VII. Methodischer Anhang

Die folgenden ausgewählten methodischen Dokumente geben einen Über- und Einblick in die Arbeitsweise der Reflexiven Grounded Theory Methode.

Memo M2 Feldzugang/Sampling Liquid Democracy e.V.

Memo M38 Kodierprozess Interview B8

Memo M40 Kode: Programmierlogik

Memo M76 Interviewprotokoll D1

Memo M82 Kode: Darstellung des vollständigen Kodebaums aus MaxQDA

Memo M47 Kode: Programmiertypen

Alle Interviewaufzeichnungen, Transkripte, Primärquellen, Memos und weitere Arbeitsdokumente liegen anonymisiert zur Nachvollziehbarkeit auf einem digitalen Datenträger vor und können für wissenschaftliche Zwecke bei der Autorin angefragt werden: anja@openstate.cc

VII.1 Interviewmemo M2 Sampling vom 27. Mai 2014

Interview Code Nr.: B1

Wie und warum kam ich auf diesen Gesprächspartner?

Ich erinnere mich daran, dass ich das erste Mal bewusst von der Idee einer Liquid Democracy (LD) auf der re:publica 2011 gehört habe. Daniel Reichert und Niklas Treutner, Mitglieder des Liquid Democracy e.V., haben dort enquetebeteiligung.de vorgestellt. Da ich mich im beruflichen Kontext meiner Stiftungsarbeit viel mit Beteiligungswerkzeugen und Social Media beschäftigt hatte, fand ich das spannend. Der Vortrag war außerdem äußerst kontrovers. Ich erinnere mich, dass viel darüber diskutiert wurde, ob der Prozess so sinnvoll ist und ein Erfolg werden könnte. Sicherlich hatte ich im Vorfeld im Kontext der Pirateninnen auch bereits etwas darüber gelesen oder gehört, aber daran erinnere ich mich nicht mehr.

Im Anschluss an die Konferenz habe ich mit mehr Aufmerksamkeit verfolgt, was in Sachen Enquete und Liquid Democracy passiert. Ich war zu dieser Zeit schon auf der Suche nach einem Doktorarbeitsthema und deshalb beson-

ders sensibel. Ende 2011 bin ich dann zurück nach Berlin gezogen. Zuvor hatte ich mit Christoph Bieber darüber gesprochen, zum Thema Liquid Democracy an der NRW School of Govenance zu promovieren. Zurück in Berlin bin ich regelmäßig zum Abend der Offenen Tür des Liquid Democracy e.V. gegangen und habe so langsam die zu dieser Zeit aktiven Mitglieder des Vereins kennengelernt.

Ich habe ein gutes Verhältnis zu einigen Vereinsmitgliedern und nehme weiterhin regelmäßig an den Veranstaltungen teil – z.B. an den FOLD-Konferenzen 2013 und 2014 sowie an den Themenabenden, die seit 2014 stattfinden. Beim ersten Themenabend habe ich sogar meinen Definitionsentwurf präsentiert und diskutiert. Ich glaube, dass ich trotz meiner Nähe dennoch eine kritische, wissenschaftliche Distanz halten kann. Beispielsweise stimme ich der LD-Definition des Vereins nicht komplett zu, auch wenn ich ihre Arbeit ansonsten für sinnvoll und notwendig halte.

Die Interviewanfragen an den Vorstand und drei der Programmiererinnen, die ganz am Anfang im Wesentlichen für die Programmierung verantwortlich waren oder aktuell vor allem an dem Code arbeiten, habe ich heute per Email mit Anschreiben versandt. Die Anfragen an die Entwicklerinnen gingen über den Vorstand in der Hoffnung, dass sie mein Forschungsvorhaben befürworten.

Aus den bisherigen Gesprächen habe ich den Eindruck, dass die Vereinsmitglieder mein Vorhaben gutheißen und sich darüber freuen, dass ich über das Thema schreibe. Ich bin nicht sicher, ob sie genau wissen, worum es in meiner Dissertation geht. Dass mich vor allem ihre Perspektive interessiert, habe ich oft vermittelt. Dabei hatte ich bisher aber vor allem mit dem Vorstand zu tun. Mit B1 habe ich zum Beispiel schon oft über Demokratietheorie diskutiert. Zu den Entwicklern habe ich bisher noch weniger Kontakt, aber zumindest B6 kennt mich von den Veranstaltungen des Vereins. Für jedes Interview werde ich nochmal einzeln einige Gedanken und Recherchedaten festhalten, sobald die Termine stehen.

VII.2 Memo M38 Kodieren des Interviews B8 vom 18. Februar 2015

In diesem Memo möchte ich versuchen, den Kodierprozess für ein Interview so vollständig wie möglich festzuhalten. Dabei werde ich sowohl mein Vorgehen als auch meine Gedanken während des Kodierprozesses schriftlich in diesem Memo festhalten. Die Schwierigkeit liegt darin, einen kreativen und subjektiven Prozess für andere und für mich selbst nachvollziehbar zu gestalten:

»Man mag sich darüber ärgern, dass die Autoren und Autorinnen der GT hier keine detaillierte Anleitung geben. Der Grund dafür liegt darin, dass die Kategorien- und Theorienbildung ein Schritt ist, der nicht von dem Subjekt – der Forscherin losgelöst werden kann. Darüber hinaus geht in diesen Schritt das individuelle Vorwissen, die Kenntnis des gesamten Textmaterials und des Umfelds seiner Entstehung, ein vielleicht noch nicht artikulierbares ›Gefühl‹ für den Gegenstand sowie jedenfalls das Wissen um das eigene Erkenntnisinteresse ein. Teilweise wird deswegen von GT nicht als Methodik, sondern als ›Kunstlehre‹ gesprochen.« (vgl. von Oertzen 2006: 146–147)

Genau diese Mischung aus Vorwissen, Kenntnis des Materials, Entstehungsgeschichte und Intuition gilt es also in diesem Memo offenzulegen.

