Code, Software und Subjekt. Zur Relevanz der Critical Software Studies für ein nicht-reduktionistisches Verständnis „digitaler Bildung“

Benjamin Jörissen und Dan Verständig

Preprint. Erschienen als:

Jörissen, B., & Verständig, D. (2016). Code, Software und Subjekt. Zur Relevanz der Critical Software Studies für ein nicht-reduktionistisches Verständnis „digitaler Bildung“. In R. Biermann & D. Verständig (Hrsg.), *Das umkämpfte Netz*. (Bde. 1–35, S. 37–50). Wiesbaden: VS Verlag.

Einleitung

„Digitale Bildung“ entwickelt sich in den letzten Jahren zum bildungspolitischen Megathema. „Digitalisierung“ wird dabei oft weniger als ereignishafter, tiefgreifender historischer Prozess verstanden denn als technisch-informationaler (und politischer-administrativer) Gestaltungsauftrag, der möglichst breit und effizient in entsprechenden Verfügungswissen und entsprechende Praxiskompetenzen umzusetzen sei (vgl. KMK 2016). Solche (im politischen Jargon vielleicht unvermeidbaren) instrumentalistischen Perspektiven suggerieren eine Form von Machbarkeit, die im Hinblick auf die erheblichen Umbrüche auf ökonomischer, infrastruktureller, kultureller, sozialer und individueller Ebene fatal, zumal als pädagogischer Umsetzungsauftrag problematisch wäre. Algorithmen, Protokolle, Datenstrukturen und Datenbestände greifen mehr und mehr in gesellschaftliche, individuelle und kulturelle Prozesse ein; sie schreiben sich in Architekturen, Infrastrukturen und Materialitäten des alltäglichen Lebens ein, stellen Kontexte für Kommunikation, Artikulation, Kreativität, Vernetzung bereits und nehmen somit einerseits einen integralen Bestandteil bei der Frage nach der Herstellung von Orientierungsrahmen ein; andererseits verändern Sie die Koordinaten im Hinblick auf Fragen der der Selbstbestimmung und Subjektivation.

Die hieraus erwachsenden Problemlagen werden recht schnell deutlich, wenn man auf das Verhältnis von Mensch, Sozialität und Technologie fokussiert. Nicht etwa weil die durch eine Interaktion gemachten Erfahrungen von den Fähigkeiten der User abhängen, sondern vielmehr dadurch, dass der technologische Rahmen entweder bestimmte Formen der Interaktion unterstützt oder diese gerade nicht ermöglicht. Damit entwickelt die Ebene der Software, die nur indirekt

anhand ihrer performativen Effekten erfahrbar wird, eine besondere Wirkmacht, die von den implizit bleibenden unterliegenden Regeln der Programmierung abhängig sind.
Die damit angesprochene Ebene bezieht sich auf das Design von Software in einem umfassenden, und bildungstheoretisch relevanten Sinn (vgl. Jörissen 2015). Design hat sich nicht zufällig, von der Erziehungswissenschaft eher unbemerkt, in den letzten Jahren als diskursive Schlüsselkategorie nicht nur ästhetischer, sondern insbesondere wissens- und machttheoretischer Diskurse formiert (Mareis, Held & Joost 2013; Tungstall 2013; Moebius & Prinz 2012; Mareis 2011). Dieser Diskurs ist zum einen durch seine erziehungs- und subjekttheoretischen Bezüge allgemeinpädagogisch relevant, zum anderen aber durch seinen direkten Zusammenhang mit dem Design von Code, Software und digitalen Architekturen unmittelbar für den Medienbildungsdiskurs von einiger Bedeutung. Aus dieser Perspektive resultiert, wie wir nachfolgend aufzeigen wollen, die Forderung nach einer systematischen Auseinandersetzung mit Code- und Softwarestrukturen im Kontext digitaler Bildung und Kultur. Es geht darum, den Blick unter die digitale Motorhaube zu wagen, um nicht nur Fragen der Produktivität und Gestaltbarkeit zu beantworten, sondern diese vor allem auch hinsichtlich sozialer Praktiken zu beurteilen. Doch wie lässt sich Code innerhalb dieser komplexen medialen Zusammenhänge denken; welche Implikationen ergeben sich hieraus für eine diesen Dynamiken angemessene Vorstellung von Bildung?

Der Beitrag geht diesen Fragen nach und diskutiert die Bedeutung von Code-, Software- und Netzwerkarchitekturen in Bezug auf Subjektivation und Bildung im Horizont digitaler Medialität. Dies erfolgt in drei Schritten. Zunächst möchten wir grundlegend auf das Verhältnis von Code, Software und Subjekt eingehen. Dazu verweisen wir zunächst auf die ausgeprägte Tradition numeralisierten Denkens und die damit verknüpften Konsequenzen für rechenbasierte Systeme sowie auf die impliziten Mechanismen sozialer Aushandlung in Abhängigkeit zu digitalen Architekturen. Dabei werden subjekttheoretische Bezüge relevant, die wir entlang ausgewählter Beispiele diskutieren. Vor diesem Hintergrund kann die Frage nach den Kontroll- und Herrschaftsstrukturen aufgegriffen und in Bezug auf die Genese von Code und Software unter Berücksichtigung der Critical Code Studies verhandelt werden.

