Diskussionsbeitrag: Handreichung zur Rezension von Forschungssoftware in der Archäologie und den Altertumswissenschaften

Timo Homburg, Anne Klammt, Hubert Mara, Clemens Schmid, Sophie Charlotte Schmidt, Florian Thiery & Martina Trognitz

Zusammenfassung – Motiviert durch zahlreiche Diskussionen rund um den zunehmenden Einsatz von Forschungssoftware im Bereich der Archäologie werden in diesem Beitrag Aspekte zu deren Rezension skizziert. Die Bewertung von Software ist ein komplexes Thema, da deren Einsatzgebiet und ihr Entwicklungskontext einen erheblichen Einfluss auf Forschungsergebnisse haben. Hinzu kommen unterschiedlichste Anwendungsfälle von z. B. Studierenden, die rasch eine Übungsaufgabe lösen, bis hin zu Projektentwicklern, die ein Softwarepaket für den Dauerbetrieb in eine bestehende Infrastruktur integrieren müssen. Obwohl an der Erstellung dieser ersten Version eines Leitfadens paritätisch Beiträge aus der Archäologie und der angewandten Informatik eingeflossen sind, liegt der Fokus auf Beurteilungskriterien von Software im Anwendungsbereich der Archäologien. Ein Ziel dieses Impulses für Softwarerezensionen ist es, künftige Rezensentinnen und Rezensenten für die Komplexität der Beurteilung von Software zu sensibilisieren. Eine Softwarerezension soll dem archäologischen Fachpublikum eine rasche, kritische und – auch den Entwicklerinnen und Entwicklern gegenüber – faire Beurteilung von Software ermöglichen. Zu den vorrangigen Empfehlungen gehört die Beschreibung des Kontextes, in dem die Rezension verfasst wurde bzw. die gestellten Anforderungen für bestimmte Anwendungsfälle. Zusätzlich soll eine kurze tabellarische Übersicht eine rasche Einschätzung der technischen, finanziellen und rechtlichen Aspekte ermöglichen. Der Bedarf nach künftigen Überarbeitungen dieses Impulses für Softwarerezensionen wurde bereits bei dessen Erarbeitung festgestellt, da sowohl bei der Softwareentwicklung wie auch deren Bewertung im digitalen Zeitalter weiterhin eine große Dynamik zu erwarten ist.

Schlagwörter – Archäologie; Archäoinformatik; Handreichung; Softwarebewertung; Forschungssoftware; Rezension

Title – Recommendations for the review of archaeological research software

Abstract – Motivated by numerous discussions around the increasing use of research software in the field of archaeology, this article outlines some aspects for its review. The evaluation of software is a complex topic, since its field of application and development context has a considerable influence. In addition, there are many very different use cases, ranging from students wanting a quick solution for an exercise to project developers, who have to integrate a software package into an existing infrastructure for continuous operation. Although this discussion paper is based on equal contributions from archaeology and applied computer science, the focus is on evaluation criteria for software in the field of archaeology. A major goal of this paper is to alert future reviewers to the complexity of software evaluation. A software review should enable a professional archaeological audience to make a quick, critical and fair, to both the product and the developers assessment of the software. Priority recommendations include a description of the context in which the review was written and the requirements for specific use cases. In addition, a short tabular overview should enable a quick assessment of the technical, financial and legal aspects. The need for future adaptations of this guideline was identified at the outset since both software development and its evaluation in the digital age are expected to remain very dynamic.