Offenes Kodieren

Der erste Schritt des offenen Kodierens besteht darin, sich das gesamte Interview noch einmal anzuhören. Beim Zuhören lässt man das Material zunächst einmal nur auf sich wirken und erfasst es in Gänze. Im nächsten Schritt geht man den Text dann abschnittweise durch und trifft eine grobe erste Zuordnung von Kodes. Ich beginne meine Interviews häufig mit einer möglichst offen formulierten Frage, um die Gesprächspartnerinnen (IP steht für Interviewpartnerinnen) einzuladen von sich zu erzählen. Aufgefallen ist mir bei diesem Gespräch, dass der IP B8 direkt zu Beginn des Interviews klarstellt, dass er beim Verein angestellt und anders als ein Vereinsmitglied nicht »Liquid Democracy Fan« sei. Diese Klarstellung scheint ihm wichtig zu sein. Ich erkenne hier ein Motiv wieder, was auch in anderen Interviews als problematisch formuliert wurde – die Überschneidung von professioneller Arbeit und persönlichem Engagement.

Anschließend spricht der IP über seine Motivation und erzählt, wie er zu Liquid Democracy gekommen ist. Er benennt dabei sowohl das Erleben von Konsensgruppen an der Uni als auch die theoretische Auseinandersetzung im Studium als Motivation. Aus dieser Motivation heraus, so beschreibt der IP, hat er sich dann für ein Praktikum beim Verein beworben und relativ schnell – enttäuscht (?) – festgestellt, dass es sehr viel Alltag und wenig Raum für seine vielen Ideen gibt. Gleichzeitig beschreibt er diese Lernerfahrung allerdings auch als wichtig, weil es schließlich bei der Demokratie auch darum gehe, dass viele Leute, Produkte wie Adhocracy benutzen.

Ich habe nachgefragt, was der IP studiert hat: Mathematik und Philosophie – eine für mich, wie er sagt, überraschend beliebte Kombination. Ich erfahre, dass das so sei, weil beide Fächer eine ähnliche Herangehensweise an Logik teilen. Der IP erklärt mir, dass ihn abstrakte Phänomene faszinieren und dass er gehofft hatte, sich auch beruflich damit auseinandersetzen zu können. Auch hier höre ich wiederrum einerseits Enttäuschung oder Ernüchterung aber auch das Eingestehen eines bereichernden Lernens. Der Zusammenhang

zwischen den Kodes Programmierlogik, also der besonderen Kompetenz, abstrakte Konzepte bis in die Grenzfälle zu durchdenken, zum Kode Lernerfahrung zeigt sich hier deutlich. Der IP spricht vom Erlernen des »ordentlichen« Programmierens. Die Art seines abwägenden Bewertens begegnete mir schon in den anderen Gesprächen mit den Programmierern. Einerseits werden eigene Ideale oder Visionen vertreten, andererseits gestehen viele meiner IPs aber auch ein, ein Verständnis für die Praxis, z.B. den politischen Prozess, die Arbeitsrealität, oder hier den Alltag des Programmierens in Teams, entwickelt zu haben. Vielleicht kann man diesen Prozess unter dem Erwachsenwerden naiver junger Menschen ohne professionelle oder politische Erfahrung abtun. Vielleicht versteckt sich in ihrem anhaltenden Infragestellen dieser anderen Realität aber auch eine berechtigte Kritik und vermutete Alternative?

Über die Arbeit im Verein kommen wir dann darauf zu sprechen, was Liquid Democracy eigentlich für den IP bedeutet. Der IP erzählt mir von seiner Philosophie-Bachelorarbeit, die er zum Thema Liquid Democracy verfasst hat. Er hat sich darin mit Jürgen Habermas Ansatz zur deliberativen Demokratie beschäftigt und diese mit seiner Erfahrung aus dem Vereinsalltag zusammengebracht. Wir sprechen eine ganze Weile über das, was ihn eigentlich reizt (Politik und Demokratie, ein kulturwissenschaftliches Studium, Auseinandersetzung mit Politik und Medien). Als ich dann nochmal nachhake, wie er LD jetzt konkret definieren würde, antwortet er, dass das für ihn im Arbeitsalltag gar nicht so wichtig sei und es eher um eine »brauchbare« Definition gehe. In einem relativ langen Abschnitt, den ich momentan auch mit Lernerfahrung kodiert habe, spricht er über die Vision des Vereins und die Gründe, die das Umsetzen dieser Vision erschweren (finanzieller Druck, Arbeitsteilung, Kompromisse und Koordination). Die Rolle und Arbeit der professionellen Designerinnen im Verein kann der IP zwar verstehen, aber nicht so richtig wertschätzen. Diesen Absatz werde ich mir im nächsten Schritt ausführlicher anschauen.

Der IP spricht dann von sich aus an, was mich als Frage interessiert: »In wie fern sind wir jetzt eigentlich kommerziell und in wie fern sind wir jetzt eigentlich gemeinnützig? Es ist halt mein Job. Irgendwo muss das Geld herkommen. Gleichzeitig gibt es im Verein aber auch ein total sinnvolles Ziel.« Der IP gesteht ein, dass er das Thema für spannend und unterstützenswert hält. Die »Ebene«, auf der das Thema Beteiligung angegangen wird, sei für ihn allerdings die falsche. Die Selbstwirksamkeitserfahrung fehle. Als Gegenbeispiel berichtet er mir von seinen persönlichen Erfahrungen mit den Studienfahrten, die er für Erstsemesterstudierenden ausrichtet. Dort gäbe es den menschlichen Faktor. Er erzählt, wie sehr es ihn beeindruckt, wenn eine Gruppe zusammenwächst und etwas über Diskussions- und Entscheidungskultur lernt. Die Studierenden werden von einem »Haufen« zu einer »Gruppe«.

Es geht ihm hier also vor allem um die persönliche Erfahrung und menschliche Beziehungen untereinander, die dem IP offensichtlich im Projektalltag fehlen.

