Die Entwicklung von Forschungssoftware als praktische Interdisziplinarität

Gerret von Nordheim / Lars Koppers / Karin Boczek / Jonas Rieger / Carsten Jentsch / Henrik Müller / Jörg Rahnenführer*

*Wir stellen in diesem Aufsatz ein Modell interdisziplinärer Zusammenarbeit zwischen Kommunikationswissenschaft und Methodenwissenschaft (hier: Statistik) vor. Dabei steht die Frage im Mittelpunkt, wie sich die Kollaboration grundverschiedener Disziplinen über einen längeren Zeitraum verstetigen lässt. Der agilen Entwicklung von Forschungssoftware, die die fachübergreifende Zusammenarbeit strukturiert, kann hierbei eine zentrale Rolle zukommen. Sie ermöglicht zwei Ebenen der Zusammenarbeit mit verschiedenen Zeithorizonten: auf der einen werden zeitlich begrenzte Projekte umgesetzt, auf der anderen die längerfristige Zusammenarbeit organisiert. Wechselnde Team-Konstellationen und Hierarchien ermöglichen kontinuierlichen Wissensaustausch, die verwendete Programmiersprache ist Grundlage für die Entstehung einer fächerübergreifenden Kommunikation und die Orientierung auf ein gemeinsames Produkt schafft eine Kultur der gemeinsamen Verantwortlichkeit. Um die theoretischen Reflexionen an der Forschungspraxis zu spiegeln, greifen die Autor*innen auf ihre Erfahrungen im Kontext des Dortmund Center für datenbasierte Medien-Analyse (kurz: DoCMA) zurück. DoCMA ist deswegen als Fallbeispiel relevant, da Struktur und Organisation hier auf die Erfordernisse der agilen Entwicklung offener Forschungssoftware ausgerichtet wurden.*

Schlüsselwörter: Interdisziplinarität, agile Software-Entwicklung, Forschungssoftware, interpositionales Wissen, intersektionales Wissen, Programmiersprache R

The Development of Research Software as Practical Interdisciplinarity

In this paper we present a model of interdisciplinary cooperation between communication science and methodological science (here: statistics). We focus on the question of how the collaboration of fundamentally different disciplines can be perpetuated over a longer period of time. The agile development of research software that structures cross-disciplinary collaboration can play a central role here. It enables two levels of collaboration with different time horizons: On one, fixed-term projects

* Dr. Gerret von Nordheim, Universität Hamburg, Institut für Journalistik und Kommunikationswissenschaft, Allende-Platz 1, 20146 Hamburg, Deutschland, gerret.vonnordheim@uni-hamburg.de;
M.Sc. Lars Koppers, Karlsruher Institut für Technologie (KIT), Department für Wissenschaftskommunikation, Englerstr. 2, 76131 Karlsruhe, Deutschland, lars.koppers@kit.edu;
Jun.-Prof. Dr. Karin Boczek, Johannes Gutenberg-Universität Mainz, Journalistisches Seminar, Alte Universitätsstraße 17, 55116 Mainz, Deutschland, karin.boczek@uni-mainz.de;
M.Sc. Jonas Rieger, Technische Universität Dortmund, Fakultät Statistik, 44221 Dortmund, Deutschland, rieger@statistik.tu-dortmund.de;
Prof. Dr. Carsten Jentsch, Technische Universität Dortmund, Fakultät Statistik, 44221 Dortmund, Deutschland, jentsch@statistik.tu-dortmund.de;
Prof. Dr. Henrik Müller, Technische Universität Dortmund, Institut für Journalistik, 44221 Dortmund, Deutschland, henrik.mueller@tu-dortmund.de;
Prof. Dr. Jörg Rahnenführer, Technische Universität Dortmund, Fakultät Statistik, 44221 Dortmund, Deutschland, rahnenfuehrer@statistik.tu-dortmund.de.

 DOI: 10.5771/1615-634X-2021-1-80

are implemented; on the other, longer-term collaboration is organized. Changing team constellations and hierarchies enable continuous knowledge exchange, the applied programming language is the foundation for the emergence of interdisciplinary communication, and the orientation towards a common product creates a culture of shared responsibility. In order to relate the theoretical reflections to research practice, the authors draw on their experiences in the context of the Dortmund Center for Data-Based Media Analysis (DoCMA). DoCMA is relevant as a case study because its structure and organization were designed to meet the requirements of agile development of open research software.

Keywords: interdisciplinarity, agile software development, research software, interpositional knowledge, intersectional knowledge, R programming language.

1. Einleitung

In einer komplexen und verschränkt globalisierten Welt bedarf es interdisziplinärer Ansätze, um gesellschaftliche Probleme zu bearbeiten. In der Wissenschaft vollzieht sich Fortschritt zunehmend „an den Grenzen beziehungsweise an den Schnittstellen zwischen den Disziplinen" (Kleiner, Vorwort zu Defila u. a., 2008, S. VII). Die Tendenz hin zur Interdisziplinarität ist nicht neu (vgl. Brozek und Keys, 1944), sie hat sich jedoch in den vergangenen Jahren verstärkt. So ist die Vergabe von Forschungsgeldern inzwischen häufig an interdisziplinäre Projektausrichtungen gekoppelt, was als Symptom und Triebfeder dieser Entwicklung gelten kann (DFG, 2020; BMBF, 2019). Für einzelne, insbesondere kleine Fächer entsteht hierdurch ein Spannungsfeld, wie Loosen u. a. unlängst in einem Konferenz-Call zum Thema interdisziplinäre Journalismusforschung konstatierten: Zwischen Selbstverständnisdebatten und disziplinärer Konturierung auf der einen und Rufen nach integrativ(er)er interdisziplinärer Forschung auf der anderen Seite (vgl. Loosen u. a., 2019). Auch Tsatsou beobachtet, dass die Forderungen nach interdisziplinärer Öffnung an die Adresse der Medien- und Kommunikationswissenschaftler*innen immer dringlicher werden (vgl. Tsatsou, 2016, S. 652). Doch auch sie sieht damit verbunden eine „fundamental tension" (ebd.): Zwar sei die Disziplin einerseits inhärent interdisziplinär mit traditionellen Querverbindungen zu Sozial-, Geistes- und Politikwissenschaften, Linguistik, Cultural Studies oder Sozialpsychologie. Andererseits beobachtet die Autorin, dass Medien- und Kommunikationswissenschaftler*innen zögerten, ihre „comfort zone" zu verlassen – sie scheiterten daran, Praktiken zu entwickeln, die eine Kollaboration mit Disziplinen ermöglichen, die eine andere Sprache sprechen und einer anderen epistemologischen Tradition angehören (vgl. ebd., S. 653).

Analog betonen Peters und Broersma (2019), dass Digitalisierung, Medienkonvergenz und die Entstehung von Web 2.0 und 3.0 die Medienlandschaft derart verändert hätten, dass Journalismusforscher*innen kaum auf sich allein gestellt schritthalten könnten. Als Antwort auf diese Herausforderungen sei in den vergangenen Jahren eine ganze Reihe „neuer und verbesserter" Forschungstechnologien vorgestellt worden (Peters und Broersma, 2019, S. 661) – diese werden zumeist unter dem Sammelbegriff „computational methods" zusammengefasst, also beispielsweise Netzwerkanalysen, automatisierte Inhaltsanalysen wie überwachte Klassifikationen oder unüberwachte Verfahren wie Topic Modeling, Word Embeddings etc. (für eine weiterführende Übersicht vgl. van Atteveldt und Peng, 2018). In einem Feld, das ohnehin zur Multidisziplinarität neige, seien solche Trends „intellectually seductive (to say nothing of overwhelming)" (ebd.) – gerade deswegen sei jedoch eine gewisse Vorsicht geboten. Um die Potenziale der neuen Methoden zu heben, müssen Medien- und Kommunikationswissenschaftler*innen Wege erkunden, die mit ihnen einhergehenden vielgestaltigen Herausforderungen zu bewäl-

tigen – etwa die Speicherung und Verarbeitung großer Datenmengen oder die Weiterentwicklung und Adaption algorithmischer Methoden.

Um einer Lösung dieser Probleme näherzukommen, möchten wir in diesem Aufsatz ein Modell interdisziplinärer Zusammenarbeit zwischen Kommunikationswissenschaft und Methodenwissenschaft (hier: Statistik) vorstellen. Insbesondere soll dabei die Frage im Mittelpunkt stehen, wie sich die Kollaboration grundverschiedener Disziplinen über einen längeren Zeitraum verstetigen lässt. Der iterativen Entwicklung von Forschungssoftware, die die fachübergreifende Zusammenarbeit strukturiert, kann hierbei eine zentrale Rolle zukommen – sie ist die Grundlage der hier beschriebenen praktischen Interdisziplinarität, ihr Bindeglied. In gemeinsamen Forschungsprojekten werden gleichzeitig Iterationen der Software-Entwicklung vorgenommen, die sich in neuen Features und Versionen manifestieren. Dabei wird deutlich, wie Software-Entwicklung die kontinuierliche interdisziplinäre Praxis stimulieren kann: Insbesondere die Prinzipien der agilen Entwicklung von Programmen strukturieren die Zusammenarbeit und fungieren als zentrales Bindeglied der interdisziplinären Forschungsarbeit.