Code, Gesetz und Subjekt

Fragt man nach Code und Software[1] im Verhältnis zum Subjekt so werden die unterschiedlichen Bedeutungsebenen recht schnell sichtbar. Eine relativ häufig rezipierte und umfassend ausgearbeitete Perspektive, ist die von Lawrence Lessig (1999, 2000, 2006). In seinem vor mehr als 16 Jahren veröffentlichten Band „Code and Other Laws of Cyberspace“ entwirft er, ausgehend von einer rechtswissenschaftlichen Betrachtung und unter Bezug auf Reidenberg (1996), die These, dass der Code eine tiefgreifende Wirkmacht auf moderne Gesellschaften hat, indem er bestimmte Optionen vorsieht und andere Möglichkeiten dafür ausschließt. Auf diese Weise bestimmt Code Wahrnehmungsweisen, bzw. – wie man aktualisierend ergänzen kann – stellt Code selbst eine genuine Wahrnehmungsweise dar (Parisi 2013), die paradigmatisch auf andere Wahrnehmungsweisen wirkt. Im Sinne dieses Paradigmatischen verwendet Lessig die Gesetzesmetapher:

> „The code regulates. It implements values, or not. It enables freedoms, or disables them. It protects privacy, or promotes monitoring. People choose how the code does these things. People write the code. Thus the choice is not whether people will decide how cyberspace regulates. People—coders—will. The only choice is whether we collectively will have a role in their choice—and thus in determining how these values regulate—or whether collectively we will allow the coders to select our values for us.“ (Lessig 2000)[2]

Somit ergeben sich aus dem Prozess der Soft- und Hardwaregestaltung neue machttheoretische Relationen, denn denjenigen, die den Code schreiben und gestalten, kommt somit nicht nur eine handwerklich-ingenieurmäßige Verantwortung zu, wie sie bei der Auslieferung eines finalen Produkts im Vordergrund steht, sondern vor allem auch eine kritisch-ethische Verantwortlichkeit, was die Gestaltung und Architektur soft- und hardwarebasierter Systeme angeht, auf die wir uns in alltäglichen Abläufen verlassen, im Hinblick also auf implizite hegemoniale Effekte:

---

[1] Wir verstehen, kurz gefasst, unter “Code” den z.B. in Form von Programmiersprachen vorliegenden symbolischen Ausdruck, während “Software” die kompilierte, an Hardware angepasste und immer in nur in Hardwarestrukturen verortete, performative Informationsstruktur darstellt. Im letztgenannten Sinn von “Software” hebt Lawrence Lessig hervor, dass immer Soft- *und* Hardware in den Blick genommen werden müssen. Es geht also nicht nur um die Ebene der Repräsentation und somit einzelner Anwendungen, sondern vielmehr auch um die zu Grunde liegende technologische Infrastruktur des Netzes (vgl. hierzu Lessig 1999, S.6). Indem Software immer von der Hardware abhängt, kann eine solche Trennung von Soft- und Hardware in dieser Perspektive ohnehin nur auf analytischer Ebene stattfinden, da Software immer Software-in-Hardware ist.

[2] In Anlehnung an Chun (2011) muss man an dieser Stelle vorwegnehmend den terminologischen Einwand erheben, dass das Gesetz als Text – im Sinne juristischer Gesetze, nicht Naturgesetze – immer auslegungsbedürftig ist, hingegen der “para-digmatische” Charakter von Software, den Lessig eigentlich referenziert, eher dem “logos” im Sinne der platonisch-ideenhaften “Vorschrift” entspricht als dem grundsätzlich deliberativ geöffneten menschlichen Gesetz. Wir gehen weiter unten auf diese Differenzierungsnotwendigkeit ein.

> „As the world is now, code writers are increasingly lawmakers. They determine what the defaults of the Internet will be; whether privacy will be protected; the degree to which anonymity will be allowed; the extent to which access will be guaranteed.“ (Lessig 2006, S. 79)

Indem sich eine Komplexitätssteigerung in diesen Abläufen abzeichnen lässt, verschieben sich auch die Machtverhältnisse hinsichtlich der digitalen Sphäre. Ubiquitär vernetzte rechenbasierte Systeme nehmen nicht nur die Rolle von Entscheidungsträgern ein, sie werden im Sinne der Komplexitätsreduktion und verstärkten Einbettung von automatisierten Lösungen zugleich zu epistemischen Akteuren, die als lernende Netzwerke, ähnlich neurobiologischen Lernprozessen, hyperkomplexe Informationsverarbeitung[3] betreiben und somit zum Akteur im gesellschaftlichen Wandel werden. David Golumbia folgend wird mit dem Computer ein Akteur ins Spiel gebracht, der das Hobbes'sche Verhältnis von Souverän und Machtaneignung bestärkt:

> „The computer encourages a Hobbesian conception of this political relation: one is either the person who makes and gives orders (the sovereign), or one follows orders. There is no room in this picture for exactly the kind of distributed sovereignty on which democracy itself would seem to be predicated“ (Golumbia 2009, S. 224).