Ich frage den IP dann nochmal nach seiner Erfahrung im Vereinsalltag, nach der Art wie im Verein gearbeitet wird. Er unterscheidet dabei einmal die Zusammenarbeit mit anderen Programmiererinnen und die mit dem größeren Team. Erstere scheint für ihn unproblematisch zu sein, denn dafür gäbe es etablierte Werkzeuge und damit auch Prozesse (Git). Die Zusammenarbeit im größeren Team gestaltet sich scheinbar schwieriger. Der IP erklärt mir, dass Scrum, ein Verfahren, das für Software-Projekte entwickelt wurde, generell ganz gut funktioniert. Probleme ergeben sich, aus seiner Sicht, einerseits durch Fehleinschätzungen in der Planung aber auch durch die stärkere Kompetenzteilung. Wenn ich ihn richtig interpretiere, sagt er, dass dadurch viel »Expertise« verloren geht und Dinge deshalb auch länger dauern. Hier sind mir auch wieder die drei Phasen des Programmierprozesses eingefallen – vom goldenen Zeitalter der Do-ocracy, als Programmiererinnen auch noch Designerinnen waren und alles selbst entschieden haben – bis hin zur Phase des professionellen Projektmanagements, in der klare Aufgaben- und Rollenteilung jede in der Selbstwirksamkeit beschränken und mehr Abstimmung und Koordinierung notwendig machen.

Mit der Antwort auf meine Frage nach seinen Vorstellungen der idealen Arbeitsbedingungen landen wir zum Ende des Interviews beim Thema Open Source. Der IP spricht über den Wunsch, frei von Ressourcenknappheit und finanziellem Druck arbeiten zu können, und kommt so auf das Thema zu sprechen. Er wünscht sich so etwas wie ein bedingungsloses Grundeinkommen und eigenmotiviertes Arbeiten. Diesen Wunsch hätte er allerdings noch nicht ausführlich durchdacht.

Axiales Kodieren

Beim axialen Kodieren setzt man die Kodes miteinander in Beziehung, fasst sie zu Subkodes zusammen und bildet möglicherweise erste zentrale Kategorien. Auch wenn die beiden Schritte in der Praxis oft nicht auseinanderzuhalten sind und ich im Material vor und zurückgehe, versuche ich diesen zweiten analytischen Schritt hier bewusst getrennt darzustellen. Im zweiten Kodierschritt wiederholt man den gleichen Prozess wie im ersten, geht allerdings Satz für Satz für ausgewählte Textstellen vor, die besonders vielversprechend sind. Man könnte auch den gesamten Text durchgehen, aber die grundlegende Heuristik geht davon aus, dass sich das Ganze in Teilen wiederfinden lässt und andersherum. Somit versuche ich zunächst den Weg über ausgewählte Textstellen zu gehen. Dazu werde ich kurz begründen, warum ich die jeweiligen Abschnitte ausgesucht habe. Anschließend interpretiere ich diese mithilfe des »Variierens« und »Vergleichens.

Lernerfahrung

Als erstes Zitat habe ich eine Stelle aus der Mitte des Interviews ausgewählt (ab 25 Min.). Der IP spricht darin über den Gegensatz der Vision von LD und der täglichen Arbeit im Verein. Momentan habe ich den gesamten Absatz unter dem Kode Lernerfahrung zusammengefasst, werde jetzt aber Satz für Satz vorgehen und weitere Kodes vergeben.

Beim mehrfachen Hören des Absatzes ist mir aufgefallen, dass der IP mehrmals hintereinander eine gleiche Argumentationslogik zu wiederholen scheint. Diesem Eindruck gehe ich nach. Die Lernerfahrung scheint für ihn damit zu beginnen, dass er anerkennt, welchen Wert das Gelernte für ihn hat. Oftmals verbindet er diese Erkenntnis mit der Aussage: »Da habe ich viel gelernt.« Im ausgewählten Zitat äußert der IP, dass er Arbeitsroutinen mittlerweile für wichtig hält. Er hatte sich nur etwas anderes vorgestellt. Seine Vorstellung von der Arbeit steht dann zur gemachten Erfahrung, die hier mehrere Jahre umspannt, in einem Widerspruch, den der IP versucht aufzulösen. Anhand der anderen Textstellen, die ich mit dem Kode gesammelt habe, werde ich prüfen, ob dieses Muster trägt und auch für andere Interviews von Relevanz ist.

Den Widerspruch versucht der IP argumentativ durch eine Gegenüberstellung verschiedener Erfahrungswelten aufzulösen. Mir kommt es vor, als spricht er mit einem imaginären Gegenüber, wenn er sagt: »Vielleicht [liegt es] einfach daran, dass ich privat Sachen mache, die ich mit niemand diskutieren muss. Ich mache vermutlich ganz viele Sachen schlechter. Aber immerhin kann ich sie einmal machen. Ich arbeite dann sicher unsauber und vergesse ganz viel.« Hier stellt sich die Frage, ob seine Arbeit wohl allein wirklich schlechter wäre? Und ob das eben der einzige Maßstab sein sollte? Dieses Abwägen zwischen Schnelligkeit und Qualität ist, soweit ich aus der Literatur weiß, ein Dilemma von Softwareentwicklung. Durch das »vermutlich« und »vielleicht« wirkt diese Selbsteinschätzung wie eine Fremdeinschätzung, die der IP eigentlich gar nicht richtig teilt. Einen ähnlichen Eindruck hatte ich auch oft im Interview mit B5. Zudem erwähnt der IP, wieder in Anerkennung, dass er möglicherweise auch nicht mitbekommt, was die Arbeit so aufwendig macht. Diese Anerkennung ist allerdings unter Vorbehalt zu betrachten, da er von »Verwaltungszeug« und »Designzeug« spricht, was einerseits, neutral betrachtet, für seine Distanz zu den richtigen Begriffen andererseits aber auch als abwertend gelesen werden kann.

