalität der existierenden Basisdienste lassen sich keine tragfähigen Aussagen über die Eigenschaften "zu erwartender" höherer Dienste ableiten.

Dies gilt natürlich nicht nur für den Themenkomplex RFID sondern auch für andere, durch bottom-up Entwicklung geprägte Bereiche. So existiert beispielsweise eine Vielzahl von Internetseiten, die auf Basis von Google Maps die unterschiedlichsten Angebote realisieren. Die Spanne reicht dabei von Nachbar-Diffarmierungsseiten wie rottenneighbor.com und lokalisierten Verzeichnissen von Sexualstraftätern bis zu innerstädtischen Mitfahrbörsen. Alle diese Seiten basieren letztendlich auf Google Maps, die mittels der von Google dafür bereitgestellten Programmierschnittstelle um zusätzliche Funktionalität und/oder Information angereichert werden. Auch hier gilt, dass es grundsätzlich unmöglich war und ist, allein aus der Existenz und den Eigenschaften von Google Maps fundierte Aussagen über die daraus entstehenden höheren Dienste und Anwendungen abzuleiten.

Als letztes Beispiel seien serviceorientierte Architekturen (SOA) genannt. Auch wenn diese derzeit primär dazu verwendet werden, Lösungen *innerhalb* des bestehenden, hierarchisch-planerischen Paradigmas "lediglich" auf veränderter technologischer Basis zu entwickeln: Der eigentliche Ansatz besteht darin, einmal erstellte Dienste zur Realisierung unterschiedlichster höherer Dienste und Anwendungen zu verwenden. Die zu Grunde liegenden Dienste

[5] Vgl. hierzu auch Pallas (2005, 2009).

sollen dabei austauschbar und grundsätzlich auch über Organisationsgrenzen hinweg miteinander verknüpfbar sein. Auch wenn solche SOA-Dienste heute in erster Linie als "Teilsysteme" anzusehen sind, die im Hinblick auf die durch das "Gesamtsystem" zu erreichenden Ziele – also top-down – entwickelt und gestaltet werden, so wird zukünftig aller Wahrscheinlichkeit nach auch hier der Aspekt der bottom-up gerichteten Neuverknüpfung bereits bestehender Dienste an Bedeutung gewinnen.

Schon anhand dieser drei Beispiele wird deutlich, dass Software zunehmend nach dem bottom-up Paradigma entsteht, bei dem – anders als im klassischen Software-Engineering grundsätzlich angenommen – die explizite und möglichst detaillierte Identifikation der verfolgten und von der Software umzusetzenden Ziele *eben nicht* am Beginn des Entwicklungsprozesses steht. Software entsteht in diesen Fällen auch nicht auf Basis eines möglichst fein spezifizierten Projektplans und auch nicht im Rahmen eines grundsätzlich hierarchischen Prozesses. Natürlich verfolgt der Entwickler auch bei einem solchen, eher orchestrierenden Ansatz immer ein bestimmtes Ziel, anders als im klassischen Ansatz wird dieses aber vor allem auf Basis bereits bestehende Dienste realisiert und ist dadurch mit oftmals deutlich geringerem Aufwand erreichbar, da der größte Anteil der verwendeten Funktionalität bereits *ex ante* besteht. Die Bedeutung der bewussten, expliziten Planung (Zieldefinition, Spezifikation, Systementwurf, Modellierung usw.) nimmt im Zuge dessen deutlich ab.

Gleichzeitig definieren die zu Grunde liegenden Dienste in diesem Modell sehr wohl, was überhaupt realisierbar ist. So sind viele der oben erwähnten Mashups erst durch die Verfügbarkeit von Google Maps *ermöglicht* worden und viele der bereits in der Öffentlichkeit diskutierten Szenarien des Ubiquitous Computing setzen zwingend eine allgemeine Nummeriertheit von Dingen voraus. Diese Beeinflussung des Möglichen ist jedoch alles andere als deterministisch im Sinne einer möglichen Vorhersagbarkeit der auf Basis eines bestimmten Dienstes zukünftig entwickelten Anwendungen. Klassisches, top-down gerichtetes Software-Engineering ist *zielgerichtet*. Bottom-up Entwicklung nicht – oder zumindest deutlich weniger.