Dies impliziert, dass eben jene Unberechenbarkeit sozialer Aushandlungen auch ein System unterlaufen und somit Teilsysteme – mit möglicherweise ganz anderen sozialen Praktiken – entwerfen kann. Gleichwohl sind die digitalen Technologien immer auch ein Produkt der sozialen Interaktion, erst eingebettet in sozio-kulturelle Praktiken entfalten sie ihre tatsächliche Wirkmacht. Nicht zuletzt aus diesem Grund werden bei Facebook auch Menschen eingesetzt, um die algorithmischen Abläufe zu prüfen und den, durch die Nutzenden, geteilten Inhalt hinsichtlich der sozialen Angemessenheit zu beurteilen (vgl. Pasquale 2015, S.191). In Anlehnung an Hobbes muss nun gefragt werden, wer die Adressaten der Befehle des Souveräns sind und wie sich diese Strukturen reproduzieren. Wenn der Rechner Befehle ausführt, die im Code festgeschrieben sind und diese dann wiederum durch menschliche Akteure beobachtet werden, verändert sich das Gefüge der Funktionszuweisungen und Ausführung von Prozessen insofern, als dass emergente Effekte innerhalb der Nutzung entstehen, die nicht festgeschrieben sind. Diese Emergenz, die man entlang

[3] Wir spielen damit nicht nur auf KI und ihre Leistungen auf dem Gebiet komplexer Mustererkennung an, sondern auch auf einfachere Software, die mithilfe massiver Datenbestände komplexe quasi-hermeutische Prozesse simuliert, wie es etwa bei gängigen Online-Übersetzungsangeboten der Fall ist.

der sozialen Medien besonders deutlich sehen kann (Münker 2009), gilt aber auch für die zugrunde liegenden Technologie-Schichten[4].
Doch Code strukturiert längst nicht nur die digitalen Räume, sondern entfaltet seine konstitutive Kraft ebenso - und das in historischer Perspektive wesentlich ausgeprägter - auch im materiellen Raum. Der Check-In Bereich eines Flughafens beispielsweise würde ohne Software und Code nicht funktionieren, da hier sehr dezidiert auf Rechensysteme zurückgegriffen wird, die eine Ordnung herstellen und somit erst den Raum definieren. Rob Kitchin und Martin Dodge (2011) bezeichnen diese Synergie als „code/space“ und zeigen entlang unterschiedlicher Studien a) wie tiefgreifend Code und Software in unsere Lebensbereiche durchdringen und b) wie sich dies auf die Gestaltung von Räumen in einer wechselseitigen auf gesellschaftliche Prozesse und die Gestaltung der Umwelt auswirkt. Derlei - von Rechnern geprägte - Räume sind so entworfen, dass sie so gesehen erst dann „funktionieren“, wenn die Software beziehungsweise der zu Grunde liegende Code reibungslos integriert wird. Code und Software sind längst schon in vielen Lebensbereichen auch „ohne Strom und Akku“ appräsent. Es reicht aus dieser Perspektive nicht aus, Digitalität anhand ihrer gegenständlichen Sichtbarkeiten (pädagogisch oder bildungstheoretisch) zu betrachten, vielmehr muss die Frage der Infrastrukturen und der Vernetzung technologischer, sozialer und materieller Akteure systematisch in den Blick genommen werden.

Damit wurden also bereits einige Facetten angesprochen, die auf die tiefe Durchdringung von Code und Algorithmen in modernen Gesellschaften verweisen. Im Anschluss hieran wenden wir uns der Frage zu, wie sich der Codebegriff hinsichtlich subjekttheoretischer Bezüge und sodann im Kontext bildungstheoretischer Fragestellungen einordnen lässt.

Critical Code Studies als Zugang für subjekttheoretische Betrachtungsweisen von Code

Während der Codebegriff bei Lessig sehr breit angelegt ist und somit zwar Spielraum für die Abstraktion der Codestrukturen entlang gesellschaftlicher Prozesse bietet, haben sich mit den Software Studies (Manovich 2001; Fuller 2008; Reichert & Richterich 2015), insbesondere im Rahmen der Critical Code Studies weitere Positionen entwickelt (vgl. Mackenzie 2003; Chun 2011; Manovich 2013). Indem über Software und Code gesprochen wird, ergeben sich weitere Implikationen für eine Abgrenzung hinsichtlich der Gestaltung und Ausführung beziehungsweise

[4] Hier in Anlehnung an das OSI-Referenzmodell gedacht.