Im nächsten Abschnitt wiederholt sich das Argumentationsmuster erneut. Der IP erkennt zunächst an, dass der Verein langfristig eine Beteiligungsstrategie verfolgt, auch wenn die aktuelle Erfahrung manchmal zu widersprechen scheint. In seiner Vorstellung würde es helfen, auch kleinere Projekte zu machen und mehr Dinge auszuprobieren. Die Erfahrung lehrt ihn allerdings, warum das nicht gemacht wird. Der Verein steht unter finanziellem Druck. Um

politisch wirksam zu sein, müssen Anträge geschrieben werden. Außerdem arbeite man mit nicht agilen Partnern. Deshalb wünscht sich der IP, dass auch Privatpersonen an der Idee weiterarbeiten, die nicht diesen institutionellen Verpflichtungen unterliegen und somit »coole Projekte machen« und »neue Ideen« entwickeln. Den Verein selbst sieht er genau in dieser »seltsamen Position« des Widerspruchs. Einerseits sei der Verein Teil der »Zivilgesellschaft«, andererseits sei die Organisation aber auch »nicht so richtig bottom-up«.

Zusammenarbeit

Die zweite von mir ausgewählte längere Textstelle (ab 36 Min. ca. 10 Minuten) behandelt das Thema der Zusammenarbeit im Verein, einen Kode, den ich neu vergeben habe. Aus meiner Kenntnis der Literatur zum Thema Programmieren ist die Zusammenarbeit von Entwicklerinnen ein wesentliches Dilemma, mit dem sich Praktikerinnen und Theoretikerinnen schon seit einigen Jahren auseinandersetzen. Die Open-Source-Software-Bewegung ist, soweit ich das verstehe, daraus erwachsen. Ich vermute, dass das Thema auch zentral für meine Arbeit sein könnte. In dem ausgewählten Interviewabschnitt erklärt mir der IP, wie die Arbeit im Verein aktuell organisiert ist und was daran problematisch ist.

Interessant ist zunächst, dass der IP zwischen der Zusammenarbeit zwischen den Programmiererinnen, die gut zu funktionieren scheint und durch Git organisiert ist, und der Zusammenarbeit mit dem restlichen Team unterscheidet. Auffällig an dieser Unterscheidung ist, dass der IP hier das Werkzeug der Zusammenarbeit mit dem Prozess gleichsetzt. Git ist für ihn Werkzeug und Norm, welches die Zusammenarbeit regelt. Ich würde gern noch mehr über dieses Werkzeug erfahren. Welche Regeln gibt es auf Git? Wie sind diese in Software übersetzt? Mir fällt dabei auch wieder ein, dass der portugiesische Programmierer, mit dem ich letzte Woche geskypt habe, mich auf Git und einen TED Talk von Clay Shirky verwiesen hat. Clay Shirky argumentiert, soweit ich weiß, dass Git als Prinzip auch auf andere Gesellschaftsbereiche ausgeweitet werden könne. Da ich gerade Evgeny Morozovs Buch gelesen habe, bin ich sensibilisiert für den möglichen Technikoptimismus, der sich in einer Aussage wie: »Git nimmt einem viele Sorgen« versteckt.

In den folgenden Sätzen empfiehlt mir der IP Git auszuprobieren. Ich bitte ihn, mir die Software zu zeigen und zu erklären. Wir kommen dann allerdings auf den übergeordneten Prozess der Zusammenarbeit aller Teammitglieder zu sprechen, der nach dem Scrum-Verfahren organisiert ist. Bevor er allerdings auf das Verfahren eingeht, problematisiert er es als »Hype«. Er erzählt mir von einer Konferenz, auf der jemand ironisch angemerkt hätte Scrum sei »alternativlos«. Er kritisiert weiterhin, dass auch das Verfahren, das der Verein anwendet, nicht »die reine Lehre« sei. Wahrscheinlich weil ich gerade so intensiv auf den Kode der Lernerfahrung geachtet habe, sehe ich ihn hier wieder. Der IP

erklärt mir, dass das, was im Verein gemacht wird nicht viel mit Scrum zu tun habe, aber erkennt auch an, dass es nicht schlecht sei. Er formuliert dann seine Vorstellung von Scrum, einem agilen Prozess, der ständig an die Anforderungen angepasst wird. Abschließend konfligiert er diese Vorstellung wieder mit seiner Berufserfahrung, in der es auch andere Gründe für einen angepassten Prozess gibt als in seiner Vorstellung (neue Teammitglieder, Gewohnheiten). Ob es so etwas wie die »reine Scrum-Lehre« gibt, bleibt offen.

Ich frage mich, ob der Scrum-Prozess nicht genau auf diese menschlichen Aspekte des Programmierprozesses eingeht? Soweit ich mich erinnere, ist das so. Sind Scrum und Git also Versuche, den Programmierprozess so effizient wie möglich zu gestalten bzw. zu perfektionieren, während die unordentliche Realität diesen Anspruch kontinuierlich in Frage stellt und problematisiert? Oder versuchen beide genau diese Problematik zu lösen? Jetzt, da ich mir den Scrum-Prozess nochmal genauer anschaue, frage ich mich zusätzlich noch, ob sich der Programmierprozess, die Zuständigkeiten und Berechtigungen unterscheiden, je nachdem, ob es darum geht Adhocracy als Software als solche weiterzuentwickeln, oder Anpassungen für einzelne Projekte vorzunehmen? Generell weiß ich noch wenig darüber, wie die Weiterentwicklung des Quellcodes und kundenorientierte Projekte zusammenhängen.

