Adrian Pohl

Software entwickeln und betreiben

Abstract: In vielen insbesondere größeren Wissenschaftlichen Bibliotheken wird Software entwickelt und betrieben. Wissenschaftliche Bibliothekar:innen besitzen wichtige Kompetenzen, um wertvolle Beiträge zu Entwicklung und Betrieb von Software zu leisten. Denn für Entwicklung, Betrieb und Instandhaltung von Software, die dauerhaft erfolgreich sind, müssen viele verschiedene Aufgaben erledigt und unterschiedlichste Rollen übernommen werden. Der Beitrag stellt zur besseren Einordnung der Aufgaben die verschiedenen Schritte im Prozess der Softwareentwicklung dar und erläutert dann die Aufgaben und Rollen, die Bibliothekar:innen in diesem Prozess übernehmen können. Dabei konzentriert er sich auf Arbeiten, die selbst nicht Programmier-, Deployment- oder Administrationsaufgaben sind. Schließlich wird erläutert, welche Kompetenzen für die Erledigung der genannten Aufgaben hilfreich sind und welche Herausforderungen bei der Arbeit entstehen können.

Keywords: Software-Entwicklung, Software-Betrieb, Software-Maintenance, Entwicklungsprozesse

Kurzbiografie: Adrian Pohl arbeitet seit 2008 am Hochschulbibliothekszentrum des Landes Nordrhein-Westfalen (hbz). Er hat Kommunikationswissenschaften und Philosophie an der RWTH Aachen studiert und ist Absolvent des ersten MALIS-Jahrgangs an der TH Köln (MALIS09). Seit 2019 leitet er die Gruppe „Offene Infrastruktur“ (neuerdings „Metadateninfrastruktur“), die sich mit offenen Standards, Software-Werkzeugen und Prozessen für die Publikation strukturierter Daten im Web befasst. Die Gruppe arbeitet – teils in Kooperation mit anderen Institutionen – an einer Reihe von Online-Diensten und Software-Werkzeugen, u. a. lobid, Metafacture, SkoHub, OERSI (gemeinsam mit der TIB) und NWBib. Adrian Pohl ist Co-Chair des Programmkomitees der Konferenz „Semantic Web in Libraries“ (SWIB). Kontakt: pohl@hbz-nrw.de

Hintergrund und Einleitung

Mit einem geisteswissenschaftlichen Hintergrund und einem Platz im ersten MALIS-Studiengang an der TH (damals: FH) Köln bin ich Ende 2008 im Hochschulbibliothekszentrum des Landes Nordrhein-Westfalen (hbz) gelandet, wo ich seitdem gearbeitet habe. Im Laufe dieser Zeit habe ich an verschiedenen Open-Source-Software-Projekten mitgearbeitet. Ich bezeichne mich selbst als „Web Librarian“ und sehe mich durchaus auch als Softwareentwickler, ein Programmierer bin ich nicht. Dennoch habe ich teilweise essentielle Rollen im Entwicklungsteam übernommen, denn für die

https://doi.org/10.1515/9783110790375-033

Softwareentwicklung werden bekanntermaßen nicht nur Programmierkenntnisse benötigt.[1]

Auch in vielen größeren wissenschaftlichen Bibliotheken findet mittlerweile Softwareentwicklung statt, sei dies eine federführende Entwicklung innerhalb eines eigenständigen Software-Projekts oder Beiträge zu Open-Source-Projekten wie Folio, VuFind, Kitodo, Catmandu, Annif, Vivo, dem International Image Interoperability Framework (IIIF) und IIIF-Viewer wie Mirador, zu Repository-Software wie Dspace, Fedora oder OPUS. Wissenschaftliche Bibliothekar:innen[2] besitzen wichtige Kompetenzen, um wertvolle Beiträge zu Entwicklung und Betrieb von Software zu leisten. Es müssen schließlich eine Vielzahl verschiedener Aufgaben erledigt und unterschiedlichste Rollen besetzt werden, damit Entwicklung, Betrieb und Instandhaltung von Software dauerhaft erfolgreich sein können. In diesem Beitrag möchte ich einige dieser Aufgaben und Rollen beschreiben und welche Kompetenzen dafür hilfreich sind.[3]

