Wissensspeicher, Projektmodell, Organisationstalent oder Schwarzes Loch - GitLab als Management-Tool für DH-Projekte

Bunselmeier, Jennifer
bunselmeier@uni-wuppertal.de
Bergische Universität Wuppertal, Deutschland

Stahn, Lena-Luise
lstahn@uni-wuppertal.de
Bergische Universität Wuppertal, Deutschland
ORCID: 0000-0003-1298-5145

Management Tools in den DH

"Just as Git has influenced the way I approach digital critical editions, the nature of critical editions has influenced the way I think about Git." (Huskey 2023, 11)

Interdisziplinarität in DH-Projekten manifestiert sich häufig in einem großen Team an Mitarbeitenden, nicht selten räumlich verteilt, in jedem Fall aber digital arbeitend, die mit ganz unterschiedlichen Vorstellungen und einer großen Bandbreite an Perspektiven, Denkweisen und "working cultures" aufeinandertreffen (vgl. Edmond 2016). Hier bedarf es einer Vermittlerrolle, die das Projektmanagement übernimmt. Diese Aufgabe fällt häufig der DH-Seite zu, begründet in der dort vorhandenen Doppelqualifikation aus technischer und geisteswissenschaftlicher (Aus-)bildung. Dennoch wird das Thema Projektmanagement in den DH-Curricula kaum abgedeckt: "Bei allen Lehrangeboten ist auffällig, dass ein Fokus auf die Projektplanung und nicht auf das Management einer laufenden Zusammenarbeit gelegt wird. Planungskompetenz ist vorhanden, und vor allem Fähigkeit zur Selbstorganisation, aber eine Moderationskompetenz sowie Fähigkeiten, Techniken und Methoden zur Organisation einer Zusammenarbeit fehlen." (Cremer 2019)

Behelfsmäßig wird dann gerne auf Verfahren und Werkzeuge aus der Software-Entwicklung zurückgegriffen und diese mehr oder weniger erfolgreich adaptiert (vgl. Chan et al. 2017). So hat sich im DH-Bereich der Einsatz von aus der Softwareentwicklung stammenden Werkzeugen und Plattformen bewährt, die auf sogenannter Versionsverwaltung basieren.[1] Git und Anwendungen wie GitHub und GitLab versprechen Transparenz, Reproduzierbarkeit und Nachhaltigkeit[2] (vgl. Nüst 2018; Broyles 2020) und werden in vielen DH-Projekten genutzt (vgl. Huskey 2023, 10).[3] Auch das Versprechen von (und die Forderung nach) Open Source (vgl. Jannidis 2017, 94) und Open Science bzw. Crowdsourcing (vgl. Schöch 2017, 208) scheint damit eingelöst werden zu können.

Literatur und Studien, die den Umgang mit diesen Tools und die Auswirkungen auf die Anwender*innen und den Anwendungskontext betrachten, gibt es kaum.[4] Ansätze wie die auf der DH2016 vorgestellte Studie (Spiro 2016a) zur Evaluierung der GitHub-Nutzung in den DH scheinen weiterführende Untersuchungen nicht inspiriert zu haben.[5] Einer der neuesten Beiträge (Huskey 2023) beschäftigt sich speziell mit der Nutzung der git commit history als Forschungstagebuch. Auf die veränderten Gewichtungen und Rollenverteilungen im Projekt-Ökosystem[6] und mögliche Konsequenzen geht der Autor jedoch nicht ein.

Am Beispiel eines bereits seit 2015 laufenden DH-Projekts stellen wir die Frage, wie sich die Nutzung von git und die geisteswissenschaftliche Arbeit gegenseitig beeinflussen. Denn festzuhalten ist: (fast) jede*r kennt, erwartet und verwendet git, kaum jemand ist sich aber über Potential, Grenzen und Probleme im Klaren.

Erfahrungen aus dem Luhmann-Projekt

Das Akademievorhaben "Niklas Luhmann – Theorie als Passion" (2015-2030)[7] hat die Digitalisierung und Erschließung des wissenschaftlichen Nachlasses des Soziologen Niklas Luhmann zum Ziel. Projekt-Datenmanagement und -Kommunikation erfolgen dabei über eine von der Universitätsbibliothek Bielefeld gehostete GitLab-Instanz.