Ausführbarkeit von Code. Es erscheint zunächst naheliegend, dass Code auch Software ist, jedoch muss mit Blick auf die Repräsentation von Code dahingehend unterschieden werden, dass der Code im Sinne von Quellcode, bzw. Sourcecode gerade nicht dem kompilierten Programm oder der Software gleichzusetzen ist. Die Bedeutung von Code ist von einer paradoxalen Struktur geprägt, indem Quellcode erst dann zur eigentlichen Quelle der Anwendung wird, wenn er durch den Prozess der Kompilation aufgelöst wird. „Source code becomes a source only through its destruction, through its simultaneous nonpresence and presence“ (Chun 2011, S.25).
Aus dieser Perspektive scheint es unumgänglich, Code und Software in einer differenzierten Weise zu betrachten: schließlich entfaltet Code – als *logos* – seine Wirkmacht erst im Zusammenspiel von symbolischer und performativer Ebene, von Text und Aktion. Programmierer sind, metaphorisch gesprochen, Regisseure, die Aspekte von Welt im Medium des Codes inszenieren und als Software zur Aufführung bringen.[5]
Hieraus entwickelt sich eine doppelte Differenz des performativen Verhältnisses von Code und Subjekt, denn einerseits wird Code erst ausgeführt (exekutiert), wenn er zur Anwendung gebracht wird und somit eine andere Repräsentationsform annimmt. Andererseits wird Code innerhalb dieses Übersetzungsprozesses durch die Regeln der Maschinensprache zwar aufgelöst, jedoch ebenso auch erst verwirklicht. In diesem Zusammenhang und in der Differenz zum Codebegriff bei Lessig bemerkt Chun schließlich:

> „What is surprising ist the fact that software is code; that code is – has been made to be – executable, and this executability makes code not law, but rather every lawyer’s dream of what law should be: automatically enabling and disabling certain actions, functioning at the level of everyday practice.“ (ebd., S.27)

Indem ein Quellcode für die Ausführung auf verschiedenartiger Hardware kompiliert also in die jeweilige *Maschinensprache* übersetzt wird, lässt sich Software zwangsläufig nicht nur auf Code reduzieren, da im Rahmen der Interaktion über das Interface kontingente Ereignisse eintreten können und folglich neue Komplexitäten im Zusammenspiel von Software und Subjekt entstehen, die zugleich in kulturelle Praktiken eingebettet sind.
Code ist insofern zu verstehen als Ressource, die in der Ausführung negiert wird, aber zugleich und gerade deswegen stets appräsent ist – so wie auf Text immer wieder zurückgegriffen werden kann.

---

[5] Wir behaupten nicht, dass Programmierer immer *gute* Regisseure sind, verweisen jedoch mit dieser Metapher auf eine auch kulturelle, ästhetische und ethische Verantwortlichkeit in diesen Inszenierungsprozessen, die nicht unbedingt zum Professionsverständnis von Programmierern gehört, aber gehören sollte. Diese reflexive Bewusstsein zeigt sich beispielsweise stark in Bereichen wie etwa der Independent-Spieleprogrammierung.

Chun verwendet den Begriff „re-source“ in einer doppelten Bedeutung und fokussiert neben der ungreifbaren Struktur des Codes vor allem das komplexe Zusammenspiel von Maschine und menschlichen Kooperationsweisen (ebd. S. 25). Wenn hier nun von einer Wirkmacht die Rede ist, dann nicht willkürlich, denn Quellcode als Ressource verstanden, heißt demnach die Bereitstellung von Software und Benutzeroberflächen, vor allem heißt es aber auch, dass eine Einbettung in kulturelle Praktiken zumindest insofern mitgedacht werden sollte, als dass Code als Gegenstand helfen kann, technologische Abläufe und automatisierte Ausführungen im Zusammenhang mit menschlicher Interaktion in ihren unterschiedlichen Logiken zu verstehen (vgl. Eke/Foit/Kaerlein/ Künsemüller 2014).

Anhand der Open Source-Bewegung lässt sich eindrücklich aufzeigen, wie das komplexe Verhältnis von Code, Software und die damit verbundenen Ideologien und kulturellen Implikationen zu fassen sind. Insbesondere durch die Open Source-Bewegung wird Code – der grundsätzlich aufgrund seiner symbolischen Verfasstheit sichtbar ist, praktisch jedoch in aller Regel den Anwendern entzogen, und in der “Nutzung” von Software lediglich appräsent ist – zugänglich und damit de facto sichtbar. Dabei geht es weniger um die Fragen, was Offenheit im Kontext von Open Source tatsächlich heißt, als um die Frage, wie sich Prozesse der Wissensarbeit generell verändern (Galloway 2011, S. 383). Während Software demnach auch als Akteur ökonomischer Machtverhältnisse gesehen werden kann, wird die freie Offenlegung von Quellcode nicht nur als Gegenzug kommerzieller Interessen im prozesslogischen Sinne verstanden, vielmehr wird der Quellcode nun in eine diskursive Interaktionsstruktur eingebettet und mit Galloway (2012) gesprochen somit selbst zu einem „Interface“:

> „The open source culture of new media really means one thing today: it means open interfaces. It means the freedom to connect to technical images. Even source code is a kind of interface—an interface into a lower-level set of libraries and operation codes.“ (ebd., S. 9)

Indem Unternehmen, Dienste und Schnittstellen (Application Programming Interfaces – APIs) bereitstellen, werden Teile des Codes freigesetzt und somit einem potenziell unabgeschlossenen Nutzerkreis zur Verfügung gestellt. Eine solche Öffnung des Codes oder zumindest von Teilen des Codes – und dies ist ein bildungstheoretisch ausgesprochen relevanter Aspekt – führt zu neuen Praktiken im Umgang mit den Daten, Schnittstellen und Code, zugleich lassen sich hieraus auch neue Kommunikations- und Produktionsprozesse, die sich losgelöst von etablierten Marktlogiken begreifen lassen, wie Christoph Koenig (2011) beschreibt.

Dass Code durch kulturelle Praktiken des Austauschs und der Kollaboration sichtbar wird, stellt mithin *einen* wichtigen Aspekt im Hinblick auf Bildung im Horizont digitaler bzw. postdigitaler Kultur dar. Dies ist jedoch, wie vor dem Hintergrund unserer Ausführungen deutlich wird, keineswegs eine vollständige oder gar ausreichende Perspektive. Denn *erstens* ermöglicht die Fähigkeit, Quellcode verstehend zu lesen, Abläufe theoretisch einzuschätzen – sie ermöglicht es jedoch nicht, die komplexen Abläufe der praktischen Umsetzung hinsichtlich Planung, Ausführbarkeit und tatsächlichem Softwareprodukt in ihren kulturellen, sozialen, organisationalen und ökonomischen Aspekten zu überblicken. *Zweitens* ist mit der Einsicht in die Logiken der Code-Erstellung in keiner Weise eine Einsicht in die performativen Aspekte und subjektivierenden Effekte von Software gegeben. Es ist also etwa ohne weiteres denkbar, dass "digital gebildeten" junge Programmiererenden eine Programmiersprache beherrschen, zugleich jedoch a) weder die sozialen und ökonomischen Aspekte ihres Handelns noch b) die impliziten ethischen und machtbezogenen Effekte der selbst programmierten Software zu beurteilen wissen. Damit sind zugleich Möglichkeitsräume angesprochen, die sich im Schnittfeld von Unterwerfung und Subjektivierung verorten lassen.

Inwiefern sich der Diskurs um Free- und Open Source Software auch auf der Subjektebene auswirkt, lässt sich anhand der Beispiele beschreiben, die Gabriella Coleman (2011) in ihrer Arbeit „Code is Speech" anführt. Coleman beschreibt Prozesse, in denen sich Programmierer und Entwickler mehr und mehr in den politischen Protest einbringen, der sich gegen das dominierende Regime über geistigen Eigentum und Urheberrecht[6] richtet und Code sehr eng mit der eigenen Stimme und einer politischen Haltung verflechtet, um ihre Ideale zu artikulieren:

„Developers construct new legal meanings by challenging the idea of software as property and by crafting new free speech theories to defend the idea of software as speech" (Coleman 2011, S. 421).

Code ist hierbei nicht einfach auf die Sprache zu reduzieren, denn in Anlehnung an Foucault (1981) lassen sich Diskurse eben nicht bloß als das Sprechen über Dinge charakterisieren, sondern eher als Praktiken, die systematisch die Gegenstände hervorbringen, von denen sie sprechen (vgl. ebd. S. 74; vgl. Wulf/Göhlich/Zirfas 2001). In dieser doppelten Differenz ist auch die subjektivierende Kraft von Code eingeschrieben, denn die Haltung, die hinter den Praktiken steht und so eng mit diesen verknüpft ist, kann exemplarisch für die Manifestation von Bildungspotenzialen gewendet

---

[6] Wie schon bei Lessig spielt hier die Frage nach Copyright und dem Umgang mit Code als Ressource im produktiv-ökonomischen aber auch im ideellen/nicht-materiellen Sinne eine entscheidende Rolle. Indem sich dieses Spannungsfeld weiterzieht, liegt die Schlussfolgerung nah, dass im Code immer auch eine politische Bedeutung eingeschrieben ist. Code ist daher nicht im Einzelfall politisch relevant, sondern aufgrund seiner amorphen Struktur und den unterschiedlichen Produktionskontexten immer ein Medium politischer Ausdrucksfähigkeit.

werden. Zugleich lassen sich hieran auch neue Formen der Öffentlichkeit ablesen, die im Prozess der diskursiven Auseinandersetzung entstehen, wie man beispielsweise an Formen des Protests durch DDoS[7]-Attacken festmachen kann. Es sind eben jene (sub-)kulturellen und politischen Ausprägungen, die Verschmelzung miteinander eine Emergenz digitaler Öffentlichkeit mit sich bringen und so die Strukturen nutzen, um den Diskursraum zu prägen beziehungsweise die Rahmenbedingungen zu destabilisieren und so zur Disposition zu stellen.