Softwareentwicklung als Prozess

Zur besseren Einordnung der Aufgaben werde ich zunächst verschiedene Schritte im Prozess der Softwareentwicklung darstellen. Es gibt eine Menge verschiedener Methodologien der Softwareentwicklung, am bekanntesten sind die Agile Entwicklung und andere iterative Entwicklungsprozesse sowie das sogenannte Wasserfallmodell. Unabhängig von der Methodologie und von der Frage, ob ich bloß eine einzelne zusätzliche Funktion oder ein gesamtes Softwaresystem entwickle, lassen sich grob folgende Schritte der Softwareentwicklung und -maintenance unterscheiden:

– Planung und Priorisierung: Die Entwicklungsressourcen einer Einrichtung oder eines Teams sind immer endlich. Bevor mit der Arbeit an der (Weiter-)Entwicklung einer Software begonnen werden kann, muss ein Überblick über alle anstehenden Entwicklungsaufgaben erstellt, deren Wichtigkeit und Dauer abgewogen und schließlich die als nächstes umzusetzenden Aufgaben priorisiert werden.

---

1 Ein großer Dank und Grüße gehen an die Kolleg:innen in meiner Gruppe „Metadateninfrastruktur“: Pascal Christoph, Fabian Steeg, Katinka Tauber, Tobias Bülte, Steffen Rörtgen, Anna Keller und Phu Tu, mit denen ich in gemeinsamen Gesprächen und Diskussionen über Softwareentwicklung und bei der Entwicklung gemeinsamer Prozesse eine Menge gelernt habe und sicher noch Vieles lernen werde.

2 Im Folgenden werde ich der Kürze halber häufig auch nur von Bibliothekar:innen sprechen, wobei ich mich primär auf Wissenschaftliche Bibliothekar:innen beziehe.

3 Meine Zusammenarbeit mit Softwareentwickler:innen und Administrator:innen war meist durch große Kontinuität gekennzeichnet. Die Arbeit an lobid begann in einem Zweierteam mit Unterstützung durch einen Entwickler. Die meiste Zeit arbeitete ich in einem Dreierteam, zusammen mit zwei Softwareentwicklern, die auch Web-Administrationsaufgaben übernahmen. Seit 2020 wuchs das Team beständig, erst auf fünf und bis Anfang 2023 auf insgesamt neun Personen, von denen drei bis vier einen bibliothekarischen Hintergrund haben. Ich habe kaum Erfahrung mit projektorientierter Arbeit in wechselnden Teams. Einige Projekte haben wir mit der Unterstützung Externer durchgeführt, mit denen wir aber meist bereits vertraut waren. Dies sei bei der Lektüre dieses Textes bedacht.

- Anforderungsanalyse: Ermittlung und Dokumentation der umzusetzenden Anforderungen, z. B. in Form eine Backlogs von Tickets, die Anforderungen als User Stories beschreiben.
- Systementwurf: Insbesondere wenn ein größeres Entwicklungsprojekt gestartet wird, geht der eigentlichen Entwicklung eine Designphase voraus, in der eine Systemarchitektur aufgrund der ermittelten Anforderungen bestimmt wird.
- Implementierung: Hierbei handelt es sich um die eigentliche Programmierung, in deren Kontext in der Regel auch automatisierte Tests entwickelt und ausgeführt werden, etwa zur Überprüfung von Code-Qualität, der Funktionalität auf Code-Ebene (Unit-Tests), sowie der Integration des Codes mit bestehenden Softwarekomponenten.
- Begutachtung: Die Begutachtung der entwickelten Software kann in zwei Schritte unterteilt werden: 1. Funktionale Begutachtung, d. h. das Ausprobieren in einer Testumgebung, ob die gewünschten Änderungen an Daten, User-Interface (UI) und/oder Schnittstelle tatsächlich umgesetzt sind und dabei keine unliebsamen Veränderungen mit eingeführt wurden. Sollten Probleme festgestellt werden, geht die Arbeit zurück an die Entwickler:in zur Anpassung. 2. Begutachtung des Codes durch eine:n zweite:n Entwickler:in (Code-Review).
- Deployment: Sind die Anforderungen umgesetzt und die Begutachtung durchgeführt, kann das Ergebnis zur Nutzung bereitgestellt – deployt – werden. Dies kann bedeuten, dass eine neue Version der Software paketiert und veröffentlicht wird oder – bei Webanwendungen – dass das aktualisierte System auf dem Produktionsserver installiert wird.
- Monitoring und Maintenance: Insbesondere bei Webanwendungen ist das System nach dem Deployment zu überwachen (Monitoring) und bei Ausfällen, Bug-Meldungen oder anderen Problemen sind entsprechende Korrekturen umzusetzen. Auch bei länger problemlos laufenden Systemen bleiben Maintenancearbeiten nicht aus. So sind etwa – zumindest sicherheitsrelevante – Versionsupdates der verwendeten Softwarebibliotheken, -komponenten und Programmiersprachen einzuspielen und eventuell neu zu integrieren.