Gleichwohl können auch die aus einem bottom-up Prozess hervorgehenden Software-Artefakte Institutionen im hier betrachteten Sinne darstellen. Sie können Verhaltensregeln definieren und / oder durchsetzen. Aufbauend auf Basistechnologien des Ubiquitous Computing lassen sich paternalistische Regelsysteme[6] formulieren und durchsetzen, ohne dass dies das explizite Ziel bei der Entwicklung der Basisdienste unterschiedlicher Ebene gewesen wäre. Google-Maps Mashups wie innerstädtische Mitfahrzentralen können individuelles Verhalten strukturieren und beeinflussen, ohne dass dies Ziel oder vorhersehbare Folge der Bereitstellung von Kartenmaterial und Programmierinterface gewesen wäre. Und auch auf Basis serviceorientierter Architektu-

[6]Siehe hierzu Spiekermann und Pallas (2007)

ren werden durch kreative Verknüpfung bereits bestehender Dienste Lösungen entstehen, die bestimmte Verhaltensoptionen verunmöglichen und / oder zwingend erforderlich machen. Wie aber lassen sich solche und ähnliche Phänomene aus institutionentheoretischer Sicht betrachten? Ein Versuch soll im folgenden Abschnitt unternommen werden.

Abweichend vom etablierten, top-down gerichteten, planerischen und grundsätzlich hierarchischen Paradigma des Software-Engineering entsteht Software zunehmend in bottom-up Prozessen. Sowohl bei serviceorientierten Architekturen als auch bei Mashups oder im Ubiquitous Computing werden Lösungen zunehmend durch Zusammenstellung bestehender Dienste realisiert. Die Gestaltung der Basisdienste erfolgt jedoch weder im Hinblick auf ein bestimmtes und wohldefinertes Ziel noch lässt sich aus der spezifischen Gestaltung eines Basisdienstes fundiert auf mögliche oder "zu erwartende" spätere Anwendungsfälle schließen. Gleichwohl beeinflusst die konkrete Ausgestaltung der Basisdienste die resultierenden Software-Institutionen.

4 Software als Ergebnis menschlichen Handelns, nicht aber menschlichen Entwurfs

Wie oben dargelegt entstehen Software und damit auch Software-Institutionen zunehmend in von unten nach oben gerichteten, nicht-hierarchischen, nicht-planerischen und vor allem nicht-deterministischen Prozessen. Die am Beginn der Entwicklung stehenden Basisdienste werden dabei nicht im Hinblick auf ein spezifiziertes, zu erreichendes Ziel entwickelt sondern dienen als Grundlage für eine Vielzahl darauf aufbauender Dienste und Lösungen. Die hieraus hervorgehenden Dienste und Anwendungen können dabei ebenfalls Software-Institutionen darstellen und müssen dementsprechend – neben den anfangs erwähnten, top-down erstellten Fällen wie E-Government-Applikationen oder DRM-Mechanismen – ebenfalls Gegenstand anzustellender Forschungen unter dem Titel "Software als Institution" sein. Es stellt sich dann die Frage, wie sich solche bottom-up entstehenden Software-Institutionen in den bestehenden Theorie-Korpus der Neuen Institutionenökonomik "einsortieren" lassen.

Ein möglicher Ansatzpunkt besteht darin, solche bottom-up entstehenden und individuelles Verhalten beeinflussenden Software-Artefakte im Sinne F. A. Hayeks (1967) als Institutionen zu verstehen, die zwar *"Ergebnisse menschlichen Handelns, nicht aber menschlichen Entwurfs"* sind. In dem von Hayek betrachteten Modell ergeben sich Institutionen als *"spontane Ordnung"*[7] aus einer Vielzahl individueller und primär von individuellen Interessen geleiteter Handlungen. Die Entstehung solcher Institutionen ist damit gewissermaßen als unbeabsichtigte Nebenwirkung individuellen Handelns zu

[7] Von Hayek (1967, S. 183)

verstehen.

Das eigentlich Interessante an solchen weder geplanten noch anderweitig explizit beabsichtigten Institutionen ist dabei, dass sie sich *sehr wohl* als tragfähige und allgemein akzeptierte Mechanismen zur Strukturierung sozialer Interaktion erweisen können – auch wenn, so Hayek, sich die Jurisprudenz weitgehend an eine Sicht klammere, *"die alle Regeln der Gerechtigkeit als ein Ergebnis bewußter Erfindung oder Planung betrachtet [...]"* (von Hayek, 1967, S. 184). Grundsätzlich lässt sich also, ohne hierauf detaillierter eingehen zu wollen, festhalten, dass soziale Interaktion sowohl von geplanten, erschaffenen oder anderswie "konstruierten" als auch von aus individuellem Verhalten als "spontane Ordnung" entstehenden Institutionen strukturiert werden kann und dass sich ungeplante Institutionen den geplanten gegenüber durchaus durchaus als vorteilhaft erweisen können.