> „Thus publicness is constituted not simply by speaking, writing, arguing and protesting but also through modification of the domain or platform through which these practices are enacted, making both technology and the law unstable.“ (Cox 2012, S. 93)

Indem Code und Software hier eingesetzt werden, um Ordnungssysteme zu destabilisieren, lässt sich eine Kontinuität in der Dynamik von Machtverhältnissen vorfinden, indem hier offenbar analog zu den netzwerkartigen Strukturen eine Machtverschiebung von zentralen (politischen) Institutionen hin zu dezentralen und unterschiedlich ausgeprägten Orten der “Vermachtung” stattfindet. Galloway und Thacker (2007) sprechen in diesem Zusammenhang von Vermachtungsstrukturen, die sich weg von den Institutionen, hin zu dezentralen Netzwerken bewegen[8] und vor dem Hintergrund von Software womöglich neuen Subjekt-Objekt Relationen hervorrufen.

> „Difficult, even frustrating, questions appear at this point. If no single human entity controls the network in any total way, then can we assume that a network is not controlled by humans in any total way? If humans are only a part of a network, then how can we assume that the ultimate aim of the network is a set of human - centered goals?“ (ebd., S. 154)

Im Anschluss an Foucault (1977) ist „Macht“ nie als nur einseitiges Phänomen zu verstehen, welches sich etwa lediglich auf der Seite „des Mächtigen“ verorten ließe, sondern vielmehr immer auf verschiedenen Punkten des Netzwerks verteilt zu denken ist – entsprechend ist sie, bzw. sind ihre Effekte nicht zentral kontrollierbar. Weil zudem Praktiken nicht per einmaliger Einsetzung Subjekte hervorbringen, sondern in langen Prozessen steter Wiederholung wirken, können die

---

[7] DDoS steht für Distributed Denial of Service und beschreibt die Praktik, ein System durch mehrfache und verteilte Aufrufe zum antworten zu zwingen, bis die Kapazitäten des Adressaten soweit erschöpft sind, dass das Zielsystem offline geht.

[8] Der Netzwerkbegriff bei Galloway und Thacker umfasst dabei einerseits eine technologische Protokollebene und andererseits die Dimension sozialer Beziehungsnetzwerke. Dieses Zusammenspiel bzw. dieses Wechselverhältnis von Mensch und Maschine führt zu neuen Ausprägungen von Machtformen. Während bisher stabil geglaubte Institutionen hierdurch von innen heraus im Handeln ausgenutzt („to exploit“) und somit aufgebrochen werden, verlieren sie ihre Bedeutungskraft als etablierte Zentren der Macht.

Effekte solcher auf Wiederholung angewiesener – und daher gewissen Variationen ausgesetzten – Machtpraktiken die Macht selbst unterwandern (Butler 2001). Dies geschieht beispielsweise dann, wenn Quellcode selbst die Basis für Artikulation wird, wie Coleman an den Protesten um DeCSS[9] aufzeigt. Während der ausführbare Quellcode unter dem Digital Millennium Copyright Act (DMCA) steht, fallen kommentierte Fassungen der Algorithmen nicht darunter. Hacker haben diese Grauzone ausgenutzt, um Fassungen in unterschiedlichen Programmiersprachen, teilweise in Gedichtform zu publizieren und weiter zu verbreiten[10] (vgl. Coleman 2011, S. 438ff). Indem gegen die Verbreitung des Codes auf juristischem Wege vorgegangen wurde, entwickelte sich die Problematik weg von einer Urheberrechtrechtsverletzung hin zu einer Debatte über das Grundrecht auf freie Meinungsäußerung und verdeutlichen somit, wie schwierig der Umgang mit Code trotz seiner tiefen Verwobenheit innerhalb gesellschaftlicher Strukturen noch immer ist. Zugleich wird deutlich, wie sich die Rolle der Programmierer und Hacker als Akteure innerhalb dieses umkämpften Feldes beschreiben lässt. Die hieraus entstehenden „coding publics“ (Cox 2012, S. 91f) können dabei als Gegenstand solcher emergenten und verstreuten Formen der Macht verstanden werden. Für die Medienbildung sind diese Verflechtung dahingehend von großem Interesse, als dass sich hieran (auch empirisch) aufzeigen lässt, wie eng Digitalität als eine Medialität zunehmend fundierendes, ihre Strukturen und Materialitäten hervorbringendes Phänomen (Jörissen 2014) mit unterschiedlichen Formen der Artikulation verknüpft ist und sich somit ein Analyserahmen aufspannen lässt, der diese reflexiven Potenziale erfassbar macht. – All diese Beispiele zeigen auf, in welch hohem Maße ein genuines digitales Orientierungswissen erforderlich ist und ein spezialisiertes Faktenwissen über Abläufe und Routinen gerade nicht ausreicht, um eine ganzheitliche Urteilsfähigkeit des Subjekts zu bewirken.