Key words – archaeology; manual; guideline; software review; research software; review

Zielsetzung des Beitrags

Ziel dieses Beitrags ist es, die Diskussion um Inhalte und Zielsetzungen von Forschungssoftwarerezensionen und im Forschungsprozess eingesetzter digitaler Werkzeuge in den Archäologien und den Altertumswissenschaften anzuregen. Anstoß für unsere Überlegungen gab eine Diskussion von Kai-Christian Bruhn, Sophie Charlotte Schmidt und Frank Siegmund im Plenum des 9. Workshops der deutschen Sektion der CAA 2019 in Wilhelmshaven. Anlässlich der Einrichtung der neuen Rubrik *„Archäoinformatik"* für solche Besprechungen in der Zeitschrift *Archäologische Informationen* haben wir unsere Gedanken nun verschriftlicht. Die wenigen zum Zeitpunkt unserer Diskussion vorliegenden einschlägigen Softwarerezensionen (insb. in der Zeitschrift *Internet Archaeology*; https://intarch.ac.uk/ [23.10.2020]) erscheinen dem Autorenteam nicht einheitlich und umfassend genug.

In dieser Handreichung erläutern wir Kriterien und Prinzipien, die gute Forschungssoftware ausmachen und die für deren Bewertung von Bedeutung sein können. Außerdem empfehlen wir eine Vorgehensweise für Softwarerezensionen und stellen einen Fragenkatalog zur kritischen Beurteilung von Software vor.

Die Beurteilung einer Publikation ist eine traditionsreiche Form des wissenschaftlichen Diskurses. Entsprechend gibt es ein unausgesprochenes Verständnis davon, was eine Rezension umfassen sollte. Zusätzlich können konkrete Maßnahmen zur Qualitätssicherung ergriffen werden. Dazu gehören Hinweise zum Anfertigen von Rezensionen oder – wie es die Redaktion der Archäologischen Informationen praktiziert – Reviews von Rezensionen.

Eingereicht: 23. Okt. 2020
angenommen: 10. Nov. 2020
online publiziert: 19. Jan. 2021

Was sollte eine gute Rezension von Software für den Einsatz in der Archäologie ausmachen? Geht es allein um die von einer Software ausgeführten Routinen, welche zur Lösung einer wissenschaftlichen Problemstellung beitragen? Oder müssen Aspekte der Nutzbarkeit, der Nachhaltigkeit und Interoperabilität ebenfalls Beachtung finden? Welche Rolle spielen technische und rechtliche Aspekte in der Besprechung, wie etwa die Dokumentation des Quellcodes oder die Lizensierung? Und schließlich: Stellt die Software vielleicht selbst einen wissenschaftlichen Beitrag dar? Welche Leistungen der Softwareentwickler sollten in einer Besprechung berücksichtigt werden? Welche Maßstäbe, die in den Empfehlungen der Deutschen Forschungsgemeinschaft (DFG) anklingen, sollte eine gute Forschungssoftware erfüllen? Und nicht zuletzt: Wer soll oder kann eine Software angemessen rezensieren?

Die zentrale Fragestellung einer Rezension nach dem wissenschaftlichen Wert des Besprechungsgegenstands wird mit diesen Fragen nur gestreift. Denn in diesem Punkt stimmen die Anforderungen an eine Softwarerezension mit denen einer Textrezension überein. Weiterhin werden aufgrund der Vielfalt und Vielartigkeit von Software nicht alle hier gesammelten Kriterien in jeder Rezension von Bedeutung sein. Es obliegt dem Rezensenten, relevante Aspekte auszuwählen und Schwerpunkte zu setzen.

Dieser Beitrag ist als Impuls gedacht. Diskussionsbeiträge und Anregungen sind ausdrücklich erwünscht. Dazu können die Autoren über die angegebenen Adressen kontaktiert oder Anmerkungen direkt auf GitHub (https://github.com/Research-Squirrel-Engineers/Impuls_SoftwareRezensionen_DGUF/ [23.10.2020]) hinterlegt werden.