Das folgende Kapitel stellt die Prinzipien des interdisziplinären Arbeitens und der agilen Softwareentwicklung dar. In einem zweiten Schritt wird das daraus abgeleitete Konzept der agilen Softwareentwicklung als interdisziplinäre Forschungspraxis in einer konkreten Umsetzung beschrieben: Das Fallbeispiel des Dortmund Center für datenbasierte Medien-Analyse (DoCMA) zeigt, wie sich jahrelange fächerübergreifende Zusammenarbeit entlang kontinuierlicher Softwareentwicklung gestalten lässt. DoCMA wurde 2014 unter Beteiligung von Professoren des Instituts für Journalistik sowie der Fakultäten Statistik und Informatik der TU Dortmund gegründet. Am Anfang der Kollaboration stand das gemeinsame Ziel, massenmediale Berichterstattung mittels Text-Mining-Verfahren auszuwerten. In gemeinsamen Forschungsvorhaben entstanden Software-Funktionalitäten, aus denen wiederum in der längerfristigen Kooperation das gemeinsame Projekt einer für Dritte nutzbaren Forschungssoftware hervorging (das R-Paket *tosca*). Diese Art der Zusammenarbeit wird im vorliegenden Paper systematisch als interdisziplinäre Forschungspraxis vorgestellt. Forschungsprobleme und -projekte werden skizziert, außerdem ihre interdisziplinäre Dimension und ihr jeweiliger Beitrag zum gemeinsamen interdisziplinären Software-Produkt.

2. Softwareentwicklung als interdisziplinäre Forschungspraxis

Im folgenden Kapitel sollen zunächst Begriffe umrissen werden, die jeweils bestimmte Konstellationen der fächerübergreifenden Forschung beschreiben: Parallele werden von integrativen Ansätzen unterschieden; Modelle, in denen Disziplinen gleichberechtigt kooperieren, werden von solchen abgegrenzt, in denen sich Fächer dem wissenschaftlichen Fortschritt anderer unterordnen. Neben diesen Formen fächerübergreifender Forschungspraxis und ihren Charakteristika werden bestimmte Elemente beschrieben, die für die Organisation der Zusammenarbeit von zentraler Bedeutung sind, etwa der Umgang und die Verteilung von Wissen, die gemeinsame Kommunikation etc. Diese allgemeine Annäherung an interdisziplinäre Formen setzt den Rahmen für den zweiten Teil des Kapitels, in dem die Prinzipien agiler Softwareentwicklung umrissen werden, um sie schließlich als Modus interdisziplinärer Forschungspraxis zu beschreiben.

2.1 Formen, Charakteristika und Elemente fächerübergreifender Forschungspraxis

Terminologisch ist grundsätzlich zwischen multidisziplinärer und interdisziplinärer Forschung zu unterscheiden. Während erstere vor allem komplementäre, aber separate

Beiträge der beteiligten Fächer zur Beantwortung gemeinsamer Fragestellungen umfasst, zielt letztere auf die systematische Integration von Ideen: „There is a requirement for a form of collaboration that leads to the design of new types of complex empirical approaches along with integrated analyses combining methods and concepts from participating disciplines“ (Fiore, 2008, S. 254). Ähnlich definieren Defila u. a. Interdisziplinarität als „integrationsorientiertes Zusammenwirken von Personen aus mindestens zwei Disziplinen im Hinblick auf gemeinsame Ziele und Ergebnisse, indem die disziplinären Sichtweisen zu einer Gesamtsicht zusammengeführt werden“ (Defila u. a., 2008, S. 12).

Analog zu der Unterscheidung zwischen multi- und interdisziplinärer Forschung differenzieren Barry u. a. (2008, S. 28) zwischen verschiedenen Modi der Forschungszusammenarbeit (die sie jedoch insgesamt unter den Begriff der Interdisziplinarität subsumieren): Im „subordination-service mode“ ordnet sich eine Service-Disziplin der Haupt-Disziplin unter, arbeitet ihr zu und kompensiert das Fehlen von Kompetenzen oder Perspektiven (ebd., S. 29). Dabei sind Disziplinen keineswegs auf eine Rolle abonniert; in verschiedenen Konstellationen sind unterschiedliche Funktionszuweisungen möglich. Sozialwissenschaften können sich beispielsweise als Haupt-Disziplin mit methodisch ausgerichteten Fächern zusammentun, aber auch als Service-Disziplin in Kooperation mit naturwissenschaftlichen Disziplinen zur Analyse ansonsten unberücksichtigter „sozialer Faktoren“ (ebd.) beitragen.

Im Gegensatz dazu stehe der „integrative-synthesis mode“, der beidseitige epistemische Transformation im oben beschriebenen wahrhaft interdisziplinären Sinne bedeute (ebd., S. 28). Beide Modi träten nicht exklusiv, sondern in der Praxis zumeist parallel auf. Der Katalysator beim Schritt von multi- zu interdisziplinären Teams ist laut Fiore die zunehmende Komplexität der Probleme, die die Forscher*innen gemeinsam zu adressieren versuchen (Fiore, 2008, S. 254). Barry u. a. betonen entsprechend, dass insbesondere integrativ-synthetische Forschung das Potenzial habe, Innovationen hervorzubringen – also Erneuerungen, die „protend and open up the space of future possibilities“ (Barry u. a., 2008, S. 26).

Zentral bei der interdisziplinären Arbeit ist die effektive Verteilung von Wissen, also die Unterscheidung zwischen Expertenwissen und generalisiertem Wissen (Porter u. a., 2006, S. 192), das gegenseitiges Verständnis und überhaupt erst Kommunikation ermöglicht. Die Pflege von „partially overlapping knowledge“ (Fiore, 2008, S. 256) manifestiert sich in einer *gemeinsamen Sprache*, in der sich die Interdisziplinarität als Hybrid der Technolekte, also der Fachsprachen der einzelnen Disziplinen, spiegelt. Schon 1944 beschrieben Brozek und Keys die fächerübergreifende Kollaboration dementsprechend als „social art“ (Brozek und Keys, 1944, S. 512), die von den Beteiligten Geduld erfordere. In ihrem kanonischen Science-Aufsatz zu den allgemeinen Aspekten interdisziplinärer Arbeit betonten sie, dass es für die fächerübergreifende Kollaboration nicht nur „abstract, theoretical intelligence (and, frequently, manipulative skill) but also ‘social intelligence’“ (ebd.) brauche.

In einem Forschungsüberblick stellt Fiore fest, dass erfolgreiche fächerübergreifende Kollaboration mit Konzepten aus der Teamwork-Forschung korrespondiere: Rollendifferenzierung, komplementäres Wissen und gemeinsame mentale Modelle sind entscheidende Erfolgsfaktoren. Förderlich für diese Qualitäten sei insbesondere der Aufbau von interpositionalem Wissen, das durch dynamische Funktionszirkulation innerhalb des Teams entstehe. Salas u. a. (2007, S. 474) beschreiben diese Technik als „cross-training“: Durch die Positionsrotation entwickeln die Team-Mitglieder ein Verständnis für das Wissen und die Fähigkeiten, die nötig sind, um die Aufgaben ihrer Kollegen auszu-

führen. Interpositionales Wissen fördere das gegenseitige Verständnis innerhalb des Teams und biete die Möglichkeit, cross-disziplinäre Perspektiven auf gemeinsame Forschungsprobleme zu entwickeln (Fiore, 2008, S. 268). Verwandt damit ist das Konzept der *shared leadership*, also die wechselnde Übertragung von Führungsaufgaben innerhalb des Teams „in order to take advantage of member strengths (e.g., knowledge, skills, attitudes, perspectives, contacts, and time available) as dictated by either environmental demands or the development stage of the team" (Burke u. a., 2003, S. 105).

Gleichzeitig zeichnet sich interdisziplinäre Zusammenarbeit durch die Logik der Verantwortlichkeit aus („logic of accountability", siehe Barry u. a., 2008, S. 31). Die Idee dahinter ist, dass die Zusammenarbeit mit fremden Fächern zahlreiche Anlässe der gegenseitigen Erklärung (oder auch Rechenschaft) erzeugt – diese Irritation kann zu einer Steigerung der Qualität interdisziplinärer Forschung beitragen. Interdisziplinäres Arbeiten katalysiere auf diese Weise Lernprozesse, Wissenstransfer und die Aneignung und Entwicklung von Fähigkeiten. Der Aufbau einer interdisziplinären Forschungsinfrastruktur ist in diesem Sinne sowohl ein Investment in materielle Ressourcen als auch in Humankapital (Woolley u. a., 2015, S. 568).