Den Textabschnitt (ab 42 Min.) zum Kode Arbeitsteilung werde ich mir abschließend genauer ansehen. Ich frage den IP hier nach der Möglichkeit, ein Veto gegen bestimmte User Stories einzulegen, weil sie vielleicht den eigenen politischen Idealen nicht entsprechen oder technisch nicht sinnvoll sind. Der IP antwortet zunächst damit, dass in seiner Wahrnehmung die Arbeits- und Kompetenzteilung deutlich zunehmen. Er erklärt, dass er für sich selbst entschieden hat, nur noch die Aufgaben zu erledigen, die mit seiner Rolle verbunden sind. Er erinnert, dass er am Anfang viel mehr gearbeitet hat und mittlerweile nur noch das macht, was er muss. Für mich ist die Ernüchterung im folgenden Satz relativ bezeichnend für das Dilemma zwischen seinem Engagement im Verein und seinem Verständnis als bezahlten Job: »Ich gebe vielleicht Feedback, aber ich engagiere mich da nicht mehr.« Ich frage mich, ob das gewollt ist? Falls nicht, gibt es nicht auch Möglichkeiten, berufliche Arbeit so zu organisieren, dass sie die Mitarbeiterinnen motiviert sich zu engagieren? Mit dieser Herausforderung beschäftige ich mich in meiner beruflichen Realität sehr viel. Wäre es politisch nicht wünschenswert, ja geradezu notwendig solche Beteiligungsstrukturen im Verein zu erschaffen? Oder wird das schon versucht und scheitert nur in diesem Fall? Der IP fragt sich beispielsweise auch, ob es nicht vielleicht sinnvoll wäre, wenn die Programmiererinnen bei den Kundengesprächen dabei wären.

Ich glaube, der neue Beteiligungsprozess im Liquid-Verein sieht vor, dass jeweils eine Programmiererin und eine Designerin an den wöchentlichen Vorstandssitzungen beteiligt sind. In diesen Treffen werden die Projekte auf ihre

Machbarkeit hin diskutiert. Der IP merkt schließlich an, dass er theoretisch jederzeit die Möglichkeit hätte, User Stories inhaltlich und technisch zu hinterfragen. Mit seiner Erläuterung deutet er an, dass das gar nicht die relevante Frage ist. Die Arbeitsbelastung, die Größe des Teams und die Anzahl der Projekte ließen es möglicherweise gar nicht zu. Auch wenn die Arbeit im Vereinsalltag sicherlich weit davon weg ist, kommt mir hier dennoch der Vergleich des IPs zu einem Rädchen im Getriebe. So zumindest seine Selbstdarstellung. Die strategischen Entscheidungen werden ohne ihn getroffen und er kann nur ausführen. Wenn er die Strategie in Frage stellen würde, würde er wahrscheinlich das Getriebe anhalten und Probleme machen.

Dazu passt auch, dass der IP dann im nächsten Abschnitt auf die Zeit vor der Professionalisierung zu sprechen kommt. Die Designerinnen versinnbildlichen für ihn den Anbruch einer neuen Phase. In der vergangenen Zeit der Do-ocracy gab es die professionelle Rollen- und Aufgabenteilung noch nicht in dem Maße. Damals waren die Programmiererinnen auch Designerinnen. Sie konnten den Prozess noch stärker mitgestalten. Jetzt scheint es so, als wären sie am Ende der Prozesskette und würden »nur noch« dafür sorgen, dass die Software funktioniert.

VII.3 Memo M40 Programmierlogik vom 26. Februar 2015

Gibt es so etwas, wie eine eigene Logik des Programmierens? Gibt es also ein Aussagensystem aus Wahrheiten und Gesetzmäßigkeiten, die nur auf die Arbeit mit Computern zutreffen? Im Interview B6 habe ich diesen Kode das erste Mal vergeben. »Ein Bug muss einfach gefixt werden«, war die darin proklamierte Wahrheit. Im Interview B8 hat sich die Annahme weiter verfestigt. Der IP erklärte mir, dass viele Mathematikstudentinnen und Programmiererinnen Philosophie als Nebenfach belegen, weil beide Fachrichtungen sich eben mit Logik und abstrakten Beweisen beschäftigen. Er verwies mich unter anderem auch auf die Russelsche Antinomie. Bücher wie »The Psychology of the Programmer« (Weinberg 1996) oder »Dreaming in Code« (Rosenberg 2008) lassen mich außerdem vermuten, dass die Interaktion zwischen Mensch und Maschine eine besondere Beziehung darstellt bzw. als solche konstruiert und charakterisiert wird. So gibt es vielleicht eine Kultur des Programmierens, »[...] a shared set of beliefs and activities which shape their day-to-day activities« (Weinberg 1996: 39).

In diesem Memo geht es mir nicht so sehr, um das soziale Miteinander und die dafür relevanten Kodes, sondern um die Art zu denken. Logik also eher als »shared set of beliefs« und vielleicht sogar als »shared set of truths«? Die Annahme, dass Wahrheiten durch die Arbeit mit Computern entstehen

und das Denken der Programmierer beeinflussen (vgl. Rosenberg 2008), gilt es dabei im Hinterkopf zu behalten. Nun könnte man natürlich sofort fragen, warum es nicht auch andersherum sein kann, dass Maschinen immer mehr so funktionieren wie Menschen beziehungsweise beide in komplexer Interaktion miteinander stehen? Diese Interpretation entspricht meinem Standpunkt eigentlich viel eher. Ich will mir das Textmaterial dennoch anschauen und frage mich: Wenn es also so etwas, wie eine eigene Logik gibt, die das Denken und Handeln von Programmiererinnen anleitet, was macht diese Logik aus? Wie beschreiben meine IPs diese Logik und was weiß ich darüber aus der Sekundärliteratur?

1. Es gibt Wahrheit: Es gibt richtig und falsch und scheinbar objektive Kriterien für die Beurteilung von Software. Welche sind das? Und wer legt sie fest? In diesem Kontext ist der Subkode Fehler vielleicht besonders spannend, in der Software-Welt heißen sie Bugs. In einem der Interviews sagt ein Programmierer zu mir: »Ein Bug muss einfach gefixt werden.« Das scheint eine unumgängliche Wahrheit zu sein. Aber warum? Was passiert, wenn der Fehler nicht behoben wird? Stürzt die Software ab? Und was überhaupt macht einen Fehler zum Fehler? Sind sie eindeutig zu erkennen?