Wissenschaftliche Bibliothekar:innen können theoretisch in allen Phasen dieses Prozesses aktiv werden. Hier geht es allerdings um jene Aufgaben, die zwar eine wichtige Begleitung für institutionelle Softwareentwicklungsprojekte darstellen, selbst aber *nicht* direkt mit dem Schreiben von Software-Code (4.), Deployment (6.) oder Administration (7.) zu tun haben.

Aufgaben und Rollen

Im Folgenden werde ich einige nicht-programmiertechnische zentrale Aufgabenbereiche und Rollen der Softwareentwicklung benennen und näher betrachten.

Product Owner

Bei einem Entwicklungsprojekt oder auch bei bereits produktiven IT-Dienstleistungen ist die Benennung eines Product Owners sinnvoll. Der Terminus kommt aus der agilen Entwicklung, konkret dem Scrum-Framework. Aber auch bei einem weniger formalisierten oder nicht ganz Scrum-konform gestalteten Prozess ist es hilfreich, wenn eine Person die Rolle eines Product Owners übernimmt. Product Owner stellen die Schnittstelle zwischen Nutzer:innen und Entwickler:innen dar. Sie sind somit die Garanten dafür, dass die Entwicklung tatsächlich nutzer:innen-orientiert stattfindet. Auch in der Bibliothekswelt haben wir erfahren, wie wichtig es ist, eine Software nicht auf Basis bloßer Annahmen über Endnutzer:innen und ihrer Bedürfnisse zu gestalten. Product Owner übernehmen insbesondere folgende Aufgaben:

- Anforderungsermittlung und -analyse: Im Gespräch mit Nutzer:innen, Auftraggeber:innen und anderen Stakeholder:innen ermitteln Product Owner kontinuierlich über den ganzen Entwicklungsprozess hinweg die Bedarfe der Nutzer:innen und die Anforderungen an die zu entwickelnde Software. Dabei können verschiedene Methoden verwendet werden, etwa die Definition von Personas, um die verschiedenen Nutzer:innentypen und Anwendungsfälle der Software für alle Beteiligten klarer zu fassen. Konkrete Anforderungspakete werden schließlich in Tickets, häufig in Form von User Stories, festgehalten. Die Liste aller Anforderungen kann in einem sogenannten Backlog gesammelt werden.
- Backlog-Pflege: Zentrale Aufgabe von Product Owner:innen ist die Backlog-Pflege. Dies umfasst zum einen die Priorisierung der Entwicklungsaufgaben, um zu Beginn des Projekts früh einen funktionieren Prototyp zu haben, anhand dessen wiederum Nutzer:innen-Feedback für eine erneute Priorisierung eingeholt werden kann. Zum anderen bedeutet Backlog-Pflege die Übersetzung von User Stories in konkrete Entwicklungsaufgaben. Dies kann etwa bedeuten, die Aufgaben zu konkretisieren oder zusätzliche Tickets anzulegen.
- Funktionales Review: Funktionales Review bedeutet die Abnahme von Neuentwicklungen im Lichte der zuvor definierten Anforderungen. Erst wenn das funktionale Review bestanden ist, werden Änderungen aus dem Entwicklungssystem in das Produktionssystem überführt.

Softwaredesign, Datenmodellierung, Standardisierung

Neben der Rolle als Schnittstelle zwischen Entwickler:innen und Nutzer:innen bringen Bibliothekar:innen auch wichtige fachliche Kompetenzen mit, mittels derer sie eine wichtige Rolle bei der grundsätzlichen Gestaltung einer Software spielen können. Dabei beziehe ich mich auf Kompetenzen im Bereich Metadaten, die in der Erfahrung mit der Lektüre und dem Verstehen von Metadatenstandards wurzeln sowie in deren Anwendung etwa in Katalogisierungsumgebungen. Auch eigene Erfahrungen in der