Der intensive Einsatz der Plattform lässt sich an einigen Zahlen ablesen: Die unter dem Luhmann-Projekt versammelten 57 Repositories haben eine Größe von insgesamt 5,5 GB und umfassen ca. 5.600 issues (2.500 davon "offen", d.h. in Bearbeitung). Die Bandbreite an Dateiformaten umfasst u.a. (TEI-)XML, XSLT, XQuery, XProc, JS, Python, HTML, CSS, MD, Tabellen und Textdokumente sowie Binärformate für Bilddateien (JPG, SVG, Tiff) und Audio (mp3, wav). Gespeichert werden die Objekte in Form von TEI-XML-Dateien selbst sowie Modelle und Skripte zur Qualitätssicherung, außerdem Skripte für die interne Editionsumgebung und für das Back- und Frontend der Digitalen Edition. Hinzu kommen Worddateien, Excelllisten, Bilder u.ä. zur internen Dokumentation. Derzeit sind 21 Mitarbeitende als project member verzeichnet. Die Zahl der pro Tag erfolgten commits ist schwer zu beziffern und für jedes Repository unterschiedlich, es dürfte sich in den Kern-Repositories um eine Zahl im mittleren zweistelligen Bereich handeln.[8]

Neben der Funktion als Datenspeicher wird GitLab von Beginn an als primäres Tool zur Kommunikation, Dokumentation und Aufgabenverteilung genutzt. Systematische Dokumentation geschieht über Markdown und git pages .

Der Austausch darüber, was im Projekt zu besprechen, entscheiden oder erledigen ist, wird über issues geführt. Die Inhalte reichen von Fehlermeldungen über Modellfragen bis hin zu fachwissenschaftlich-editorischen Detaildiskussionen. Durch den nicht verpflichtenden Einsatz von labeln kann eine Kategorisierung der issues vorgenommen werden. Todos werden in unterschiedlichen Granularitätsstufen festgelegt (spezifische wie "Button xy in der Anzeige um 10px nach rechts verschieben" oder allgemeine wie "alle Literaturverweise im edierten Text auszeichnen"). Durch assignments ist jedes issue einer verantwortlichen Person zugewiesen. Das mentioning von Personen stellt eine Möglichkeit dar, Mitarbeitende auf einen Sachverhalt aufmerksam zu machen, ohne dass akuter Handlungsbedarf besteht.

Beobachtungen aus der Anwendung

Die zur Routine gewordene Nutzung von GitLab wird im Arbeitsalltag nur noch selten wahrgenommen. Von Anfang an unterliegt sie keiner Kontrolle, d.h. es werden keine Vorgaben zu issue-Gestaltung, label-Verwendung und Ähnlichem gemacht. Das bietet die notwendige Flexibilität, den unterschiedlichen Ansprüchen gerecht zu werden, führt aber, neben bewährten Best Practices, teilweise auch zu Wildwuchs, wie die folgenden Beispiele verdeutlichen.

Die oft kritisierte, nicht nur für technik-ungeübte Menschen schwer überwindbar scheinende Einstiegshürde der abstrakten git-Arbeitsweise (Chacon & Straub 2014) wird mit der Installation des GitHub Desktop Clients[9] umgangen. Allgemein kommen die Mitarbeitenden so gut mit den relevanten git-Befehlen zurecht. Es kommt selten zu merge-Konflikten, da sich im Zuge der Nutzung auch das Bewusstsein für die Repository-Struktur, das Verhältnis zwischen remote und lokaler Kopie und die Gefahr möglicher Überschneidungen aufgrund fehlender Absprache entwickelt hat.

Die anfallende Maintenance etwa bei Installation oder Update wird durch die eigentliche Stärke von git – Versionierung und Versionskontrolle – ausgeglichen. Die Möglichkeit, "falsche" commits auch nach längerer Zeit rückgängig machen zu können, Fehler durch die Nachvollziehbarkeit von Änderungen schnell zu identifizieren sowie das "Backup-Gefühl", dass durch die zentrale Datenhaltung kein Verlust von lokal gespeicherten Daten droht, werden von den Mitarbeitenden als positiv und rückversichernd wahrgenommen.