Die Konstellationen interdisziplinärer Zusammenarbeit sind in mehrfacher Hinsicht dynamisch: Hierarchien orientieren sich an wechselnden Fragestellungen – die Disziplin, die sich in einem Projekt unterordnet, übernimmt in anderen Phasen den Lead. Mit dieser integrativen Form der Interdisziplinarität gehen indes hohe Anforderungen einher. Unter die Oberbegriffe der Teamfähigkeit oder der sozialen Intelligenz lassen sich vor allem drei zentrale Herausforderungen fächerübergreifender Kollaboration zusammenfassen: (1) ein effizientes Gleichgewicht zwischen disziplinärem Expertenwissen und allgemeinem, intersektionalem Wissen; (2) ein hohes Maß an gegenseitigem Verständnis für die Aufgaben und Arbeitsabläufe anderer Gruppenmitglieder, also die Pflege von interpositionalem Wissen; (3) eine Kultur geteilter Verantwortlichkeit auf Basis flacher Hierarchien und konstanter Kommunikation zwischen allen Beteiligten. Eine interdisziplinäre Infrastruktur, also der organisationale Rahmen der Kollaboration, muss auf diese Faktoren ausgerichtet sein.

Das beschriebene Ideal interdisziplinärer Zusammenarbeit ist in der Praxis allerdings nicht ad hoc umzusetzen und ist regelmäßig das Resultat jahrelanger Kooperation. Interdisziplinäre Arbeit muss also auch als Entwicklung betrachtet werden, die nicht im Idealzustand beginnt und in ihren Anfängen sogar an idealisierten Ansprüchen scheitern könnte. Es ist daher zentral, auch den – in der Forschung eher wenig beachteten – *Prozess der interdisziplinären Annäherung* zu beschreiben und Anforderungen zu definieren, die für das jeweilige Entwicklungsstadium angemessen sind. Beispielsweise ist es denkbar, dass interdisziplinäre Zusammenarbeit zunächst in Projekten organisiert wird, die bewusst Service- und Haupt-Disziplin definieren und erst später eine sukzessive Integration anstreben. Gleichzeitig legitimiert sich die Kollaboration erst durch wissenschaftlichen Output – diese eher kurzfristigen Ansprüche bilden ein Spannungsfeld mit den langfristigen Perspektiven, etwa dem Aufbau gemeinsamer Ressourcen, dem Finden einer gemeinsamen Sprache etc. – die Organisation interdisziplinärer Zusammenarbeit muss beide zunächst konfligierenden Modi und die damit assoziierten Geschwindigkeiten berücksichtigen.

An die Frage danach, wie dieser Weg zunehmender Integration entlang eines gemeinsamen, projektübergreifenden Zwecks zu gestalten ist, knüpfen die folgenden Überlegungen einer Kollaboration an, in deren Fokus die agile Entwicklung gemeinsamer Forschungssoftware steht.

2.2 Agile Software-Entwicklung als Modus der Interdisziplinarität

Agilität als Philosophie der Software-Entwicklung basiert auf adaptiven Prozessen und evolutionärer Entwicklung (Beck u. a., 2001). Die agile Entwicklung von Software ermöglicht, mittels inkrementeller Arbeitskadenzen („Sprints") auf Unvorhersehbarkeiten im Produktionsprozess zu reagieren (Yadav u. a., 2015, S. 542). Dies ist auch ihr entscheidender Vorteil gegenüber Methoden, die den Anspruch verfolgen, vor Beginn der Entwicklung den gesamten Entwicklungsprozess zu determinieren (und sich dadurch wiederum durch verlässlichere Planbarkeit auszeichnen).

Die agile Planung, Implementierung und das Testen neuer Code-Teile finden in sich ständig wiederholenden Kreisläufen statt, die jeweils eine neue Iteration des schnell arbeitsfähigen Software-Produkts erzeugen („Rapid Prototyping"): „The key point is that agile approaches plan features, not tasks" (Highsmith und Cockburn, 2001, S. 121). Iterationen können in diesem Sinne auch als kleine, in sich abgeschlossene Projekte beschrieben werden (Yadav u. a. bezeichnen dieses Vorgehen als „mini-waterfalls", Yadav u. a., 2015, S. 543), die aus fünf Aktivitäten bestehen (Biju, 2008, S. 98): Am Anfang steht die Planung, am Ende die Integration – dazwischen inkrementell-iteratives Design, Coding und Testen.

Zum Zweck wiederkehrender Nachjustierung und Qualitätssicherung werden an verschiedenen Stellen des Prozesses unterschiedliche Stakeholder (Kunden, Entwickler und Programmierer, Analysten und Tester, Projekt-Manager und Team-Leader) involviert (Galkina und Yachenko, 2014, S. 37). Die Behebung von Fehlern („Bugs") findet dementsprechend fortlaufend statt und nicht nur am Ende des Prozesses.

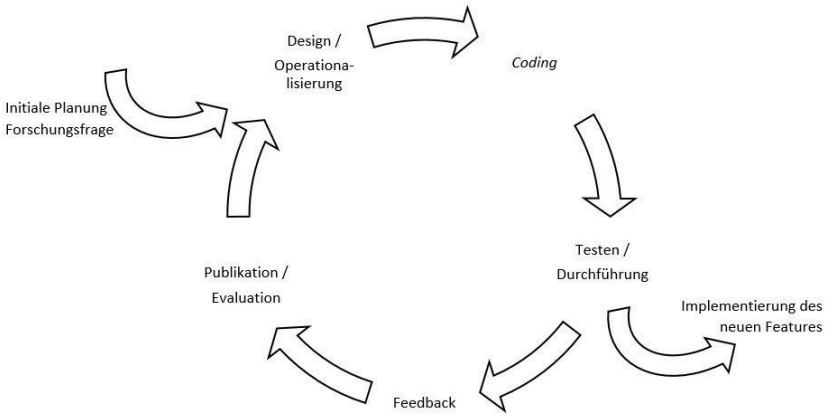
Zentrale Anforderung an die Teams, die agile Software-Entwicklung betreiben, ist Flexibilität – sie zeichnen sich zudem durch flache Hierarchien und Selbstorganisation aus (Kakar, 2017). In den verschiedenen Iterationen können und sollen die Team-Mitglieder verschiedene Funktionen übernehmen; während einer Iteration jedoch soll die Rollenzuordnung unverändert bleiben (Yadav u. a., 2015, S. 543). Diese Dynamik erfordert ständige Kommunikation (Galkina und Yachenko, 2014, S. 37), vor allem Übergaben und eine extensive Dokumentation des Codes (Yadav u. a., 2015, S. 543).

Überträgt man die Prinzipien der Agilität auf die interdisziplinäre Entwicklung von Forschungssoftware, lassen sich zwei Ebenen der Zusammenarbeit nach ihren verschiedenen Zeithorizonten unterscheiden: auf der einen werden zeitlich begrenzte Projekte umgesetzt, auf der anderen die längerfristige Zusammenarbeit organisiert:

Auf Ebene eins gibt es zunächst die konkreten Forschungsprojekte mit klar verteilten Rollen und fixen Hierarchien. In der Sprache der agilen Softwareentwicklung lassen sich diese zeitlich klar definierten Projekte als Iterationen bezeichnen (Abbildung 1 beschreibt den Zyklus einer solchen Iteration). Die einzelne Iteration ist damit ein für sich zu betrachtendes, abgeschlossenes Forschungsprojekt, dessen Output ein neues Feature, also eine neue Funktionalität der Software, darstellt. Der Ausgangspunkt jeder dieser Iterationen ist die Erkenntnis, dass die durchzuführenden Analysen später wiederholt und in ähnlicher Weise durchgeführt werden – die Automatisierung in Form von Software steigert so langfristig die Effizienz der Analysen (siehe forschungspraktische Beispiele in Folgekapiteln).

Die oben genannten Aktivitäten in jeder Iteration lassen sich als Phasen eines Forschungsprojekts beschreiben (siehe Abbildung 1), bei dem die Operationalisierung der Forschungsfrage mit der Entwicklung eines neuen Software-Features einhergeht, also mit dem Coding (Umsetzung in Programmcode), der praktischen Erprobung (Testen/Durchführung) und letztendlich der Implementierung eines neuen Features, also einer Erweiterung der Softwarefunktionalität. Der folgende Feedback- und Publikationspro-

Abbildung 1: Die Iteration der Forschungssoftware-Entwicklung als Spezialfall der agilen Softwareentwicklung



Anm.: Jeder Kreislauf stellt auch ein Forschungsprojekt dar. Die jeweils durchzuführenden Analysen werden als Programmcode umgesetzt und tragen dadurch gleichzeitig als neue Funktionalitäten – oder Features – zur Entwicklung der gemeinsamen Software bei (eigene Darstellung).

zess (bspw. im Rahmen eines Review-Verfahrens) entspricht der Überprüfung der Software durch Dritte.