Entwicklung von (Metadaten-)Standards sowie Kenntnisse von allgemeinen Webstandards und ihren Implementierungen sind hier sehr hilfreich. Entsprechendes Know-how befähigt Bibliothekar:innen etwa zur Erledigung folgender Aufgaben:

- Datenmodellierung und -validierung: Das Datenmodell ist in der Regel der konzeptuelle Kern einer Software und in seiner Bedeutung kaum zu überschätzen. Es definiert unter anderem die in einem System erfassten Typen von Dingen, die an sie geknüpften Datenfelder und deren Inhalte. Mit Kenntnissen in RDF Schema (RDFS) und Web Ontology Language (OWL) etwa können Bibliothekar:innen direkt maschinenlesbare Datenmodelle erstellen. Wenn für die Datenmodellierung Validierungssprachen wie JSON Schema oder OpenAPI zum Einsatz kommen, kann das Modell direkt zum Aufbau der Software beitragen, nicht nur durch die Validierung von Instanzdaten und API-Ansprachen, sondern etwa auch durch die halbautomatische Generierung von Eingabeformularen. Für die Erstellung und Pflege von kontrollierten Vokabularen in einer maschinenlesbaren Form eignet sich bestens das Simple Knowledge Organization System (SKOS). SKOS-kodierte kontrollierte Vokabulare können von Entwickler:innen direkt in die Software integriert werden. So wird die Entwicklung des Datenmodells zu *der* Aufgabe, wo sich Kompetenzen von Bibliothekar:innen und Entwickler:innen die Hand geben. Die Datenmodellierung muss sich selbstverständlich auch zuvorderst an den ermittelten und definierten Anforderungen orientieren und sollte darüber hinaus im besten Fall bestehende Metadatenstandards nachnutzen und ggf. erweitern. Gleichzeitig muss es flexibel genug für etwaige zukünftige Erweiterungen sein.
- URL Design: Bibliothekar:innen, die mit den grundlegenden Funktionsprinzipien des World Wide Web vertraut sind und beim Browsen auch mal ein Auge auf die konkrete Anwendung der grundlegenden Web-Technologien werfen (Hypertext Transfer Protocol, http und Uniform Resource Locator, URL), bringen wertvolle Kenntnisse für die Web-Entwicklung mit. Bei einer jeden Web-Anwendung spielt das Design der URLs eine zentrale Rolle für die Nutzbarkeit, Erweiterbarkeit und Langlebigkeit des Webauftritts – ganz gleich, ob die Anwendung sich vornehmlich an Menschen richtet oder es sich um eine Web-Schnittstelle zur Anwendungsentwicklung (API, Application Programming Interface) handelt.
- Standardisierung: Bibliothekarische Web-Anwendungen müssen meist für einen langfristigen und verlässlichen Betrieb ausgelegt sein bei möglichst geringen Instandhaltungsaufwänden. Dazu sollen sie im besten Fall so gebaut sein, dass sie Interoperabilität mit anderen Anwendungen unterstützen. Dafür sind standardisierte Schnittstellen und Daten die beste Voraussetzung, zumindest eine klare und leicht nutzbare Dokumentation ist nötig. Aus diesem Grund sind Kenntnisse von Web-Standards (allen voran HTTP), Datenstrukturen und -formaten (JSON, RDF, YAML, XML, CSV etc.), (Meta)datenstandards (schema.org, Dublin Core, Bibframe) und standardisierter bzw. verbreiteter APIs (z. B. International Image Interoperability Framework IIIF oder OpenRefine Reconciliation Service API) unerlässlich für die Entwicklung und das Testen von Webanwendungen.

Darüber hinaus sind Erfahrungen mit Standardisierungsprozessen und -Tools häufig sehr nützlich. Zum einen kann so an der (Weiter-)Entwicklung von Standards mitgearbeitet werden – etwa im Rahmen des W3C, von IIIF oder der Dublin Core Metadata Initiative –, zum anderen kann – sollte kein passender Standard vorhanden sein – die eigene Entwicklung offen, transparent und klar dokumentiert werden, so dass sie von anderen leicht genutzt und übernommen werden kann. Perspektivisch kann eine solche eigene Lösung sogar zu einem neuen Standard ausgebaut werden.