Die Bearbeitung eines issues findet im Allgemeinen mit einem commit der bearbeiteten Daten einen Abschluss, ergänzt durch eine verpflichtende stichwortartige commit message. Dieser Workflow weist auf die Funktion der Repositorien als "Forschungstagebuch" hin. Aus den meist ausführlichen Diskussions-issues und der commit history entsteht so über die Zeit neben der expliziten Projektdokumentation eine umfassende implizite Dokumentation der gesamten Arbeit und Forschung: Theoriebildung (z.B. "Was sind unsere Editionsprinzipien?", "Was ist das *Werk*?") findet oft als Ergebnis der Kombination von Problem und daraus folgender fachwissenschaftlicher-technischen Diskussion statt.

Neben dieser Möglichkeit zur freien geisteswissenschaftlichen Arbeit unterstützen issue assignment und todos die tägliche Arbeit, indem Verantwortlichkeiten eindeutig zugewiesen sind. Sichtbarkeit der eigenen Arbeit für sich selbst und andere sowie ein Gefühl des "miteinander Arbeitens" trotz räumlicher Trennung sind weitere positive Aspekte.

Gleichzeitig zeigen sich hier die Schattenseiten. Das Tool birgt die Gefahr, zum Kontrollinstrument zu werden, ob beabsichtigt oder nicht. Indem alle einer Person zugewiesenen Aufgaben, das Ergebnis und der Zeitpunkt ihrer Erledigung jederzeit klar für alle einsehbar sind, wird das GitLab ungewollt zu einer datenschutzrechtlich nicht unproblematischen Kontrollinstanz über Arbeitszeiten und -leistungen. Da die Mitarbeitenden keine Möglichkeit haben, sich gegen die Nutzung dieses Tools zu entscheiden, stehen Arbeitgeber und Projektleitung in der Verantwortung, sicherzustellen, dass die Mitarbeitenden keine arbeitsrechtlichen Konsequenzen oder unangemessene Kontrolle befürchten müssen.

Eine ungefilterte Kommunikation über issue-Kommentare kann zu Missverständnissen führen ("Dienstleistergefühl"), etwa wenn Mitarbeitende sich durch das unabgesprochene Setzen von Zu-Erledigen tasks oder due dates und ungefragtes Kommentieren bevormundet fühlen oder dies als übergriffig empfinden. Der eingangs erwähnte Wildwuchs wiederum ist häufig auf die sehr unterschiedlichen Arbeitsweisen der Mitarbeitenden zurückzuführen, insbesondere hinsichtlich der Strukturierung und Verwendung von issues.

Fehlende guidelines und nicht vorhandene Sachkenntnis haben negative Auswirkungen auf die Verlässlichkeit und Nachhaltigkeit des GitLabs, beispielsweise durch fehlende oder unsachgemäße Nutzung von labeln oder einem mentioning vieler, nicht betroffener Personen, was ein "Aufblähen" der persönlichen todo-Liste zur Folge hat. Nichtssagende commit messages schränken die Dokumentationsfunktion der commit history ein.[10]

Die Kommentarfunktion und freie Gestaltbarkeit sind ein unverzichtbares Mittel, um Ideen, Konzepte und Theorien entwickeln zu können. Die Folge sind jedoch viele bis zur Undurchdringlichkeit angewachsene issues, in denen zu viele Themen unstrukturiert "zusammengesammelt" worden sind. Groß ist zudem die Verlockung, das GitLab als persönliches Notizbuch zu gebrauchen, wobei mitunter die Einsehbarkeit für die anderen Projektmitglieder vergessen wird. Diese Nutzung als "ausgelagertes Gehirn" des Projekts macht das GitLab zwar zum zentralen Forschungsdatenrepositorium, aber beim Versuch, dessen effiziente (Nach-)Nutzung zu gewährleisten, prallen die Notwendigkeit, eine möglichst freie und flexibel nutzbare Arbeitsplattform bereitzustellen, und der Bedarf nach Ordnung durch Moderation und Vorgaben spürbar aufeinander.