Wenn aber sowohl spontane Ordnungen als auch explizite Planung, sowohl Märkte als auch Hierarchien, in der Neuen Institutionenökonomik ihren Platz als mögliche Paradigmen zur Strukturierung sozialer Interaktion haben, dann mag dies möglicherweise auch für den Bereich der Software-Institutionen gelten. Man könnte dann zu dem Schluss kommen, dass auch in oben beschriebenen, bottom-up gerichteten Prozessen Software-Institutionen entstehen können, die sich als tragfähige und durchaus sinnvolle (was auch immer das heißen mag) Mechanismen zur Strukturierung sozialer Interaktion erweisen. Eine wie auch immer geartete Agenda zur Erforschung von Software-Institutionen dürfte sich dann aber nicht auf die ausschließliche Betrachtung von in hierarchisch-planerischen, "konstruierenden" Prozessen entstehenden Software-Institutionen beschränken sondern müsste auch den Bereich derjenigen Software-Institutionen berücksichtigen, die zwar *"Ergebnis menschlichen Handelns, nicht aber menschlichen Entwurfs"* sind.

Neben anderen Überlegungen stellt sich dann die Frage, ob und wenn ja wie sich ein solcher Prozess der spontanen Entstehung von Software-Institutionen beeinflussen lässt. Zudem wäre möglicherweise zu klären, ob sich aus der grundsätzlichen Möglichkeit der Beeinflussung individuellen Verhalten durch bottom-up entstandene Institutionen auch Implikationen für die Informatik ergeben. Beide Bereiche sollen im folgenden Abschnitt kurz betrachtet werden.

Institutionen sind nicht gezwungenermaßen das Ergebnis bewusster Planung sondern können sich auch als "spontane Ordnung" aus individuellem Handeln ergeben. Die so entstandenen Institutionen können sich zudem als durchaus tragfähig erweisen. Ein wie auch immer geartetes Forschungsgebiet "Software als Institution" wird diesem Umstand Rechnung tragen müssen und auch solche Software-Institutionen zu betrachten haben, die in bottom-up Prozessen und ohne explizite Planung entstanden sind.

5 Implikationen und Forschungsfragen

Wie bereits oben erwähnt findet (staatliche) Regulierung im Bereich der top-down gestalteten Institutionen vor allem am Beginn des Erstellungsprozesses statt, indem die von der zu erstellenden Institution verfolgen Ziele (bspw. das zu erreichende individuelle Verhalten) auf unterschiedliche Weise beeinflusst werden. So kann der Staat beispielsweise die zu erreichenden Ziele entweder nahezu vollständig selbst spezifizieren oder die durch private Institutionen-Gestalter verfolgten Ziele durch Ge- und / oder Verbote beeinflussen. Die Institution wird dann vollends im Hinblick auf die so gesetzten oder beeinflussten Ziele hin gestaltet.

Anders stellt es sich für den Bereich der bottom-up entstehenden Software-Institutionen dar, deren Gestaltungsprozess eben *keinen* expliziten Zielen im Sinne eines durch die Institution zu erreichenden individuellen Verhaltens folgt und die somit auch keinen Ansatzpunkt für eine wie auch immer gearte-te Zielmodifikation durch staatliche oder andere regulierende Akteure bieten. Die "Institutionen-Werdung" von bereits existierenden Basisdiensten kann in diesem Modell allein durch einen einzigen, geradezu plötzlichen Verknüpfungsvorgang geschehen, der schlichtweg nicht vorhersehbar ist. Ist ein solcher Verknüpfungsvorgang erst einmal erfolgt, dann ist die Institution in der Welt und fortan in der Lage, individuelles Verhalten und soziale Interaktionen zu strukturieren. Wie also lässt sich in einem solchen Modell erreichen, dass gesellschaftlich wünschenswerte Institutionen entstehen, nicht wünschenswerte jedoch nicht?

Einen möglichen Ansatzpunkt in diese Richtung könnte die Forderung nach "simplen Regeln" sein, wie sie für nicht-technische Kontexte beispielsweise von Autoren wie Epstein (1995) oder Polanyi (1951) erhoben wird[8]: Komplexe Systeme, so die regelmäßige Aussage, müssten auf simplen und möglichst konstanten Regeln basieren, um den eigentlichen, autonom handelnden Akteuren zu erlauben, im Rahmen dieser einfachen Regeln durch ständige Interaktion zu Ergebnissen zu gelangen, die denen von hochkomplexen und sich ständig verändernden Regelwerken überlegen sind. Für die Regulierung von bottom-up entstehenden Software-Institutionen würde dies bedeuten, dass (staatliche) Regulierung eben nicht versucht, mittels hochkomplexer Vorgaben den unterschiedlichen Interessen aller möglichen Beteiligten möglichst gut gerecht zu werden, sondern dass vielmehr *einfache* Regeln einen Rahmen vorgeben, in dem sich Software-Institutionen als "spontane Ordnungen" nach den üblichen, keinem klar spezifizierten Ziel folgenden bottom-up Vorgehensweisen herausbilden können.