Weitere mögliche Aufgaben

Datentransformation

In vielen bibliothekarischen Softwareprojekten spielt die Transformation von Daten aus einem bestimmten System auf ein Zielformat eine wichtige Rolle: insbesondere bei Systemmigrationen, Aggregationsprojekten wie der Erstellung eines Suchindex, bei Datenaufbereitungs- und Datenanreicherungsprojekten sowie bei Datenanalyseprojekten. Wie im vorherigen Abschnitt erläutert, bringen Bibliothekar:innen, insbesondere Data Librarians, mit ihren Erfahrungen in Bezug auf Metadaten und deren Erfassung, essentielles Wissen und Know-how mit. Aus diesem Grund ist es ratsam, Datentransformationsaufgaben direkt von Bibliothekar:innen erledigen zu lassen. Software zur Datentransformation wie OpenRefine, Metafacture oder Catmandu richtet sich explizit auch an Nicht-Programmierer:innen.

Usabiblity Tests

Die Ergebnisse von Software-Entwicklungsprojekten sollten idealerweise bereits in einer frühen Phase durch Usability-Tests begleitet werden. Dadurch wird die nutzer:innen-orientierte Entwicklung sichergestellt. Es gibt verschiedene Methoden der Testdurchführung. Erfahrungen in der Rekrutierung von passenden Testpersonen und der Durchführung der Tests sind sehr hilfreich, insbesondere für Projekte, die sich an Endnutzer:innen richten.

Prozesse aushandeln und dokumentieren

In jedem Entwicklungsprojekt, sei es durchgeführt im Rahmen einer stabilen Organisationseinheit oder irgendeiner anderen Kooperation, spielt die Einigung auf einen gemeinsamen Prozess und die klare Dokumentation desselben eine wichtige Rolle. Im Idealfall werden Verfahrensfragen durch einen Blick in ein offenes Prozessdokument

geklärt.[4] Bei neu aufkommenden Fragen, die durch das Prozessdokument nicht beantwortet werden, sollte das Dokument entsprechend verbessert werden.

Das gemeinsame Festlegen auf einen Prozess verlangt einiges an Kommunikation, die – je nach Team – mehr oder weniger moderiert von statten gehen muss. Hier können Bibliothekar:innen aktiv werden.

Dokumentation & Support

Wie oben bereits geschrieben spielt die Erstellung und Aktualisierung von Dokumentation eine grundlegende Rolle für den Erfolg eines Softwareprojekts. Ob es sich um das Datenmodell, die Schnittstellen, oder auch den Entwicklungsprozess und die Regeln für potentiell Beitragende handelt: Eine gute Dokumentation ist das A und O. Es ist alles andere als trivial, einen Zustand guter Dokumentation zu erreichen und vor allem auch zu halten. Bibliothekar:innen können hier eine wichtige Rolle spielen, indem sie Dokumentation schreiben und Nutzer:innen-Feedback sammeln, um schnell auf Unzulänglichkeiten in der Dokumentation zu reagieren.

Öffentlichkeitsarbeit, Marketing und Community Management

Für viele Softwareprojekte und Webangebote ist eine Öffentlichkeitsarbeit sinnvoll oder sogar notwendig, etwa wenn es sich um ein Open-Source-Projekt mit einer überinstitutionellen Community handelt. Die Bandbreite der Aufgaben reicht von der Gestaltung und Pflege eines Webauftritts über das Verfassen von Blogbeiträgen bis hin zur Betreuung von Social-Media-Konten im Fediverse oder auf kommerziell-proprietären Plattformen wie Facebook oder Twitter. Im Kontext von lobid hat sich zum Beispiel gezeigt, dass einige Nutzer:innen Bugs, Daten- oder Verfügbarkeitsprobleme gerne und wiederholt über Microblogging-Dienste melden, vorausgesetzt natürlich, sie bekommen zeitig eine Rückmeldung auf ihre Anfrage. Schließlich können Vorträge und Fachartikel dazu dienen, auf eine Software oder einen Dienst aufmerksam zu machen und neue potenzielle Nutzer:innen neugierig zu machen. Dazu kommt das Angebot von Workshops und Tutorials für Menschen, die eine Softwarelösung bereits verwendet haben oder daran interessiert sind. So können die Vorteile, Anwendungsfälle und Funktionen einer Software praktisch vermittelt werden.

Bei Open-Source-Projekten mit einer überinstitutionellen Entwickler:innen-Community kommen Aufgaben im Bereich Community-Entwicklung und -Management dazu.