Die Iteration trägt damit gleichzeitig zu einer übergeordneten Entwicklung bei, indem sie die Software auf einer zweiten Ebene als langfristiges, gemeinsames Projekt weiterentwickelt. Anknüpfend an die Iteration in Abbildung 1 ließe sich diese über das Einzelprojekt hinausweisende Entwicklung als (aufwärtsgerichtete) Spirale aus aufeinanderfolgenden Iterationen darstellen.

Insbesondere auf dieser zweiten Ebene ist interdisziplinäres Arbeiten im integrativ-synthetischen Sinn möglich. Das gemeinsame Projekt ermöglicht und katalysiert den Aufbau von intersektionalem und interpositionalem Wissen. Die Investition in gemeinsame Ressourcen und der Aufbau einer interdisziplinären *Forschungsinfrastruktur* erzeugen geteiltes Wissen. Die in verschiedenen Rollen- und Hierarchiekonstellationen durchgeführten Iterationen ermöglichen kontinuierliches Cross-Training und *shared leadership*. Die Aktivität der Softwareentwicklung führt eine *dritte Sprache* ein, die über den Code und die Dokumentation zur interdisziplinären *lingua franca* wird und so zur Integration beiträgt. Die Orientierung auf ein gemeinsames Produkt schafft eine Kultur der gemeinsamen Verantwortlichkeit gegenüber Teammitgliedern und externen Testern (bspw. Reviewer*innen) und Anwender*innen.

In den folgenden Abschnitten sollen die zwei vorgestellten Konzepte – Interdisziplinarität und Agilität – am Beispiel der Kooperation zwischen Kommunikationswissenschaft und Statistik (verallgemeinert: Anwender- und Methodendisziplin) zusammengeführt werden. Hierdurch soll deutlich werden, dass die Prinzipien der agilen Softwareentwicklung prädestiniert sind, die Arbeit fächerübergreifender Teams über lange Zeiträume zu strukturieren. Es werden hierbei verschiedene Stufen interdisziplinärer Zusammenarbeit beschrieben, zunächst sehr konkret in ihrer möglichen Operationali-

sierung, um von dort aus wieder verallgemeinerbare Rückschlüsse bezüglich der zuvor formulierten Modelle agiler Interdisziplinarität zu ziehen.

3. Software-Entwicklung als interdisziplinäre Praxis am Beispiel des Dortmund Center für datenbasierte Medien-Analyse (DoCMA)

3.1 Infrastruktur

Um die vorangegangenen theoretischen Erwägungen konkret zu exemplifizieren, greifen die Autor*innen auf ihre Erfahrungen im Kontext des Dortmund Center für datenbasierte Medien-Analyse (kurz: DoCMA) zurück. DoCMA ist deswegen als Fallbeispiel relevant, da Struktur und Organisation der interdisziplinären Zusammenarbeit hier auf die Erfordernisse der agilen Entwicklung offener Forschungssoftware ausgerichtet wurden.

Im Folgenden soll erklärt werden, was dies im Einzelnen für den Aufbau einer interdisziplinären Forschungsinfrastruktur bedeutet und wie diese (1) intersektionales und interpositionales Wissensmanagement strukturiert und (2) flache Hierarchien beziehungsweise *shared leadership* organisiert.

3.1.1 Intersektionales und interpositionales Wissensmanagement

Zentrale Grundlage für den Aufbau des gemeinsamen, generalisierten Wissens (Porter u. a., 2006) ist die Ausbildung einer gemeinsamen Sprache sowie die Etablierung gemeinsamer Kommunikationsmedien und -formen. Sie sind die Basis für ein hohes Maß an gegenseitigem Verständnis für die Aufgaben und Arbeitsabläufe anderer Gruppenmitglieder.

Tatsächlich lassen sich die Erfordernisse gemeinsamer Softwareprojekte als Ausgangspunkt der Formalisierung eigener technolektischer Codes nutzen. So werden beispielsweise an Datenformate (Input und Output der Software) sowohl technische als auch inhaltliche Anforderungen formuliert – sie sind damit ausgehandelte, interdisziplinäre Sprache, die zum gegenseitigen Verständnis und damit zur Ausprägung des intersektionalen Wissens beitragen. So müssen beispielsweise Textdaten aus verschiedenen Quellen in ein einheitliches Format überführt werden, um von der Software verarbeitet zu werden. Dieses Zielformat definiert unter anderem, welche Metadaten unter welchen Bezeichnungen in welcher Objektstruktur abgelegt werden, und ist damit eine Konvention, die auf inhaltlichen und technischen Erwägungen basiert.

In der Text-Mining-Praxis lassen sich diese Prozesse unter dem Begriff des Forschungsdatenmanagements subsumieren. Korpora können in standardisierter Form auf einer gemeinsamen zentralen Server-Infrastruktur abgelegt werden. Das ermöglicht, die ursprünglich heterogenen Korpora (zumeist verschieden strukturierte HTML- und XML-Formate) in eine einheitliche Form zu bringen, die für die Anwendung von standardisierten Funktionen zur Analyse vonnöten ist. Durch die interne Bereitstellung und Archivierung der Korpora auf einem Datenserver ist neben der Reproduzierbarkeit von Ergebnissen aus Forschungsprojekten auch eine ständige Verfügbarkeit der Daten für alle Beteiligten gesichert.

Der Ausgangspunkt der Software-Entwicklung ist die Erkenntnis, dass Analysen wiederholt und in ähnlicher Weise durchgeführt werden. Neben dem Datenformat entsteht hier als nächste Stufe gemeinsamer Sprachentwicklung die Programm-Funktion, als verallgemeinerte Version des konkreten Analyseschritts. Die Formulierung dieser Funktion muss sich wiederum den Konventionen einer Programmiersprache unterord-

nen, die so ebenfalls zum gemeinsamen Medium interdisziplinärer Arbeit wird. Forschungspraktisch bietet die Implementierung als Softwarepaket in eine bestehende Programmiersprache den Vorteil, dass sie deutlich ressourcensparender und damit leichter forschungsaktuell zu halten ist als beispielsweise eine eigenständige Software mit grafischer Oberfläche.

Die im Rahmen von DoCMA erstellte Software wird in R (R Core Team, 2020) programmiert. R ist neben Python eine der am weitesten verbreiteten Sprachen für Datenanalysen. Beide Sprachen enthalten bereits für die Kommunikationswissenschaft nützliche Programmpakete (weit verbreitet ist beispielsweise das R-Paket *quanteda*, Benoit u. a., 2018). Neben der Voraussetzung, dass für die Benutzung von R-Paketen grundlegende Kenntnisse der Sprache benötigt werden, hat die Software-Entwicklung als R-Paket Vorteile in Bezug auf Nachhaltigkeit, Validierung, Anschlussfähigkeit und Kontribution, die bei der Entwicklung von Forschungssoftware unabdingbar sind. Zu jedem R-Paket gehört beispielsweise ein Handbuch, in dem jede Funktion in einer standardisierten Form dokumentiert ist. Diese Dokumentation kann ebenso aus dem laufenden Programm heraus für einzelne Funktionen aufgerufen werden und liefert eine schnelle Hilfestellung für alle Anwender*innen. Als ausführlichere Anleitung werden R-Pakete außerdem in sogenannten Vignetten beschrieben. Da sich die Software an die interdisziplinär ausgerichtete R-Community richtet, muss auch für Handbuch und Vignette eine Sprache gefunden werden (samt veranschaulichenden Beispielen, Demo-Datensätzen etc.), die die einzelnen Fachterminologien überwindet und sowohl für Anwender aus Geistes- und Sozialwissenschaften als auch für Interessenten aus den Methodenwissenschaften angemessen ist.

3.1.2 Flache Hierarchien und gemeinsame Verantwortung

Flache Hierarchien sind die Grundlage für eine Kultur gemeinsamer Verantwortlichkeit. Entscheidend für die Entwicklung dieser Kultur sind indes als Gegenpart verschiedene interne und externe Instanzen der Rechenschaft, gegenüber denen verschiedene Beteiligte verantwortlich zeichnen. Die (agile) Softwareentwicklung geht mit verschiedenen Rechenschaftspflichten einher, die die Etablierung geteilter Verantwortung oder *shared leadership* fördern.