Fazit

Ausgehend von den unterschiedlichen Dimensionen, die sich entlang von Code und Software entfalten, erscheint es sinnvoll und notwendig zugleich, sich mit der Frage auseinanderzusetzen,

---

[9] DeCSS ist ein freie Software, mit der es möglich ist, den Inhalt einer DVD zu dekodieren, der zuvor mit dem Content Scramble System (CSS) verschlüsselt wurde.

[10] Die Formen der Verbreitung beschränken sich eben nicht nur auf digitale Räume, so gibt es beispielsweise auch T-Shirts auf denen der Algorithmus als Statement bzw. in Kommentarform gedruckt ist. Derartige subversive Praktiken, die sich im Rahmen der Medialität aufzeigen lassen, lassen sich auch in anderen Kontexten, wie dem kritischen Umgang mit der Software selbst oder der Reflexion von Inhalten aber auch Strukturen beobachten. Beispielhaft - abermals mit Bezug auf digitale Spiele - lässt sich dies entlang des Phänomens um die Let’s-Play-Videos in Verflechtung zu deren Spielkulturen zeigen (vgl. Holze & Verständig 2016).

wie Code seine Gestalt erhält, wer Entscheidungsträger im Prozess der Erstellung und Produktion sind und in welchem kulturellen, epistemischen, ästhetischen und ökonomischen Bezugshorizont – in wessen Namen, in wessen Auftrag, aufgrund welcher Problemdefinition – sie hervorgebracht wird. Wer legt das Design fest, wie gestaltet sich ein solcher Prozess? Dabei geht es sicherlich um die Implementation von Ideologemen, aber darüber hinaus wesentlich um Brüche und Differenzen zwischen Projektierungen und Modellen, ihren Implementationen den daran anschließenden Praktiken, die immer auch mit Auslegungsprozessen einhergehen und somit Möglichkeitsräume eröffnen. Mit der potenziellen Unabgeschlossenheit von Software ergibt sich auch designtheoretisch eine weitere Perspektive, denn mit dem Herstellungs- und Reaktualisierungsprozess von Code und Software gehen zugleich komplexe Beobachtungsleistungen einher, die das Verhältnis von Wissen über Körper, Subjekte, Situationen und Gebrauchsszenarien in den Blick nehmen.

Wie am Beispiel der Open Source-Bewegung verdeutlicht, wird Software dann gar zum reflexiven Gegenstand von Design selbst, wenn sie geöffnet wird und somit die Möglichkeit der Erweiterung und Rekontextualisierung über unterschiedliche Programmierschnittstellen bereitgestellt wird. Dies ist insofern von Bedeutung, als dass Design eben keine kontextfreien Dinge herstellt, sondern Szenarien entwirft, die Dinge, Kontexte und Nutzermodelle umfassen. Indem Kontexte und Nutzermodelle in den Blick genommen werden, geht es vor allem auch um die Möglichkeit Gebrauchsweisen aufzugreifen, zu transformieren, bzw. auch neu zu projektieren.
Wie ist es möglich – aus der Perspektive der Forschung, aber letztlich aus der Perspektive und Position der in und durch diese Prozesse involvierten und in ihnen hervorgebrachten Subjekte – diese zu verstehen und sich zu ihnen bildungstheoretisch-diskursiv, professionell-pädagogisch oder auch in Alltagsvollzügen zu verhalten? Deutlich wurde, dass in jedem Fall eine technologisch verengte Perspektive auf Code und Software nur unzureichende Antworten auf die hier vorgestellten und problematisierenden Fragen geben können. Vielmehr müssen, wie wir aufgezeigt haben, die sozialen, kulturellen und subjektiven Implikationen sowie die performativen Effekte von Code und Software fundiert reflektiert und beurteilt werden können.

Das hier skizzierte wechselseitige und komplexe Verhältnis von digitalen Strukturaspekten einerseits und denen aus ihnen sowie auch gegen sie hervorgehenden subjektivierenden Praktiken ist angesichts der soziotechnologischen und kulturellen Entwicklungsdynamiken ein wesentliches Feld bildungstheoretischer und empirischer Forschung. Denn die hier verhandelten Problemstellungen lassen sich sogleich auf weitere Bereiche ausweiten, die im Rahmen des

Beitrags nur peripher angerissen wurden. So bleibt zu klären, wie mit Produkten umgegangen wird, die zukünftig im Kontext des Internet der Dinge entwickelt werden und welche pädagogischen Wendungen damit einhergehen. Es ist davon auszugehen, dass die Geräte, deren Abläufe automatisiert geregelt werden in naher Zukunft mit dem Netz kommunizieren und über das Netz auch miteinander interagieren. Die Rolle der Akteure wird dabei erneut oder weitergehend zu verhandeln sein, denn es muss davon ausgegangen werden, dass vor dem Hintergrund dieser Entwicklungen auch neue bildungstheoretische Konstellationen ergeben.

Literatur

Coleman, G. (2009). Code Is Speech: Legal Tinkering, Expertise, and Protest among Free and Open Source Software Developers. *Cultural Anthropology* 24,3. 420–454.