---

4 Vergleiche etwa das für alle Entwicklungsarbeiten im Kontext von Metafacture gültige Dokument: Tauber, Katinka [u. a.]. (2021): Contributing to Metafacture. https://github.com/metafacture/metafacture-core/blob/master/CONTRIBUTING.md (22.12.2022).

Projektmanagement und Führung

Über die genannten Aufgaben hinaus können Bibliothekar:innen natürlich auch die allgemeine Rolle des Projektmanagers/der Projektmanagerin in einem Softwareprojekt übernehmen. Da es sich dabei nicht um eine IT-spezifische Rolle mit IT-spezifischen Aufgaben handelt, gehe ich hier darauf nicht näher ein.

Kompetenzen

Die oben näher betrachteten Aufgaben von Wissenschaftlichen Bibliothekar:innen in der Software-Entwicklung erfordern allesamt Kompetenzen in den allgemeinen Tätigkeitsbereichen *Zuhören und Kommunizieren*, *Konzipieren und Planen*, *Organisieren und Steuern*, *Festhalten/Erfassen und Prüfen*. Im Folgenden werde ich einen näheren Blick auf eine Schlüsselkompetenz in der Softwareentwicklung richten und abschließend ein paar fachliche Kernkompetenzen erläutern.

Schlüsselkompetenz schriftliche Kommunikation

Softwareentwicklung ist von vorne bis hinten ein schriftlicher oder zumindest schriftlich unterstützter Prozess: Bei allen oben genannten Schritten spielt Verschriftlichung – kombiniert mit Diagrammen, Mockups und anderen Visualisierungen – eine fundamentale Rolle: Die Programmierung selbst ist das Verfassen eines Textes, des Source-Code, Anforderungen werden be*schrieben*, Ergebnisse von Tests und Begutachtungen schriftlich dokumentiert. Auch das Projektmanagement dreht sich heutzutage meist um die Manipulation von Schrift, wenn etwa schriftlich formulierte Anforderungen auf Karten in einem Kanban-Board verschoben werden. Ebenso geschieht die oben erwähnte Dokumentation der Anwendung für Nutzer:innen zum Großteil schriftlich. Vor diesem Hintergrund ist die Fähigkeit zu einer klaren, offenen und für alle Stakeholder transparenten schriftliche Kommunikation essentiell für alle, die in einem Softwareprojekt mitarbeiten.

Dieser Punkt bekommt vor allem auch deshalb hier einen eigenen Abschnitt, weil er den Schlüssel darstellt für Softwareprojekte, die möglichst nachhaltig und zukunftsfähig sein sollen, weil sie nach der initialen Entwicklung über Jahre instandgehalten, aktualisiert und bei Bedarf weiterentwickelt werden müssen. Eine gute schriftliche Kommunikation, die mit größtmöglicher Klarheit Anforderungen an die Software, Diskussionen zu Implementierungsfragen, Entwicklungsfortschritte und Fehlentscheidungen, auftretende Probleme und deren Lösungen festhält, ist eine wertvolle Quelle der Information, und zwar für alle, die jetzt und zukünftig an der Software arbeiten. Ich finde mich regelmäßig dabei wieder, wie ich zur Lösung eines aktuellen Problems in Tickets, Blogbeiträgen oder Dokumentation stöbere, die wir teilweise zehn Jahre vorher verfasst haben.

Gut abgestimmte und eingespielte Prozesse der schriftlichen Kommunikation haben sich auch – insbesondere im Laufe der Covid-19-Pandemie – als großer Vorteil für die Organisation von gemeinsamer Arbeit in verteilten und asynchron arbeitenden Teams herausgestellt.

Die Notwendigkeit und Bedeutung mündlicher Kommunikation soll durch diesen Fokus auf Schriftlichkeit nicht in Abrede gestellt werden, etwa in regelmäßigen Treffen im Entwicklungsteam sowie im Austausch mit Partner:innen und Nutzer:innen. Aber auch hier ist das schriftliche Festhalten der Ergebnisse und der anfallenden Aufgaben unabdingbar.

Fachliche Kompetenzen

Als fachliche Kompetenzen haben sich bereits im vorherigen Kapitel einige im Bereich Metadaten- und Webstandards herausgeschält. Dazu gehört zum Beispiel die Fähigkeit der Recherche und schnellen Orientierung in der Menge der existierenden Web- und Metadatenstandards, um die für eine Anwendung relevanten Standards zu identifizieren. Daran anknüpfend ist wichtig, die formalen Dokumente zur Spezifikation der Standards sowohl kursorisch erfassen zu können als auch anhand einer tiefgehenden Lektüre die Implementierung der Standards unterstützen zu können.