Im Folgenden werden die internen und externen Instanzen beschrieben, gegenüber denen sich die DoCMA-Team-Mitglieder zur Rechenschaft verpflichten. Es wird deutlich, dass diese die interdisziplinäre Zusammenarbeit durch spezifische Affordanzen strukturieren. Sie erzeugen die Notwendigkeit für Aushandlungen und Vereinbarungen, letztlich für eine Kommunikation über Kommunikation. Es entsteht auf diesem Weg eine gemeinsame Sprache, die Funktionen und Verantwortlichkeiten gegenüber verschiedenen Rechenschaftsinstanzen so definiert, dass sie intersektional verstanden und interpositional ausgefüllt werden können.

Ein R-Paket sollte beispielsweise so programmiert werden, dass Funktionalitäten automatisiert intern überprüft werden. Das heißt, dass Tests, die den vollen Funktionsumfang des Pakets abdecken, schon als wichtiger Bestandteil der iterativ-agilen Programmierung selbst vorgesehen sind (vgl. Abbildung 1). In R gibt es eine Auswahl an Paketen, die beim Erstellen solcher Tests unterstützen (Wickham, 2011; Lang, 2017). Durch die netzwerkartige Struktur der voneinander abhängigen Pakete (die meisten R-Pakete integrieren Funktionen anderer Pakete), weisen diese Tests auch auf Funktionsänderungen in Programmcodes Dritter hin – die „Depedencies“ der Software entsprechen also einem Netz gegenseitiger Verantwortlichkeiten.

R-Pakete, die im offiziellen Archiv CRAN (Comprehensive R Archive Network, https://cran.r-project.org/) gelistet werden, müssen des Weiteren grundlegende Eigenschaften erfüllen (https://cran.r-project.org/web/packages/policies.html), die von fachkundigen Mitarbeiter*innen des R-Core-Teams vor einer initialen Veröffentlichung und vor jedem folgenden Update des Pakets kontrolliert werden. Jedes Paket benötigt hauptverantwortliche Autor*innen, die für die Wartung des Pakets zuständig sind. Diese Verantwortung gegenüber der externen Instanz CRAN spiegelt sich vor allem in einer allgemeinen Erreichbarkeit, die eine zeitnahe Beantwortung von Fragen bei Problemen gewährleistet sowie eine Weiterleitung von Schwächen oder Fehlern des Pakets an die entsprechend zuständigen Programmierer*innen. Bei der Erstveröffentlichung eines Paketes muss eine Open-Source-Lizenz gewählt werden, unter der das Paket veröffentlicht wird. Um fehlerhafter Programmierung vorzubeugen, werden R-Pakete auf CRAN also nicht nur als Binärdateien, sondern auch als komplett einsehbarer Quellcode bereitgestellt. R-Pakete sind zudem in der Regel plattformunabhängig anwendbar, ein Paket soll aber mindestens unter zwei der gängigen Plattformen für R (Windows, Linux, MacOS) lauffähig sein. Diese Offenheit verschafft unabhängigen Wissenschaftler*innen die Möglichkeit zu schneller Intervention bei ungewolltem Verhalten oder bei fehlerhaften Berechnungen durch die Software.

Qualitätsmerkmale für nachhaltige Softwareentwicklung sind unter anderen die einfache Möglichkeit des Austausches über die Implementation (Community), des Meldens von Fehlern und von Rückmeldungen und Wünschen zu zukünftiger Funktionalität (Supportability) sowie die einfache Integration fremden Programmcodes in bestehende Software (Interoperability) (Jackson u. a., 2011). Diese Merkmale spiegeln sich im Design Git-basierter Plattformen (wie zum Beispiel GitHub), die als Repositorium von Beta-Versionen, Code-Archiv, als soziales Medium und Team-Plattform genutzt werden können und eine weitere Vernetzung mit Projekten und Entwickler*innen ermöglichen.

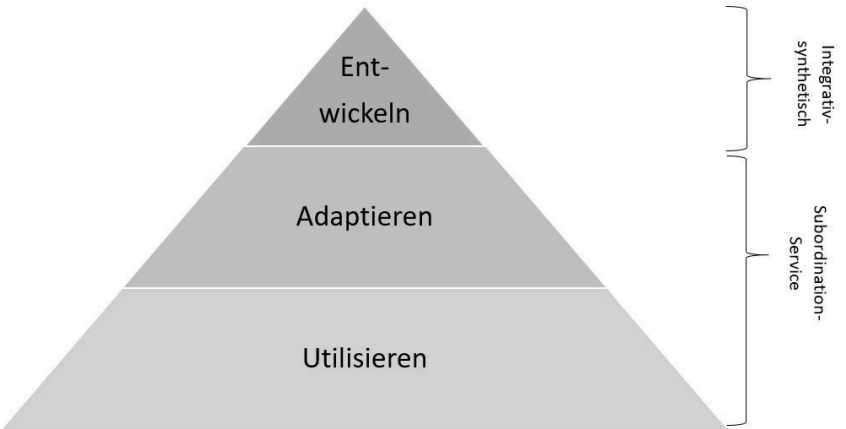
3.2 Stufen interdisziplinärer Software-Entwicklung

Im Folgenden sollen idealtypisch drei Stufen der längerfristigen Evolution (oder integrativen Annäherung) interdisziplinärer Forschungssoftware-Entwicklung beschrieben werden: die Phasen des Utilisierens, des Adaptierens sowie eine Phase der Entwicklung. Die Phasen lassen sich als Pyramide darstellen, die die zunehmenden Ansprüche an die interdisziplinäre Zusammenarbeit, aber auch ihre zunehmende Innovationsleistung visualisiert (siehe Abbildung 2): Während es auf der ersten Stufe (Utilisieren) vor allem darum geht, Funktionalitäten bestehender Software zu kombinieren und so für eigene Fragestellungen nutzbar zu machen, geht es im zweiten Schritt (Adaptieren) darum, Programmcode anzupassen und zu ergänzen. Auf der letzten Stufe (Entwickeln) schließlich werden Fragestellungen als eigenständige, neue Software-Features operationalisiert.

Entsprechend dem oben beschriebenen Bild der aufwärtsgerichteten Spirale wird die Entwicklung von Stufe eins bis drei durch Iterationen vorangetrieben, an deren Ende der Software Features hinzugefügt werden. Sie entsprechen einzelnen Forschungsprojekten, die im Folgenden exemplarisch beschrieben werden.

Zu Beginn einer interdisziplinären Zusammenarbeit zwischen Anwender- und Methodendisziplinen ist es naheliegend, zunächst das bestehende Methoden- und Softwarerepertoire mit möglichen Fragestellungen abzugleichen, also der Frage nachzugehen: Für welche Art von Untersuchungen gibt es bereits Methoden beziehungsweise Software (utilisieren) oder lassen sich Methoden oder Software anpassen (adaptieren)? Wie in Abbildung 2 dargestellt, bildet die Stufe des Utilisierens das Fundament der Anforderungspyramide. Diese Anfangsphase der interdisziplinären Zusammenarbeit

Abbildung 2: Anforderungspyramide interdisziplinärer Software-Entwicklung



Anm.: Mit den drei Stufen der Entwicklung interdisziplinärer Arbeit sind jeweils unterschiedliche Anforderungen und Innovationspotenziale assoziiert: Auf den ersten Stufen geht es um das Anwenden und Anpassen bestehender Forschungssoftware, auf der dritten Stufe um die Entwicklung neuer Features (eigene Darstellung).

zeichnet sich durch die Implementierung bekannter Methoden aus und stellt im Allgemeinen durch den hohen Aufwand und zunächst geringen Mehrwert für beide Parteien eine Herausforderung für die weitere Zusammenarbeit dar. Die frühzeitige Implementierung sich wiederholender Analyseschritte als Forschungssoftware (*rapid prototyping*) verringert jedoch den Aufwand späterer Analysen und schafft damit Kapazitäten für methodenwissenschaftliche Forschung (s. u.).

Das R-Paket *tosca* (Tools for Statistical Content Analysis, Koppers u. a., 2020) vereint die im Rahmen von DoCMA geschriebenen Funktionen. Auf der Ebene des Utilisierens sind, im Rahmen konkreter Forschungsprojekte, Features für das Einlesen von Daten, die Vorverarbeitung und das Anwenden des Topic-Modells *Latent Dirichlet Allocation* (kurz: LDA, Blei u. a., 2003) entstanden. Topic-Modeling-Verfahren wie LDA ermöglichen es, latente thematische Zusammenhänge in großen Textkorpora sichtbar zu machen. Bei der LDA werden diese Zusammenhänge als Topics modelliert, als Wahrscheinlichkeitsverteilungen über alle Wörter. Zur Interpretation der Modelle bedarf es Visualisierungen, die dem Paket bedarfsorientiert hinzugefügt wurden. Parallel wurden verschiedene Validierungsmethoden für LDA implementiert (Maier u. a., 2018; Song u. a., 2020): *tosca* ermöglicht Validierung mittels intruder words und topics (Chang u. a., 2009) sowie topic coherence (Mimno u. a., 2011).