2. Die Randfälle müssen bedacht werden: Die Programmierlogik scheint unterschiedliche Arten von Anwendungsfällen zu kennen, die unterschiedlich wichtig für das Funktionieren einer Software sind. Dabei gibt es mindestens die beiden Subkodes Standardanwendungen und Extremanwendungen. Die Randerscheinungen sind von besonderem Interesse, weil es hier zu Problemen kommen kann. Welche Probleme können das sein? Warum ist das so wichtig, diese Extreme zu bedenken? Ich erinnere hier den Hinweis aus der Literatur, dass Programmiererinnen durch dieses logische Durchdenken aller Randfälle möglicherweise den Blick für die Mitte verlieren.

3. Software muss funktionieren: Funktionalität scheint eine zentrale Rolle in der Logik des Programmierens zu spielen. Sie wird in dem Zitat von B8 als Kriterium zur Beurteilung herangezogen (»es funktioniert«, »es funktioniert gut«). Aus der Literatur weiß ich, dass Funktionalität ein Kernelement im Qualitätsdreieck ist, neben Zeit und Kosten (Rosenberg 2008). Aber auch hier frage ich mich, was bedeutet es, wenn etwas »gut funktioniert«? Woher kommen die Kriterien dafür? Für Gerald Weinberg (1996) ist das wichtigste Kriterium, dass Software korrekt funktioniert. Korrekt bedeutet für ihn, das korrekte Ergebnis für alle möglichen Eingaben. Wenn man »correct« aus dem Englischen übersetzt, findet man fehlerfrei, genau, richtig, vorschriftsgemäß. es scheint aber eher darum zu gehen, dass Software so funktioniert, wie es von den Macherinnen intendiert ist:

»In a way, the most important reason for studying the process by which programs are written by people is not to make the programs more efficient, more compact, cheaper,

or more easily understood. Instead, the most important gain is the prospect of getting from our programs what we really want – rather than just whatever we can manage to produce in our fumbling, bumbling way.« (Weinberg 1996: 12)

Aber wie weiß man, dass sie das tut? Eventuell ist es einfach zu lösen, wenn nur eine Person Software für den eigenen Nutzen schreibt. Aber wie es, wenn zwei an der Software arbeiten? Ist dann immer klar, was zu wollen ist? Ich vermute, dass genau aus dieser Schwierigkeit der Glaube an Expertise entstanden ist. Auf der Suche nach der einen Wahrheit muss es das eine Genie geben, das sie kennt. Werden die Wahrheiten nicht doch auch schon zwischen Programmiererinnen ausgehandelt? Ich vermute, dass die Open-Source-Software-Bewegung genau dazu Ansätze entwickelt hat. Vielleicht geht es eben nicht nur um das formale Offenlegen des Codes und der Lizenzen, sondern auch um ein neues soziales Miteinander, eine Koordinations- und Kooperationskultur?

4. Die formale Struktur muss eingehalten werden: Ein wichtiges Element der Programmierlogik ist die formale Struktur. In dem Interview mit B8 geht es zwar um Mathematik, aber der IP zieht die Parallele zum Programmieren. Er trifft eine Unterscheidung zwischen der Konzeption (»abstrakte Konzepte« entwickeln) und der Umsetzung (»programmierlastig«). Er reflektiert, dass er im Studium diese notwendige »Formalität« auf der abstrakten Ebene erlernt hat, »vollständige Beweise« zu erbringen, »sehr exakt zu arbeiten« und »auch halbwegs übersichtlich aufzuschreiben«. Beim Programmieren als Praxis hat er allerdings ebenfalls »sehr viel gelernt« und programmiere »tausendmal ordentlicher als noch vor einem Jahr«.

Die Formalität findet ihrem Ausdruck am besten in den Programmiersprachen, die versuchen auf mehreren Ebenen die »Rigidität der Maschinen« (Weinberg 1996: 211) in menschliche Sprache zu übersetzen. Dabei lässt sich die Ambiguität menschlicher Sprache nie komplett auflösen: »To make progress in programming languages we must first give up the holy grail of trying to program in a 'real' language, for programming languages can never be the same as human speech« (ebd.: 214).. Sind Programmiersprachen also der Versuch sich der Maschinenlogik anzupassen. Wie denken die Liquid-Programmiererinnen darüber?

VII.4 Interviewprotokollbogen D1 vom 09. Oktober 2014

Dauer: 1 h

Ort/Räumlichkeiten: beim IP zuhause

Kontaktweg für Rückfragen: Email vorhanden

Alter: siehe Memo

Ausbildung: Informatikstudium

Beruf/Berufsstatus: selbstständiger Software-Entwickler für Lern-Software

Wie waren die Umstände der Terminverabredung, des Gesprächsortes, des Raum- und Zeitarrangements?

Wir haben uns in der Privatwohnung des IP getroffen. Zu Beginn war die Stimmung freundlich und höflich, er hat mir Tee und Hausschuhe angeboten. Wir haben mit dem Siezen angefangen, sind aber im Laufe des Gesprächs beide in das Du gefallen. Ich habe ihn nach dem Interview gefragt, ob es okay ist, wenn ich »Du« sage. Das war es.

Welche Schwierigkeiten, Vorbehalte, Misstrauensprobleme, Missverständnisse sind aufgetaucht?

Ich habe am Anfang relativ lange über mein Promotionsprojekt gesprochen und die unterschiedlichen Datenschutzoptionen erklärt. Der IP hat sich dann ziemlich schnell für die Freigabe des Transkripts entschieden und auch Interesse an der weiteren Entwicklung meiner Arbeit angemeldet.

Dieses Interview ist das erste, das ich ohne Gesprächskarten geführt habe. Ich habe dennoch den ersten Eindruck, dass wir viele mir wichtige Themenkomplexe besprochen haben. Beim Transkribieren werde ich nochmal bewusst auf meinen Redeanteil achten und reflektieren, ob ich Gedankenpausen zu vorschnell abgebrochen habe.