Dazu kommen Grundkenntnisse im Umgang mit der Unix-Shell (z. B. Bash) und Versionskontrollsystemen (insbesondere Git) sowie grundlegende Fähigkeiten in der Nutzung von Softwareentwicklungsplattformen wie GitHub, GitLab oder Forgejo/Gitea.

Herausforderungen

Zum Ende möchte ich einige Herausforderungen nennen, mit denen Bibliothekar:innen, die in der Softwareentwicklung arbeiten, konfrontiert werden können.

- Prinzipien leben: Prinzipien wie Offenheit und Transparenz lassen sich leicht proklamieren, sie aber tatsächlich im Alltag wo immer möglich umzusetzen, kann durchaus eine Herausforderung sein. Bis heute wird in der Bibliothekswelt oft Insiderwissen gepflegt und als persönlicher oder institutioneller Vorteil verstanden. Transparenz geht häufig nicht über die Mitglieder eines Gremiums hinaus und exkludiert so alle, die nicht Teil dieser Gruppe sind, von möglicherweise relevanten und hilfreichen Informationen. Ein solches Verhalten muss erst verlernt werden, um sämtliche Kommunikation, die nicht aus gutem Grund vertraulich sein muss, öffentlich zu dokumentieren. In diesem Umlernprozess lässt sich viel von Open-Source-Foundations und Organisationen, wie dem World Wide Web Consortium (W3C), und ihren Prozessen lernen.
- Erwartungshaltung der Nutzer:innen: Nutzer:innen erwarten Zuverlässigkeit und Stabilität von einer Software. Sie nutzen und orientieren sich dabei häufig an ih-

ren Erfahrungen mit Diensten, hinter denen millionen- oder milliardenschwere Unternehmen stehen. Mit einer solchen Haltung konfrontiert zu werden, kann leicht den bereits existierenden Stress vergrößern, gerade wenn der Betrieb eines Web-Angebots in der eigenen Institution gestört ist.
- Einbeziehung in Datenmodellierung durchsetzen: Von Programmierer:innen zu verlangen, Konfigurationen und Prozesse so aufzusetzen und zu kapseln, dass sie von der Fachabteilung übernommen werden können (z. B. Datenmodellierung mit JSON Schema und SKOS oder Datentransformationen mit Metafacture oder Catmandu anstatt mit Python- oder Perl-Skripten), kann auf Widerstände stoßen. Hier kann etwa mit verbesserter Nachhaltigkeit und Entlastung der Programmierer:innen durch Integration der Fachebene argumentiert werden.
- Balance halten und Nein sagen: In der Bibliothekswelt und auch darüber hinaus gibt es heutzutage einen Fokus auf „innovative“ Projekte und *the next big thing*. Dabei treten der Betrieb nachhaltiger, zukunftsfähiger Infrastrukturen und vor allem auch die damit verbundenen Aufwände in den Hintergrund. Deshalb ist es – nicht zuletzt zur Schonung der Mitarbeiter:innen – wichtig, die Zustimmung zu neuen Projekten von einer realistischen Ressourcenplanung abhängig zu machen.
- Miteinander auf Augenhöhe: Es kann schwierig sein, im Entwicklungsteam ein gemeinsames Miteinander und einen Austausch auf Augenhöhe herzustellen und zu halten. Von Seiten der Bibliothekar:innen ist hier Offenheit und Neugierde nötig. Außerdem sollte nachgefragt und auch kritisch hinterfragt werden, wenn Aussagen oder Entscheidungen von Programmierer:innen nicht klar verstanden werden. So gewinnt ein Team zunehmend ein gemeinsames Verständnis der Anforderungen und gemeinsamen Aufgaben und schafft damit die Voraussetzungen zur gemeinschaftlichen Entwicklung passender Lösungen.

Wichtig ist immer, dass man aus diesen und anderen Herausforderungen und Problemen lernt. Insgesamt kann ich versichern, dass die nötigen Kompetenzen und Erfahrungen kein Hexenwerk sind und dass man diese von Projekt zu Projekt nach und nach erwirbt. Gezieltes Ausprobieren und das regelmäßige Begehen und Reflektieren von Fehlern sind dabei notwendige Schritte im Lernprozess.