Die Implementierung wurde dabei möglichst generisch vorgenommen, sodass die Funktionen nicht nur bei der konkreten Anwendung, sondern auch für verallgemeinerte Probleme angewandt werden können. In von Nordheim, Boczek, Koppers und Erdmann (2018) wurde bereits das Grundgerüst der Datenvorverarbeitung genutzt, so dass von Nordheim, Boczek und Koppers (2018) auf diese fundamentalen Funktionen aufbauen konnten. Insbesondere das Bereitstellen von Funktionalitäten zur Erschließung neuer Datenquellen wird zumeist durch neue Projektideen motiviert. Um die Analysen für Boczek und Koppers (2020) durchführen zu können, musste tosca etwa um eine

Funktion zum Einlesen von WhatsApp-Daten erweitert werden. Die vollständige Pipeline des Pakets konnte zusätzlich in mehreren Abschlussarbeiten und Dissertationen (Boczek, 2019; von Nordheim, 2019) im Bereich der Journalistik genutzt werden, wodurch Absolvent*innen befähigt wurden, eigene Datenprojekte durchzuführen.

Der Einsatz der Software im Bereich der Lehre ermöglicht insbesondere Studierenden der Journalistik, ohne große Programmierkenntnisse nach kurzer Einführung eigene Datenanalysen durchzuführen. Die Nutzung von *tosca* außerhalb der eigenen Forschungsgruppe wird durch die Anbindung an andere R-Pakete, wie zum Beispiel an die für *quanteda* (Benoit u. a., 2018) oder *tidyverse* (Wickham, 2019) gängigen Datenstrukturen, erleichtert.

Eine kontinuierliche und nachhaltige Kooperation wird wahrscheinlicher, wenn die beteiligten Akteure gegenseitigen Nutzen aus ihr ziehen, also möglichst gleichberechtigt zusammenarbeiten. Dies würde bedeuten, den einseitigen Subordination-Service-Modus zugunsten einer integrativ-synthetischen Zusammenarbeit zu überwinden. Dieser Modus entspricht im genannten Beispiel der Stufe der integrativen Software-Entwicklung; hierbei ist die wissenschaftliche Innovationsleistung (also der Erkenntnisfortschritt) zwischen den Disziplinen quasi gleich verteilt. Die Zusammenarbeit erzeugt auf dieser Stufe Erkenntnisse, die in beiden Disziplinen wissenschaftlich verwertbar sind.

Die Organisation dieser Form von gleichberechtigter Interdisziplinarität ist besonders komplex, da schon die Formulierung synergetisch zu bearbeitender Fragestellungen ein hohes Maß an gegenseitigem Verständnis erfordert. Wie durch die folgenden Beispiele klar wird, entsteht dieses Verständnis eben mit der Erfahrung wiederholter, ähnlich gelagerter Projekte und der damit verbundenen, iterativen Optimierung von Prozessen. So entstehen Fragestellungen auf der Ebene der Entwicklung vor allem aus der Forschungspraxis und sind damit zumeist methodischer Natur. Wiederholt auftretende Forschungsdilemmata werden zum Ausgangspunkt gemeinsamer Methodenentwicklungen und neuer Software-Features.

Praxisbeispiel 1: Effizientes Sampling

Ein Beispiel eines solchen Dilemmas aus der DoCMA-Forschungspraxis ist das zeitaufwendige Sampling von Artikeln aus einer Grundgesamtheit: Für die Beantwortung einer Forschungsfrage mit Hilfe von großen Textsammlungen müssen diese oft zu Beginn der Analyse mittels passender Suchworte auf die relevanten Texte eingeschränkt werden. Die Validierung solcher Wortfilter ist sehr zeitaufwendig, da eine ausreichend große Zufallsstichprobe von Codierer*innen betrachtet und bezüglich ihrer Relevanz beurteilt werden muss. Dieser Aufwand führt in vielen Studien dazu, dass Gütekriterien wie Precision und Recall nicht berichtet werden (Stryker u. a., 2006). Dieses forschungspraktische Dilemma erwies sich als fruchtbares interdisziplinäres Problem. Die Entwicklung eines gewichteten Schätzverfahrens für diesen Anwendungsfall ist eine typische statistische Forschungsfrage. Ein passendes Verfahren wurde entwickelt und als neues Feature in *tosca* implementiert.

Praxisbeispiel 2: Reliable LDA

Ein weiteres Beispiel ist die Frage, wie mit der Zufallskomponente von LDA-Topic-Modellen methodisch umgegangen werden kann. Bei dem probabilistischen Verfahren können sich die Ergebnisse von unabhängig auf identischen Daten durchgeführten LDA-Läufen mit gleicher Parametrisierung unterscheiden. Diese Instabilität der LDA wird in der Literatur nur selten thematisiert (Agrawal u. a., 2018). Im Sinne guter wis-

senschaftlicher Praxis ist die Eigenschaft der Instabilität ein Makel, schränkt sie doch die Reproduzierbarkeit der Ergebnisse ein. Aus dieser Beobachtung heraus hat sich das Bedürfnis nach reliableren Ergebnissen entwickelt. Zwar haben sich sowohl aus dem Bereich der Statistik und Informatik (z. B. Mantyla u. a., 2018; Su u. a., 2016; Greene u. a., 2014; Nguyen u. a., 2014; Stevens u. a., 2012; Newman u. a., 2011; Mimno u. a., 2011) als auch aus dem Bereich der Kommunikationswissenschaften (z. B. Maier u. a., 2018; Niekler, 2016; Niekler und Jähnichen, 2012) bereits vereinzelte Forschungsgruppen mit dem Thema beschäftigt – ein valides generisches Vorgehen stand aber bislang aus.

Der Lösungsansatz für das beschriebene Problem der LDA-Instabilität orientiert sich an einem klassischen Vorgehen aus der Statistik. Für eine gewählte Parametrisierung werden eine große Anzahl von LDA-Läufen (typischerweise etwa 100) durchgeführt, deren Repräsentanten es zu bestimmen gilt. Dabei wird die Ähnlichkeit von zwei Modellen aus zwei LDA-Läufen dadurch quantifiziert, wie gut sich die zugehörigen Topics paarweise aus beiden Läufen zuordnen lassen. Schlussendlich bildet das Modell den Prototyp, das die höchste mittlere paarweise Ähnlichkeit zu allen anderen LDA-Läufen besitzt. Dieses Vorgehen wird in Rieger u. a. (2020) beschrieben, ist als getestetes R-Paket auf CRAN und GitHub unter dem Namen *ldaPrototype* (Rieger, 2020) verfügbar und bereits in ersten Forschungsprojekten zum Einsatz gekommen (von Nordheim und Rieger, 2020).

Beide Beispiele verdeutlichen, dass interdisziplinäre Innovationen, d. h. Forschungsergebnisse, die in allen beteiligten Disziplinen einen Erkenntnisfortschritt darstellen, im Fall von DoCMA erst durch eine vorangegangene Phase der Annäherung möglich wurden. Die Phase des Utilisierens und Adaptierens bildete hier also ein notwendiges Fundament, das gegenseitiges Verständnis (um nicht zu sagen: gegenseitige Empathie) förderte und so die Identifikation integrativ-synthetischer Fragestellungen ermöglichte.

4. Zusammenfassung

Ausgangspunkt dieser Arbeit war die Erkenntnis, dass wissenschaftlicher Fortschritt zunehmend durch interdisziplinäre Verschränkung ermöglicht wird: Impact und Interdisziplinarität sind „twin challenges" (Tsatsou, 2016, S. 655). Insbesondere für die Medien- und Kommunikationswissenschaften ist es vor dem Hintergrund von Digitalisierung und Medienkonvergenz zwingend notwendig, sich Disziplinen zu öffnen, deren Kerngeschäft die computergestützte Analyse großer Datenmengen ist. Ein möglicher Weg, diese Annäherung zu motivieren, wurde in dieser Arbeit beschrieben – die agile Entwicklung von Forschungssoftware. Sie katalysiert interdisziplinäre Zusammenarbeit, indem sie…

– *…zwei Geschwindigkeiten ermöglicht:* Die Pyramide interdisziplinärer Anforderungen (Abbildung 2) verdeutlicht, dass die Wahrscheinlichkeit für interdisziplinären Erkenntnisfortschritt (im integrativ-synthetischen Modus) wächst, wenn hierfür zunächst Grundlagen geschaffen werden. Die Bildung dieses Fundaments lässt sich als Aufbau einer gemeinsamen Infrastruktur beschreiben. Um die interdisziplinäre Kollaboration nicht zu überfordern, sollten auf jeder Stufe realistische Erwartungen an die gemeinsame Zusammenarbeit formuliert werden – in dieser ersten Phase des Ressourcenaufbaus sollten entsprechend Projekte im Mittelpunkt stehen, die sich mit dem Utilisieren und Adaptieren bestehender Methoden (im Subordination-Service-Modus) in Form von Software befassen. Iterationen – für sich genommen sowohl Forschungsprojekte als auch Feature-Entwicklung auf basalem Niveau – führen so zu kurzfristigen Ergebnissen und zahlen gleichzeitig auf die langfristige Entwicklung der Zusammenarbeit ein. Diese Annäherung ermöglicht es später, Probleme aus

einem neuen Blickwinkel zu betrachten, zu definieren und neue Lösungsansätze im Sinne des integrativ-synthetischen Modus zu entwickeln.
Diese zwei parallelen Prozesse, die der kurzfristigen wissenschaftlichen Verwertungslogik Rechnung tragen (durch output-orientierte Iterationen) und gleichzeitig langfristige Evolution ermöglichen (durch sukzessiven Ressourcenaufbau), lassen sich als aufwärtsgerichtete Spirale beschreiben.