In der Neuen Institutionenökonomik sind einige solche "simple Regeln" oder Prinzipien bekannt, die der marktbasierten – und damit bottom-up erfolgenden – Herausbildung insgesamt wünschenswerter Ergebnisse zumindest för-

[8] Für eine einführende Zusammenfassung siehe auch Zywicki (1998).

derlch sind. Sicherlich zu nennen sind hier die Internalisierung von Externalitäten bzw. die Pflicht zur angemessenen Entschädigung. Sofern die Transaktionskosten hinreichend gering sind, lassen sich auf dieser Basis durch Nutzbarmachung verteilten individuellen Wissens[9] im Rahmen von marktähnlichen Prozessen Ergebnisse erzielen, die für alle Beteiligten vorteilhaft sind.[10] Auf Basis einer solchen, einfachen und grundsätzlichen Regel ließe sich möglicherweise auch die Herausbildung von bottom-up Software-Institutionen dahingehend beeinflussen, dass nur solche Institutionen entstehen, die für alle Beteiligten vorteilhaft sind.[11] Eine andere in Frage kommende "simple Regel" wäre möglicherweise das Prinzip der Reziprozität, das schon heute bspw. in üblichen P2P-Protokollen dafür sorgt, dass kein Free-Riding möglich ist. Sieht man hier die ganze P2P-Applikation als Software-Institution an, dann kann diese Institution nach Belieben verändert, weiterentwickelt oder im Zuge einer kompletten Neuimplementation durch eine alternative Applikation ersetzt werden. So lange das zu Grunde liegende und das Prinzip der Reziprozität durchsetzende Protokoll beibehalten wird, werden die individuellen Beteiligten nur diejenigen Applikationen (und damit Institutionen) wählen, die für sie den größten individuellen Mehrwert bieten. Ähnlich stellt es sich im Übrigen auch für die von Messaging-Applikationen und bei der IP-Telefonie verwendeten Protokolle dar. Möglicherweise müsste man daher auf die Bedeutung der von verteilten Software-Institutionen verwendeten Protokolle zukünftig noch genauer eingehen. Als typischerweise vergleichsweise simple Konstrukte strukturieren diese oftmals individuelles Verhalten und soziale Interaktionen in hoch- und höchstkomplexen "Systemen".

Zudem ergeben sich auch für die Informatik möglicherweise Implikationen aus der Erkenntnis, dass dem bottom-up Prinzip folgende Institutionen grundsätzlich auch zu besseren Ergebnissen führen können als solche, die in einem Prozess hierarchischer, expliziter Planung erstellt wurden. Dies bedeutet möglicherweise, dass die Informatik sich von dem erwähntermaßen in der Lehre vorherrschenden Gedanken verabschieden muss, dass Software-Artefakte *grundsätzlich* in einem wohlstrukturierten, planerischen Prozess erstellt werden müssen oder sollen. Anstatt "Systeme" zu erstellen, deren Wirkung und Funktion klar definiert sind, müssten auch andere Strukturierungs- und Ordnungsparadigmen Einzug halten. Abermals könnten hier möglicherweise die oben genannten "simplen Regeln" relevant sein. Hochkomplexe Protokollspezifikationen, die bereits auf Ebene der Basisdienste versuchen, höhere Dienste und Anwendungen – und damit Institutionen – zu beeinflussen, würden demnach dazu führen, dass sich vorteilhafte Institutionen auf höheren Ebenen möglicherweise gar nicht erst herausbilden.

[9] Vgl. hierzu bspw. von Hayek (1945).
[10] Vgl. hierzu insb. Coase (1960).
[11] Einige grundlegende Gedanken zur Anwendung dieses Prinzips auf den Bereich des Datenschutzes in Ubiquitous-Computing-Umgebungen finden sich bspw. in Pallas (2009).