Lessons Learned

Eine effektive Nutzung von GitLab als Projektmanagement-Tool erfordert vor allem eine strikte Trennung der verschiedenen issue-Typen (Diskussion, Bug, etc.). Die Implementierung von issue templates, eine standardisierte label-Vergabe und Regeln darüber, zu welchem Zeitpunkt ein Thema in ein neues issue ausgelagert wird, können helfen, Redundanz zu verringern und die Qualität der Dokumentation zu verbessern. Überlange issues sollten vermieden oder die Ergebnisse übersichtlich dokumentiert werden. All dies erfordert ein Bewusstsein für den hohen Pflegeaufwand, für den personelle Ressourcen eingeplant und ausdrücklich freigestellt werden müssen.

Fazit

Wir haben eingangs die Frage gestellt, wie ein Werkzeug, das auf Produktion und Softwareentwicklung ausgerichtet ist, den Umgang der Projektbeteiligten miteinander und mit dem eigenen Arbeiten beeinflusst. Unsere Beobachtungen zeigen, dass es eine DH-spezifische Nutzung von git gibt. Macht erst – und gerade – diese nicht intendierte geisteswissenschaftliche Nutzung aus Plattformen wie GitLab ein – adäquates – Projektmanagement-Tool? Eine GitLab-Instanz kann ein Forschungstagebuch darstellen und als solches bedeutende Einblicke in die Entwicklung und Theoriebildung von Projekten bieten, deren Veröffentlichung die Forschung nachhaltig bereichern würde. Diese Anwendung erfordert jedoch sorgfältige Planung, konstante Pflege und kommt nicht völlig ohne Regeln aus. Um allgemeingültige Empfehlungen für die DH-Community entwickeln und Best Practices etablieren zu können, bedarf es allerdings weiterer Erfahrungsberichte aus der DH-Praxis, die zeigen, wie Werkzeuge dieser Art mit Erfolg eingesetzt werden.

Fußnoten

1. Zur Eignung speziell von GitHub und Versionierungsverwaltung als Digital Library s. die Studie von Minkova (2018).
2. Git (https://git-scm.com/), GitHub (https://github.com/), GitLab (https://gitlab.com/).
3. Dazu beitragen dürfte der Umstand, dass git gerade Text-basierte Dateiformate besonders gut verarbeiten kann. Vgl.Huskey 2023, 19, und Spiro & Smith 2016, Folie 20.
4. Eine etwas ältere Übersicht (Stand 2019) bietet F. Cremer in seinem Blogpost (Cremer 2019), in der er die im deutschsprachigen kaum existente und im englischsprachigen Raum vorwiegend auf Projektplanung ausgerichtete Literatur beklagt. Das gerade erschienene Werk von Cremer et al. 2024 beschäftigt sich ausführlicher mit scrum, einem Modell für das agile Projektmanagement, geht jedoch auch nicht weiter auf die kollaborative Arbeit mit git ein.
5. Die Ergebnisse dieser speziell auf GitHub ausgerichteten Studie umfassten Bedenken bez. Geschäftsmodell und Langzeitsicherung der Daten einerseits, die hohe Einstiegshürde beim Erlernen von git sowie die relative Ungeeignetheit für nicht-textliche Daten andererseits. Vgl. dazu auch Spiro & Smith 2016 und den Blogartikel Spiro 2016b.
6. Zur Rollenverteilung in kollaborativen DH-Projekten s. auch Gardiner & Musto 2015, 43-66. Komprecht und Röwenstrunk (2016, 519) weisen auf die Hürden bei der Festlegung von Zuständigkeiten und die Bedeutung einer “offene[n] und kollegiale[n] Informationspolitik” hin.
7. Siehe das laufende Portal "Niklas Luhmann-Archiv" unter https://niklas-luhmann-archiv.de/ .
8. Als Bsp. für ein derzeit stark in Bearbeitung befindliches Repository ("manuskripte") ergibt eine git log-Abfrage (für den Zeitraum 11. und 12.07.24) 42 commits von insgesamt 4 bearbeitenden Personen.
9. Projektwebseite, Download und Dokumentation unter https://github.com/apps/desktop .
10. Vgl. dazu "But that in itself does not solve the problem of haphazard record-keeping. In the absence of any project guidelines for using Git, contributors can make as few or as many commits as they please. And since there are few rules about the content of a commit message, it is possible, and unfortunately common, for contributors to do the bare minimum by writing generic messages (e.g., "Adding some files" or "Fixed a bug") without any further documentation, greatly reducing the usefulness of a project’s log [...]." Huskey 2023, 14.