– *…die Entstehung einer gemeinsamen Sprache fördert:* Die kollaborative Entwicklung von Software schafft Anlässe, sprachliche Konventionen auszuhandeln. Die Programmiersprache ist gemeinsame Grundlage, aus der das Vokabular des interdisziplinären Teams entsteht; Medien der Softwareentwicklung bilden den Rahmen für diesen Prozess. *Datenformate* müssen nach inhaltlichen und technologischen Gesichtspunkten definiert werden, *Funktionen* müssen als standardisiertes Aggregat wiederkehrender Analysen formuliert und *Dokumentationen* für ein interdisziplinäres Publikum verfasst werden. All diese Aufgaben fördern die Bildung eines gemeinsamen Technolekts. Durch diese Sprachentwicklung entlang der Affordanzen der Software-Entwicklung entsteht unweigerlich gemeinsames Wissen – die elementare Grundlage interdisziplinären Arbeitens.
– *…eine Logik der Rechenschaft etabliert:* Geteilte Verantwortung und flache Hierarchien bedingen sich. Dieser Grundsatz erfolgreicher interdisziplinärer Zusammenarbeit spiegelt sich bei der Entwicklung von Open-Source-Software in verschiedenen Konstellationen der Accountability: Die Programmierer*innen sind gegenüber der Community rechenschaftspflichtig, gegenüber Distributor*innen (beispielsweise den CRAN-Betreiber*innen), anderen Entwickler*innen und Usern.

Abschließend ist es wichtig zu betonen, dass der Aufbau interdisziplinärer Strukturen über die Entwicklung gemeinsamer Forschungssoftware als Investment zu sehen ist, das sich nicht immer kurzfristig im Sinne der Verwertungs- beziehungsweise Publikationslogik des Wissenschaftsbetriebs auszahlt. Eine Disziplin wie die Medien- und Kommunikationswissenschaft könnte indes den Aufbau von Strukturen anregen, indem sie Orte institutionalisiert, die die Sichtbarkeit solcher interdisziplinärer Arbeit erhöhen. Dies geschieht sicherlich zunehmend: Publikationen[1] und Konferenzen schaffen Diskurs- und Sichtbarkeitsräume für Interdisziplinarität im Allgemeinen und Forschungssoftware im Speziellen – wir befinden uns jedoch erst am Anfang dieser Entwicklung.

Gleichzeitig müssen die Voraussetzungen für interdisziplinäre Annäherung über die gemeinsame Arbeit an Programmcode schon in der Ausbildung geschaffen werden. Aus Sicht der Medien- und Kommunikationswissenschaft kann insbesondere der Erwerb neuer Kompetenzen, etwa das Erlernen von Programmiersprachen, zur Hürde für den interdisziplinären Austausch werden. Die derzeit zu beobachtende Integration entsprechender Kompetenzen in die Curricula medien- und kommunikationswissenschaftlicher Studiengänge kann in diesem Sinne zu einer wünschenswerten Öffnung der Fächer beitragen.

Es bedarf zudem einer frühen Sensibilisierung für den Wert (und die Mühen) der interdisziplinären Entwicklung von Forschungssoftware – bestenfalls durch praktische Tandem-Projekte (zum Beispiel Hackathons), bei denen Studierende verschiedener Fächer zusammenarbeiten. Hierbei kann früh vermittelt werden, dass es bei einer interdisziplinären Entwicklung gemeinsamer Forschungssoftware nicht zuletzt um soziale Faktoren geht: die Bereitschaft, sich auf eine neue Fachkultur einzulassen, also Offenheit,

1 Neben diesem Heft selbst steht beispielsweise das 2019 erstmals erschienene Journal „Computational Communication Research“ für die zunehmende Wichtigkeit der inhärent interdisziplinären computergestützten Methoden im Fach.

Empathie und Neugierde – denn am Ende bleibt Interdisziplinarität vor allem eine „social art“ (Brozek und Keys, 1944, S. 512).

Literaturverzeichnis

Agrawal, Amritanshu, Wei Fu und Tim Menzies (2018). What is Wrong with Topic Modeling? And How to Fix It Using Search-Based Software Engineering. In: *Information and Software Technology* 98, S. 74–88. doi: 10.1016/j.infsof.2018.02.005.

Barry, Andrew, Georgina Born und Gisa Weszkalnys (2008). Logics of Interdisciplinarity. In: *Economy and Society* 37(1) (Feb.), S. 20–49. doi: 10.1080/03085140701760841.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland und Dave Thomas (2001). *Manifesto for Agile Software Development*. http://www.agilemanifesto.org/ [10.01.2021].

Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller und Akitaka Matsuo (2018). quanteda: An R Package for the Quantitative Analysis of Textual Data. In: *Journal of Open Source Software* 3(30), S. 774. doi: 10.21105/joss.00774.

Biju, Soly Mathew (2008). Agile Software Development. In: *E-Learning and Digital Media*, 5(1), S. 97–102. doi: 10.2304/elea.2008.5.1.97.

Blei, David M., Andrew Y. Ng und Michael I. Jordan (2003). Latent Dirichlet Allocation. In: *Journal of Machine Learning Research* 3, S. 993–1022. doi: 10.1162/jmlr.2003.3.4-5.993.

BMBF (2019). *Richtlinie zur Förderung von Projekten für inter- und transdisziplinär arbeitende Nachwuchsgruppen in der Sozial-ökologischen Forschung*. https://www.bmbf.de/foerderungen/bekanntmachung-2346.html [10.01.2021].

Boczek, Karin (2019). *Vielfalt als journalistischer Wert? Eine Analyse der Nutzung von Expertenquellen in der Berichterstattung mit Text-Mining und klassischer Inhaltsanalyse*. TU Dortmund: Dissertation. doi: http://dx.doi.org/10.17877/DE290R-20109.

Boczek, Karin und Lars Koppers (2020). What's New about Whatsapp for News? A Mixed Method Study on News Outlets' Strategies for Using WhatsApp“. In: *Digital Journalism* 8(1), S. 126–144. doi: 10.1080/21670811.2019.1692685.

Brozek, Josef und Ancel Keys (1944). General Aspects of Interdisciplinary Research in Experimental Human Biology. In: *Science* 100(2606) (Dez.), S. 507–512. doi: 10.1126/science.100.2606.507.

Burke, C. Shawn, Stephen M. Fiore und Eduardo Salas (2003). The Role of Shared Cognition in Enabling Shared Leadership and Team Adaptability. In: *Shared Leadership: Reframing the Hows and Whys of Leadership*. Thousand Oaks, CA: SAGE Publications, S. 103–122. doi: 10.4135/9781452229539.n5.

Chang, Jonathan, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber und David M. Blei (2009). Reading Tea Leaves: How Humans Interpret Topic Models. In: Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams und Aron Culotta (Hrsg.), *Advances in Neural Information Processing Systems 22*. NIPS. Vancouver, Canada: Curran Associates Inc., S. 288–296.

Defila, Rico, Antonietta Di Giulio und Michael Scheuermann (2008). *Management von Forschungsverbünden – Möglichkeiten der Professionalisierung und Unterstützung.* Deutsche Forschungsgemeinschaft. Standpunkte. Weinheim: Wiley.

DFG (2020). *Die Förderstrategie der DFG*. https://www.dfg.de/dfg_profil/geschichte/foerderung_gestern_und_heute/aktuelle_strategie/index.html [10.01.2021].

Fiore, Stephen M. (2008). Interdisciplinarity as Teamwork: How the Science of Teams Can Inform Team Science. In: *Small Group Research* 39(3), S. 251–277. doi: 10.1177/1046496408317797.