Unabhängig vom hier diskutierten Entstehungsparadigma (top-down vs. bottom-up) könnte man außerdem auch über Software nachdenken, die nicht ein bestimmtes, vorab als "wünschenswert" identifiziertes Verhalten durchsetzt oder "unerwünschtes" Verhalten unterbindet, sondern anstatt dessen (positive wie negative) Externalitäten individuellen Verhaltens internalisiert.[12] So könnten bspw. in eine Software bzw. ein Protokoll implementierte "Anreize" dafür sorgen, dass knappe Ressourcen (Bandbreite, CPU-Zeit, etc.) möglichst effizient zugewiesen werden. Anstatt zu versuchen, diese Allokation mittels eines detaillierten, zentral spezifizierten Regelwerkes zu realisieren, würde man hierzu möglicherweise den Preismechanismus nutzen. Solche und ähnliche Ordnungsprinzipien könnten zukünftig zumindest in ausgewählten Bereichen an die Stelle des etablierten Paradigmas möglichst vollständiger initialer Spezifikation treten. Ob die Zunft der Informatiker sich damit wird abfinden können, das muss sich allerdings erst noch zeigen.

Es wird deutlich: Eine institutionentheoretische Sicht auf Software eröffnet ein weites, hochinteressantes Feld für zukünftige Forschungen. Die hier primär diskutierte Unterscheidung zwischen top-down "gestalteten" und bottom-up "entstehenden" Software-Institutionen kann hierbei nur ein Baustein unter vielen sein. Zudem sind die hier angestellten Überlegungen selbstverständlich nur allererste, eher ungerichtet-explorative Gedanken als dass sie bereits auch nur Ansätze von wirklichen Antworten geben würden.

Was jedoch klar geworden sein sollte ist, dass Software und damit aller Voraussicht nach auch Software-Institutionen zunehmend im Rahmen des bottom-up Paradigmas entstehen. Es wäre daher geradezu fahrlässig, sich bei der Untersuchung von "Software als Institution" ausschließlich auf die klassischen, in hierarchisch-planerischen Prozessen gestalteten Software-Institutionen zu beschränken. Die Erforschung von dem bottom-up Prinzip folgenden Software-Institutionen wird in einem ersten Schritt vermutlich vor allem empirische Untersuchungen erfordern. Diese mögen vielleicht mühselig sein – notwendig jedoch sind sie allemal.

Literatur

Coase, R. H. (1960). The Problem of Social Cost. *Journal of Law and Economics 3*(1), 1–44. DOI: 10.1086/466560.

Epstein, R. A. (1995). *Simple Rules for a Complex World.* Harvard University Press.

Hobbes, T. (1651). *Leviathan or the Matter, Forme and Power of a Commonwealth Ecclesiastical and Civil.*

[12] Einführend hierzu abermals Coase (1960).

Malone, T. W. (2004). *The Future of Work - How the New Order of Business Will Shape Your Organization, Your Management Style, and Your Life.* Boston: Harvard Business School Press.

Pallas, F. (2005). RFID als Infrastruktur – Von geschlossenen und offenen Systemen. CAST-Forum RFID/Smartcard, Darmstadt. Online: `http://ig.cs.tu-berlin.de/ma/fp/ap/2005/` [09.12.2008].

Pallas, F. (2009). Antwort auf Stefan Klein - 4 Thesen zu RFID, Cookies & Co. Sammelband zur Fachkonferenz "Internetökonomie und Ethik", im Erscheinen.

Polanyi, M. (1998 [1951]). *The Logic of Liberty: Reflections and Rejoinders.* Liberty Fund.

Richter, R. und E. G. Furubotn (2003). *Neue Institutionenökonomik* (dritte Aufl.). Tübingen: Mohr Siebeck.

Spiekermann, S. und F. Pallas (2007). Technologiepaternalismus – Soziale Auswirkungen des Ubiquitous Computing jenseits von Privatsphäre. In: F. Mattern (Hrsg.), *Die Informatisierung des Alltags – Leben in smarten Umgebungen*, S. 311–325. Berlin, Heidelberg, New York: Springer Verlag.

Spolsky, J. (2008). *More Joel on Software: Further Thoughts on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers and Managers and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity.* Berkeley: Apress.

Voigt, S. (2002). *Institutionenökonomik.* München: Utb / W. Fink.

von Hayek, F. A. (1945). The Use of Knowledge in Society. *The American Economic Review 35*(4), 519–530.

von Hayek, F. A. (1967). Die Ergebnisse menschlichen Handelns, aber nicht menschlichen Entwurfs. In: *Rechtsordnung und Handelnsordnung – Aufsätze zur Ordnungsökonomik*, S. 178–189. Tübingen: Mohr Siebeck.

Williamson, O. E. (1975). *Markets and Hierarchies – Analysis and Antitrust Implications.* New York: Free Press.

Williamson, O. E. (1985). *The Economic Institutions of Capitalism.* New York: Free Press.

Zywicki, T. J. (1998). Epstein and Polanyi on Simple Rules, Complex Systems, and Decentralization. *Constitutional Political Economy 9*(2), 143–150. DOI: 10.1023/A:1009012111547.