Nachtrag vom 17.10.2014 nach dem Transkribieren: Aufgefallen ist mir, dass ich diesmal nach längeren Redeblöcken oft vom Thema weg zu einem neuen gesteuert habe. Hier vermittle ich eventuell zu stark eine Gesprächsagenda, die ich eigentlich nicht habe oder haben will. An einigen wenigen Stellen habe ich ihn auch unterbrochen oder seine Einschätzung kommentiert. Ich werde versuchen, beim nächsten Gespräch noch zurückhaltender in meinen Bewertungen zu sein und noch mehr offene Fragen stellen. Insgesamt bin ich wieder mit dem Gefühl gegangen, dass meine Auseinandersetzung mit den LD-Entwicklerinnen richtig ist und es sich hier tatsächlich um sehr kluge, engagierte und spannende Menschen handelt.

Check

Einverständniserklärung unterschrieben: Ja

Infoblatt abgebeben mit Kontaktinfo: Ja

VII.5 Memo M47 Programmiertypen vom 29. Oktober 2015

Gegenüberstellung ausgewählter, an der Entstehung von Liquid Democracy in allen drei Phasen beteiligten, Programmiererinnen anhand verschiedener Kodes und deren Eigenschaften zur Ermittlung von *Programmiertypen*.

	A1v	*B5*	*B6*	*B8*	*B9*	*D1*	*D2*
Studium	Informatik	Informatik	Medienwissenschaft	*Mathematik Philosophie*	*Informatik*	*Informatik*	*Mathematik*
Programmieren: selbsterlernter Coder oder studierter Informatiker	Informatiker	Coder	Coder	*Coder*	*Informatiker*	*Coder*	*Informatiker*
politisches Engagement	Netzsperren	F/OSS als Antrieb	F/OSS als Antrieb (Oekonux/ CCC) Debattier-clubs	*Hochschulpolitik Fachschaft*	*F/OSS als Antrieb (Oekonux)*	*Hochschulpolitik, Fachschaft Diplomarbeit zu Filesharing, Vater ist Lokalpolitiker*	*Netzsperren*
LD-Software: Meinungsäußerung (MÄ) oder Arbeit? (A)	*MÄ (A)*	*MÄ (A)*	*MÄ (A)*	*A*	*A*	*MÄ (A)*	*MÄ*

VII.6 Kodebaum aus MaxQDA vom 05. Oktober 2016

Kodesystem	kodierte Stellen	1149
	Programmierlogik	9
	Formalität	5
	Funktionalität	2
	Wahrheit	4
	Bug	2
	Russelsche Antinomie	2
	Anwendungsfälle	4
	Randfälle	5
	Lernerfahrung (Feedbackloops)	5
	der menschliche Faktor	1
	Lernprozess	21
	Vorstellung	14
	Erfahrung	13
	ungelöster Widerspruch	10
	Lösungsstrategien	0
	Zynismus	0
	Abstumpfen	1
	Rechtfertigung	8
	Rationalisierung	4
	Anerkennung des Widerspruchs (Dialektik)	10
	gelöster Widerspruch	2
	anerkanntes Scheitern	1
	Kompromiss	1
	Bestätigung	1
	Lernbereiche	0

	technisches Wissen	2
	politisches Lernen	2
	akademisches Lernen	1
	institutionelles Lernen	4
	Do-ocracy	14
	Pragmatismus	1
	Liquid Democracy	0
	Wurzeln	57
	Vorbilder	0
	Linux	3
	Loomio	4
	P2P-Netzwerke	1
	Hacker	2
	Island	2
	Wikipedia	6
	Mailinglisten	3
	Stack Overflow, Reddit, Slashdot	2
	DemoEx	1
	Netzwerk	2
	CCC	1
	Volksgesetzgebungsverfahren in Lettland	1
	Definition	18
	Vision (Utopie)	8
	technische Vision	5
	plattformunabhängig	2
	regelmäßige Updates und Iterationen	2
	nützlich und perfomant sein	4
	digitale Beteiligungs-Infrastruktur	10

	in Echtzeit	1
	embeddbar	2
	Beteiligung mit Geo-Bezug	1
	gesellschaftspolitische Vision	1
	interaktive Demokratie	1
	liquid-demokratische Partei	3
	Beteiligungsdemokratie	1
	bessere Repräsentation der eigenen Interessen	2
	Berücksichtigung der Stimme unabhängig von Beteiligungszeit	1
	Beteiligungsschwelle senken	5
	praktisches Entscheidungstool für Parteien	9
	Gesellschaftskonzept	2
	mögliche Staatsform	1
	offensive Marktstrategie	3
	Customizing	1
	software as a service	2
	mehr demokratische Lebenszeit	4
	Direkter Parlamentarismus	2
	Demokratisierung von Expertise	1
	mehr Laien-Expertise einzelner Bürger einbeziehen	5
	Zentralisierung von Macht aufbrechen	1
	Legitimation von Experten	4
	thematisch-funktionale Differenzierung abbilden	1
	demokratischem Ideal näher kommen	4
	politisches Experiment	6
	Antwort auf ein neues Bedürfnis (Bedarf)	7

	Politik attraktiver gestalten	2
	Verfahren	4
	Eigenschaften	0
	Klonresistenz	2
	Offenheit	0
	Transparenz	2
	Moderationsfreiheit	7
	Flexibilität	5
	reader to leader	2
	Diskurs	11
	Bewertung/Sortierung	1
	strukturiertes quantifiziertes Feedback	8
	kollaborative Textarbeit	5
	Entscheidungsmechanismus	26
	Wahlverfahren	0
	Schulze-Methode	2
	Delegation/delegated voting	28
	global	1
	transitiv	4
	nach Themen(bereichern) oder Politikfeldern	2
	Voraussetzungen	5
	Ressourcenkopplung	9
	neue politische Strukturen und Entscheidungsabläufe	2
	Sachorientierung	1
	Software	5
	Nachvollziehbarkeit	6
	Identität	2