Bibliographie

Broyles, Paul A. 2020. Digital Editions and Version Numbering. *Digital Humanities Quarterly*, 14(2). http://www.digitalhumanities.org/dhq/vol/14/2/000455/000455.html (Zugriff: 09.07.24).

Chan, Anela, Richard Chenhall, Tamara Kohn und Carolyn Stevens. 2017. "Interdisciplinary collaboration and brokerage in the digital humanities." *Digital Humanities Quarterly*, 11(3). http://digitalhumanities.org:8081/dhq/vol/11/3/000336/000336.html (Zugriff: 13.07.24).

Chacon, Scott, und Ben Straub. 2014. *Pro git*. Berlin: Springer Nature. https://git-scm.com/book/en/v2 (Zugriff: 12.07.24).

Cremer, Fabian. 19. März 2019. "Gottes Werk und Teufels Beitrag: Ein Essay zu Digital Humanities und Projektmanagement." *DHdBlog*. https://dhd-blog.org/?p=11283 (Zugriff: 11.07.24).

Cremer, Fabian, Swantje Dogunke, Anna Maria Neubert, Thorsten Wübbena. 2024. *Projektmanagement und Digital Humanities. Zur klugen Gestaltung der Zusammenarbeit.* Bielefeld: Bielefeld University Press.

Edmond, Jennifer. 2016. "Collaboration and Infrastructure". In *A new companion to digital humanities*, hg. von Susan Schreibman, Ray Siemens, und John Unsworth, 54-65. Chichester, West Sussex: John Wiley & Sons.

Gardiner, Eileen, und Ronald G. Musto. 2015. *The digital humanities: A primer for students and scholars*. New York, NY: Cambridge University Press.

Huskey, Samuel J. 2023. "Committing to reproducibility and explainability: using git as a research journal." *International Journal of Digital Humanities* (2024) 6: 9-21. https://link.springer.com/article/10.1007/s42803-023-00076-9 (Zugriff: 11.07.24).

Jannidis, Fotis. 2017. "Grundbegriffe des Programmierens". In *Digital Humanities*, hg. von Fotis Jannidis, Hubertus Kohle und Malte Rehbein, 68-95. Stuttgart: J.B. Metzler.

Komprecht, Anna Maria, und Daniel Röwenstrunk. 2016. "Projektmanagement in digitalen Forschungsprojekten. Ein Leitfaden für interdisziplinäre und kooperative Drittmittelprojekte im Umfeld Digitaler Editionen." In *'Ei, dem alten Herrn zoll'ich Achtung gern'. Festschrift für Joachim Veit zum 60. Geburtstag*, hg. von Kristina Richts und Peter Stadler., 509-522. München: Allitera Verlag.

Minkova, Miglena. 2018. "Github as a digital library." MA Thesis. https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1305693&dswid=-7777 (Zugriff: 17.07.24).

Nüst, Daniel. 2018. “git für DH.” Zenodo. https://doi.org/10.5281/zenodo.1299031. (Zugriff: 11.07.24).

Schöch, Christoph. 2017. "Crowdsourcing". In *Digital Humanities*, hg. von Fotis Jannidis, Hubertus Kohle und Malte Rehbein, 206-212. Stuttgart: J.B. Metzler.

Spiro, Lisa. 2016a. "Evaluating GitHub as a Platform of Knowledge for the Humanities." In *Digital Humanities 2016: Conference Abstracts*. Jagiellonian University & Pedagogical University, Kraków: 688-690. https://dh2016.adho.org/abstracts/225 (Zugriff: 11.07.24).

Spiro, Lisa. 11.05.2016b. "Studying How Digital Humanists Use GitHub." *Digital Scholarship in the Humanities. Exploring the digital humanities.* https://digitalscholarship.wordpress.com/2016/05/11/studying-how-digital-humanists-use-github/ (Zugriff: 11.07.24).

Spiro, Lisa und Sean Morey Smith. 2016. "Evaluating GitHub as a Platform of Knowledge for the Humanities." Präsentation. https://pdfs.semanticscholar.org/487a/ec5a6a15adc5f03c90ddf3c84e29ab7f2358.pdf (Zugriff: 11.07.24).