Cox, G. (2012). Speaking Code: Coding as Aesthetic and Political Expression. Cambridge, MA: MIT Press.

Eke, N. O., Foit, L., Kaerlein, T., & Künsemöller, J. (2014) (Hrsg.). Logiken strukturbildender Prozesse: Automatismen. Schriftenreihe des Graduiertenkollegs "Automatismen. Paderborn: Fink Wilhelm.

Foucault, M. (1977). Überwachen und Strafen. Die Geburt des Gefängnisses. Frankfurt am Main: Suhrkamp.

Foucault, M. (1981). Archäologie des Wissens. Frankfurt am Main: Suhrkamp.

Fuller, M. (2008). Software Studies: A Lexicon. MIT Press.

Galloway, A. R., & Thacker, E. (2007). The Exploit: A Theory of Networks. Univ of Minnesota Press.

Galloway, A. R. (2011). What is New Media? Ten Years after The Language of New Media. *Criticism: a quarterly for literature and the arts (Wayne State Univ., Detroit, MI), 53* (3), 377.

Galloway, A. R. (2012). *The interface effect*. Cambridge: Polity.

Golumbia, D. (2009). The Cultural Logic of Computation. Harvard University Press.

Holze, J. & Verständig, D. (2016). It’s not just a game – Subversive Praktiken in digitalen Spielkulturen. In J. Ackermann (Hrsg.), Phänomen Let’s Play-Video – Entstehung, Ästhetik,

Aneignung und Faszination aufgezeichneten Computerspielhandelns (S. 225–239). Wiesbaden: Springer VS.

~~*Jörissen, B. (2014). Digitale Medialität. In: C. Wulf & J. Zirfas (Hrsg.), Handbuch Pädagogische Anthropologie (S. 503-514). Wiesbaden: VS-Verlag.*~~

Jörissen, B. (2015). Bildung der Dinge: Design und Subjektivation. In B. Jörissen & T. Meyer (Hrsg.), Subjekt Medium Bildung (S. 215-233). Wiesbaden: Springer VS.

Kitchin, R., & Dodge, M. (2011). Code/space: Software and everyday life. Software studies. Cambridge, Mass.: MIT Press.

KMK [Sekretariat der Ständigen Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland] (2016). Strategie der Kultusministerkonferenz „Bildung in der digitalen Welt“, Version 1.0 (Entwurf), Stand 27.04.2016. URL: https://www.kmk.org/fileadmin/Dateien/pdf/PresseUndAktuelles/2016/Entwurf_KMK-Strategie_Bildung_in_der_digitalen_Welt.pdf zugegriffen am 27.06.2016.

Lessig, L. (1999). *Code and other laws of cyberspace*. New York, NY: Basic Books.

Lessig, L. (2000). Code Is Law. On Liberty in Cyberspace. http://harvardmagazine.com/2000/01/code-is-law-html

Mackenzie, A. (2003) The problem of computer code: Leviathan or common power. URL http://www.lancaster.ac.uk/staff/mackenza/papers/code-leviathan.pdf

Manovich, L. (2001). The Language of New Media. London: MIT Press.

Manovich, L. (2013). Software Takes Command. Bloomsbury: Bloomsbury Academic.

Mareis, C. (2011). Design als Wissenskultur: Interferenzen zwischen Design- und Wissensdiskursen seit 1960. Bielefeld: Transcript.

Mareis, C., Held, M., & Joost, G. (Hrsg.). (2013). Wer gestaltet die Gestaltung?: Praxis, Theorie und Geschichte des partizipatorischen Designs. Bielefeld: Transcript Verlag.

Moebius, S., & Prinz, S. (Hrsg.). (2012). Das Design der Gesellschaft: zur Kultursoziologie des Designs. Bielefeld: Transcript.

Münker, S. (2009). Emergenz digitaler Öffentlichkeiten. Die sozialen Medien im Web 2.0 (Edition Unseld, Bd. 26, Orig.-Ausg., 1. Aufl). Frankfurt am Main: Suhrkamp.

Parisi, L. (2013). Contagious architecture: Computation, aesthetics, and space. Technologies of lived abstraction. Cambridge, Massachusetts l, London, England: The MIT Press.

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

Reidenberg, J. R. (1996). Governing Networks and Rule-Making in Cyberspace. Emory Law Journal 45.

Reichert, R., & Richterich, A. (Hrsg.). (2015). Digital Culture & Society: Vol. 1.2015,1. Digital material/ism. Bielefeld: transcript.

Tungstall, E. (2013). Decolonizing design innovation: design anthropology and indigenous knowledge. In W. Gunn, T. Otto, & R. C. Smith (Hrsg.), Design Anthropology: Theory and Practice (S. 232-250). A&C Black.

Wulf, C., Göhlich, M., & Zirfas, J. (Hrsg.). (2001). Grundlagen des Performativen: Eine Einführung in die Zusammenhänge von Sprache, Macht und Handeln. Weinheim: Juventa.