Galkina, Olga und Vladimir Yachenko (2014). Application of Iterative Software Development Methodologies for Reducing Service Quality Gaps. In: *Proceedings of the 2014 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference*. Feb. 2014, S. 36–37. doi: 10.1109/ElConRusNW.2014.6839195.

Greene, Derek, Derek O'Callaghan und Pádraig Cunningham (2014). How Many Topics? Stability Analysis for Topic Models. In: Toon Calders, Floriana Esposito, Eyke Hüllermeier und Rosa Meo (Hrsg.), *Machine Learning and Knowledge Discovery in Databases*. Berlin, Heidelberg: Springer, S. 498–513.

Highsmith, Jim und Alistair Cockburn (2001). Agile Software Development: The Business of Innovation. In: *Computer* 34(9), S. 120–127. doi: 10.1109/2.947100.

Jackson, Mike, Steve Crouch und Rob Baxter (2011). *Software Evaluation: Criteria-based Assessment.* https://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf [11.01.2021]

Kakar, Adarsh Kumar (2017). Assessing Self-S Development Teams. In: *Journal of Computer Information Systems* 57(3), S. 208–217. doi: 10.1080/07362994.2016.1184002.

Koppers, Lars, Jonas Rieger, Karin Boczek und Gerret von Nordheim (2020). *tosca: Tools for Statistical Content Analysis*. R package version 0.2-0. doi: 10.5281/zenodo.3591068.

Lang, Michel (2017). checkmate: Fast Argument Checks for Defensive R Programming. In: *The R Journal* 9(1), S. 437–445.

Loosen, Wiebke, Nina Springer, Petra Werner und Daniel Nölleke (2019). *Call for Papers zur Tagung „Interdisziplinäre Journalismusforschung – Journalismus interdisziplinär"*. https://leibniz-hbi.de/uploads/media/default/cms/media/ruq58fj_CfP_FGJournalistik_Journalismusforschung2020.pdf [07.01.2021].

Maier, Daniel, Annie Waldherr, Peter Miltner, Gregor Wiedemann, Andreas Niekler, Alexa Keinert, Barbara Pfetsch, Gerhard Heyer, Ueli Reber, Thomas Häussler, Hannah Schmid-Petri, und Silke Adam (2018). Applying LDA Topic Modeling in Communication Research: Toward a Valid and Reliable Methodology. In: *Communication Methods and Measures* 12(2-3), S. 93–118. doi: 10.1080/19312458.2018.1430754.

Mantyla, Mika V., Maelick Claes und Umar Farooq (2018). Measuring LDA Topic Stability from Clusters of Replicated Runs. In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM. Oulu, Finland: Association for Computing Machinery. doi: 10.1145/3239235.3267435.

Mimno, David, Hanna M. Wallach, Edmund Talley, Miriam Leenders und Andrew McCallum (2011). Optimizing Semantic Coherence in Topic Models. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP. Edinburgh, United Kingdom: Association for Computational Linguistics, S. 262–272.

Newman, David, Edwin V. Bonilla und Wray Buntine (2011). Improving Topic Coherence with Regularized Topic Models. In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. NIPS. Granada, Spain: Curran Associates Inc., S. 496–504.

Nguyen, Viet-An, Jordan Boyd-Graber und Philip Resnik (2014). Sometimes Average is Best: The Importance of Averaging for Prediction Using MCMC Inference in Topic Modeling. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP. Doha, Qatar: Association for Computational Linguistics, S. 1752–1757. doi: 10.3115/v1/D14-1182.

Niekler, Andreas (2016). *Automatisierte Verfahren für die Themenanalyse nachrichtenorientierter Textquellen*. Universität Leipzig: Dissertation.

Niekler, Andreas und Patrick Jähnichen (2012). Matching Results of Latent Dirichlet Allocation for Text. In: *Proceedings of ICCM 2012, 11th International Conference on Cognitive Modeling*. Berlin: Universitätsverlag der TU Berlin, S. 317–322.

Peters, Chris und Marcel Broersma (2019). Fusion Cuisine: A Functional Approach to Interdisciplinary Cooking in Journalism Studies. In: *Journalism* 20(5), S. 660–669. doi: 10.1177/1464884918760671.

Porter, Alan L., J. David Roessner, Alex S. Cohen und Marty Perreault (2006). Interdisciplinary Research: Meaning, Metrics and Nurture. In: *Research Evaluation* 15(3), S. 187–196. doi: 10.3152/147154406781775841.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. Wien, Österreich: R Foundation for Statistical Computing.

Rieger, Jonas (2020). ldaPrototype: A Method in R to Get a Prototype of Multiple Latent Dirichlet Allocations. In: *Journal of Open Source Software* 5(51), S. 2181. doi: 10.21105/joss.02181.

Rieger, Jonas, Lars Koppers, Carsten Jentsch und Jörg Rahnenführer (2020). Improving Reliability of Latent Dirichlet Allocation by Assessing Its Stability Using Clustering Techniques on Replicated Runs. arXiv:2003.04980 [cs, stat].

Salas, Eduardo, Diana R. Nichols und James E. Driskell (2007). Testing Three Team Training Strategies in Intact Teams: A Meta-Analysis. In: *Small Group Research* 38(4), S. 471–488. doi: 10.1177/1046496407304332.

Song, Hyunjin, Petro Tolochko, Jakob-Moritz Eberl, Olga Eisele, Esther Greussing, Tobias Heidenreich, Fabienne Lind, Sebastian Galyga und Hajo G. Boomgaarden (2020). In Validations We Trust? The Impact of Imperfect Human Annotations as a Gold Standard on the Quality of Validation of Automated Content Analysis. In: *Political Communication* 37(4), S. 550–572. doi: 10.1080/10584609.2020.1723752.

Stevens, Keith, Philip Kegelmeyer, David Andrzejewski und David Buttler (2012). Exploring Topic Coherence over Many Models and Many Topics. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP/CoNLL. Jeju Island, Korea: Association for Computational Linguistics, S. 952–961.

Stryker, Jo Ellen, Ricardo J. Wray, Robert C. Hornik und Itzik Yanovitzky (2006). Validation of Database Search Terms for Content Analysis: The Case of Cancer News Coverage. In: *Journalism & Mass Communication Quarterly 83*, S. 413–430. doi: 10.1177/107769900608300212.

Su, Jing, Derek Greene und Oisin Boydell (2016). Topic Stability over Noisy Sources. In: *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*. Osaka, Japan: The COLING 2016 Organizing Committee, S. 85–93.

Tsatsou, Panayiota (2016). Can Media and Communication Researchers Turn the Present Challenges of Research Impact and Interdisciplinarity into Future Opportunities? In: *International Communication Gazette* 78(7), S. 650–656. doi: 10.1177/1748048516655718.

van Atteveldt, Wouter und Tai-Quan Peng (2018). When Communication Meets Computation: Opportunities, Challenges, and Pitfalls in Computational Communication Science. In: *Communication Methods and Measures, 12*(2–3), S. 81–92. https://doi.org/10.1080/19312458.2018.1458084.

von Nordheim, Gerret (2019). *Journalism in the Age of Sigularization. Inter-Media Perspectives through Computational Methods*. TU Dortmund: Dissertation.

von Nordheim, Gerret, Karin Boczek und Lars Koppers (2018). Sourcing the Sources. In: *Digital Journalism* 6(7), S. 807–828. doi: 10.1080/21670811.2018.1490658.

von Nordheim, Gerret, Karin Boczek, Lars Koppers und Elena Erdmann (2018). Digital Traces in Context | Reuniting a Divided Public? Tracing the TTIP Debate on Twitter and in Traditional Media. In: *International Journal of Communication* 12(2018), S. 548–569.

von Nordheim, Gerret und Jonas Rieger (2020). Im Zerrspiegel des Populismus. In: *Publizistik, 65*(3), S. 403–424. doi: 10.1007/s11616-020-00591-7.

Wickham, Hadley (2011). testthat: Get Started with Testing. In: *The R Journal* 3, S. 5–10.

Wickham, Hadley (2019). Welcome to the Tidyverse. In: *Journal of Open Source Software* 4(43), S. 1686. doi: 10.21105/joss.01686.

Woolley, Richard, Mabel Sánchez-Barrioluengo, Tim Turpin und Jane Marceau (2015). Research Collaboration in the Social Sciences: What factors are associated with disciplinary and interdisciplinary collaboration? In: *Science and Public Policy* 42(4), S. 567–582. doi: 10.1093/scipol/scu074.

Yadav, Monika, Neha Goyal und Jyoti Yadav (2015). Agile Methodology over Iterative Approach of Software Development. A Review. In: *2015 – 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. März, S. 542–547.