Forschungssoftware

Die weiter unten vorgestellten Kriterien zur Rezension von Software zielen primär auf die Begutachtung von Forschungssoftware ab. Unter Forschungssoftware verstehen wir Software, die mit dem Fokus auf einen Einsatz in der Forschung entwickelt wurde, also um Forschungsdaten zu erzeugen, zu verarbeiten oder zu analysieren (Hettrick et al., 2014). Dies sind beispielsweise Programme, die der Kalibrierung und Umrechnung von Messwerten, der Visualisierung von räumlichen Daten, der Annotation von Texten und Objekten oder der Bereitstellung und Verknüpfung fachlich relevanter Vokabulare dienen. Forschungssoftware ist stets Teil eines Forschungsprozesses, der in allen seinen Punkten nachvollziehbar und – soweit möglich – reproduzierbar sein muss.

Im Gegensatz zu Forschungssoftware steht Software, die zur Nutzung spezifischer Geräte erforderlich ist, wie etwa zur Vermessung, zur fotografischen Dokumentation oder zur Analyse von Oberflächen und Inhaltsstoffen. Häufig handelt es sich dabei um proprietäre Software, die zusammen mit der Hardware vertrieben wird. Obwohl sie im Forschungsprozess eingesetzt wird, entspricht sie nicht unserem Verständnis von Forschungssoftware. Auch digitale Werkzeuge (Tools), die eine erhebliche Rolle in der praktischen Arbeit, jedoch keinen eigentlichen Anteil an der Erhebung, Bearbeitung und Analyse der Daten innehaben, sind in unserem Sinne keine Forschungssoftware. Dazu gehören z. B. Textverarbeitungs- oder Tabellenkalkulationsprogramme. Die vorgestellten Kriterien können indes zur Beurteilung jedes Software-Typs verwendet werden.

Forschungssoftware ist in besonderer Weise Herausforderungen nachhaltiger Entwicklung und Pflege ausgesetzt. Als einen Hauptfaktor für fehlende Nachhaltigkeit von Forschungssoftware identifizieren wir den Mangel an langfristiger Finanzierung für Personal und Infrastruktur zur Erzeugung von nachhaltiger Software (Anzt et al., 2020, 2). Ein Überblick über Bedarfe, Herausforderungen und Lösungsansätze für nachhaltige Forschungssoftwareentwicklung wurde 2019 in einem Workshop der deRSE-Konferenz in Potsdam (https://de-rse.org/de/conf2019/index.html [23.10.2020]) erstellt (Bach et al., 2019).

Forschungssoftware als wissenschaftliche Leistung

In der modernen Forschung ist Arbeiten ohne digitale Werkzeuge undenkbar. Dies trifft auch auf die historische und altertumswissenschaftliche Forschung zu. Mit dem Aufschwung der Digital Humanities wird Software zunehmend ein wichtiger Bestandteil des Forschungsprozesses und greift implizit und explizit tief in ihn ein (für die Archäologie z. B. Schmidt & Marwick, 2020). Nur durch die Offenlegung von Quellcode ist eine Bewertbarkeit der durch die Software geleisteten Abläufe wirklich gewährleistet.

Trotz der tragenden Rolle, die Forschungssoftware in vielen Projekten einnimmt, wird die Leistung jener Personen, die hinter der Entwicklung und Programmierung stehen, akademisch oft nicht ausreichend anerkannt (Hettrick, 2016; Katerbow & Feulner, 2018; Scheliga et al., 2017). Das wird der

Tatsache nicht gerecht, dass für die Entwicklung von Forschungssoftware wissenschaftliche Expertise und darüber hinaus fortgeschrittene technische Kompetenz erforderlich sind. Durch die Umsetzung in Code wird Praxis und Wissen explizit manifestiert und weiterentwickelt. Wir argumentieren, dass diese Leistungen gewürdigt und sichtbar gemacht werden müssen, zumal der wissenschaftliche Durchbruch häufig erst durch die Software ermöglicht wird (Helmholtz Open Science, 2019; Katerbow & Feulner, 2018; Scheliga et al., 2017).