	Wille zum aktiven Mitgestalten	1
	Anwendungsfälle	10
	BEO	4
	Bundesverband Piratenpartei	8
	meinBerlin.de	0
	Beteiligungsverfahren Tempelhofer Feld	0
	Julia Reda	1
	LQPP LV Bayern	1
	Online-Antrag Die Linke	2
	Online-Antrag SPD	2
	Enquete-Kommission für Internet und digitale Gesellschaft	3
	SMV Mecklenburg-Vorpommern	1
	SMV Sachsen	1
	LQPP LV Berlin	11
	SMV Berlin	5
	Organisationsform	3
	Open Source-Lizenz	3
	Agentur	1
	neue Strukturen	6
	Genossenschaft	6
	Piraten-Squad	8
	Verein	16
	Arbeitsgruppe	5
	offene Abende	3
	Probleme von LD auf drei Ebenen	0
	Probleme bei den Anwendungsfällen	0
	Akkreditierung	2

	praktischer Aufwand	2
	Berliner Blase	1
	Angepasstheit	3
	Mitgliederverlust und Inaktivität	6
	fehlende politische Anbindung	5
	Probleme auf Ebene der Software	7
	technologische Grenzen	1
	Überforderung	1
	Echo-Chambers	1
	Sicherheit ist nicht gewährleistet	1
	Probleme auf Ebene der Theorie	3
	Komplexität politischer Arbeit unterschätzt	11
	Beteiligungswillen überschätzt	6
	Delegationsmechanismus funktioniert nicht wie erwartet	3
	aufgeladener Begriff	2
	Potenzial von Software überschätzt	2
	nicht zu lösende theoretische Probleme	1
	unbeabsichtigte Folgen von Beteiligung	3
	taktisches Wahlverhalten	1
	Verbindlichkeit vs. Anonymität	3
	Demokratie- und Politikverständnis	16
	Komplexität	6
	Betroffenheit	1
	Anspruch auf Systemverbesserung	6
	politisches Experiment als Ideologie	2
	Solutionismus	4
	Internetzentrismus	1

	Geschichtsvergessenheit	0
	funktionales Demokratieverständnis	0
	Commons und Peer Produktion	4
	Beteiligungsdemokratie	2
	Verhandlungsdemokratie	4
	Machtkonzentration	1
	Entscheidungsfähigkeit	0
	Kompromisse	1
	Diskursdemokratie	4
	Abstimmungsdemokratie	7
	zu schützende Werte der Demokratie	0
	Rechenschaft	2
	Grenzen	1
	Ergebnisoffenheit	1
	Pluralität	1
	Kanalfreiheit	1
	Versammlungsfreiheit	1
	Meinungsfreiheit	1
	Probleme der Demokratie	2
	Mehrheiten	2
	Strukturkonservatismus	1
	fehlendes Selbstwirksamkeitsgefühl	1
	Zynismus	1
	Intransparenz	1
	uncoole und unwichtige Struktur	2
	Lobbying	4
	Skalierbarkeit	6

	Distanz zwischen Repräsentanten und Repräsentierten	1
	Zensur	2
	Unterdrückung	1
	strukturelle Begrenztheit	1
	Sicherheitsprobleme	1
	Programmierprozess	0
	Scrum	5
	Hype	3
	reine Lehre	1
	Zusammenarbeit	1
	Open Source Community	6
	Arbeitsteilung	3
	der Rolle entsprechend	1
	freiwillig	1
	zunehmend	1
	Git	6
	Regeln	1
	Werkzeug	1
	automatische Dateibenennung	1
	Pull Request	1
	Aufgaben	0
	Design entwickeln	0
	Vision entwickeln	1
	definieren, was möglich ist	3
	aufschreiben, was man macht	1
	verstehen, wie Demokratie funktioniert	1
	gemeinsame Sprache finden	3

	Konzept erstellen	3
	mit Kunden sprechen	1
	Evaluieren	1
	Ideen entwickeln	1
	Anforderungen definieren	4
	Vorschläge vorstellen	1
	Planen	1
	kurzfristig und detailliert ausarbeiten	2
	langfristigen Plan erstellen	1
	absprechen, was gemacht wird	3
	Zeit schätzen	2
	Entscheiden, was gemacht wird	3
	rechtliche Fragen klären	1
	konkrete Software-Schritte festlegen	4
	neue Verfahren einführen	1
	Aufnahme von Code in den Kern	1
	Architektur wählen	1
	Code schreiben	8
	Testen	1
	Software betreuen	1
	Anpassen	3
	Bugs fixen	1
	Finanzierung	6
	Forschen	3
	Probleme	3
	Verständnisprobleme	12
	Probleme bei der Umsetzung der Vision in Projekten	32

	Probleme beim Einsatz der Software	3
	Probleme beim Entwickeln der Vision und des Konzepts	24
	Interessenkonflikte	3
	Beteiligungsproblem	6
	Busfaktor	2
	Probleme beim Programmieren der Software	16
	Phasen	1
	Goldene Zeit der Do-ocracy	12
	Eigeninitiative	1
	unbezahlt	1
	gemeinsame Visionen	2
	Phase des Projektwachstums	8
	Diktatur der Aktiven	11
	Enquete-Kommission	1
	mediale Aufmerksamkeit	2
	gemeinsames Entscheiden über Kosten und Zeit	1
	Zeitdruck	2
	mangelnde Qualität	1
	Chaos	2
	teilweise bezahlt	1
	Software as a Service	5
	Geschäftsmodell	1
	Scrum-Prozess	3
	Rollen	0
	Scrum-Rollen	6
	Arbeitsrolle	4
	Administrator	1

	Fundraiser	1
	Projektmanager	2
	Designer	2
	Chef	2
	Mitarbeiter	3
	Kunde	2
	Programmier-Rollen	4
	Lead-Programmierer	2
	Politiker	1
	Vereinsrolle	6
	Vorstand	5
	Fördermitglied	2
	Gründungsmitglied	2
	Vollmitglieder	3
	Ehrenmitglied	2
	Programmierbiografie	0
	Politisierung	5
	Motivation	7
	Überzeugungstäter	12
	Bedingungsloses Grundeinkommen	1
	F/OSS	14
	Wertebezug	3
	Mathematik/Funktion	6
	Selbstwirksamkeit	12
	Programmiertypen	3
	Selbst-Fremdwahrnehmung	14
	beruflicher Softwareentwickler	6
	geborener Programmierer	10