Als ein wichtiger erster Schritt zur Sichtbarmachung wurde 2012 die Berufsbezeichnung des *„Research Software Engineers"* (RSE, RSEng) geprägt (Baxter et al., 2012; Hettrick, 2016). Inzwischen hat sich eine aktive, interdisziplinäre Gemeinschaft gebildet, die Empfehlungen zum Umgang mit Forschungssoftware entwickelt (Anzt et al., 2020). So organisieren nationale RSE-Sektionen (https://sorse.github.io/contact/chapters/ [23.10.2020]) – in Deutschland *„de-RSE e.V."* – disziplinübergreifende RSE-Konferenzen wie z. B. SORSE (https://sorse.github.io [23.10.2020]). Im Rahmen des Aufbaus einer Nationalen Forschungsdateninfrastruktur (NFDI) nehmen Forschungssoftware und RSEs einen großen Stellenwert in der Weiterentwicklung der Forschungslandschaft und der Angebote der Forschungseinrichtungen in Deutschland ein (Goedicke & Lucke, 2020; Löffler, 2020).

Eine der Forderungen des de-RSE e.V. ist, dass Forschungssoftware explizit mittels geeigneter Publikationsmodalitäten sichtbar wird. Einerseits erhöht eine Publikation die Auffindbarkeit und vermeidet so redundante Neuentwicklungen (Anzt et al., 2020, 10), andererseits ist die mit einer Publikation einhergehende eindeutige Autorenschaft von großer Bedeutung, da erst dadurch die Leistung der Softwareentwicklung an die entsprechenden Personen geknüpft wird. Dies ermöglicht eine akademische Laufbahn, wofür bislang jedoch oft nur klassische Textpublikationen gewertet wurden. Daher sollen zukünftig Datenpublikationen, Softwareentwicklung (Druskat et al., 2017), Annotationen sowie deren Zitationen (https://citation-file-format.github.io [17.12.2020]) Kriterien der Beurteilung der wissenschaftlichen Leistung sein (NFDI4Culture, https://nfdi4culture.de [23.10.2020], RSE4NFDI, (https://www.rse4nfdi.de/de/index.html [23.10.2020], NFDI4Objects, https://www.nfdi4objects.net [23.10.2020]). Hierbei sind technische Herausforderungen zu meistern, da Software nie endgültig abgeschlossen ist, oft auf bestehenden Modulen aufbaut und mitunter von wechselnden Teams über Jahrzehnte betreut wird (Katz et al., 2016).

Eine offen verfügbare Software kann, ganz so wie in der bisherigen Praxis der altertumswissenschaftlichen Forschung, begutachtet werden. In einer Rezension kann diese dann einem breiteren Publikum vorgestellt werden. Durch eine dedizierte Rezension von Software analog zu Rezensionen wissenschaftlicher Publikationen wird die wissenschaftliche Leistung der Autoren und Autorinnen von Forschungssoftware mittels gewohnter Formate sichtbar und anerkannt. Gleichzeitig werden auch ihre Verantwortung und Teilhabe manifest.

Gute Forschungssoftware

Um Kriterien zur Rezension und Bewertung von Forschungssoftware aufstellen zu können, stellt sich zunächst die Frage, wie man deren Qualität bestimmt und nach welchen Standards diese beurteilt werden soll. Dazu gehört eine Vielzahl fachlicher und technischer Aspekte, die weiter unten beleuchtet werden. Darüber hinaus muss Software aber, wie jedes andere Werkzeug, zunächst hinsichtlich ihrer generellen Eignung für nachhaltiges und ethisches wissenschaftliches Arbeiten bewertet werden.

Die DFG formuliert mit ihren *„Leitlinien zur Sicherung guter wissenschaftlicher Praxis"* (DFG, 2019) eine Reihe von Vorgaben für gutes wissenschaftliches Arbeiten. Die Leitlinien entsprechen einem internationalen Konsens und etablierten Prinzipien. Zu den Anforderungen gehören etwa die Einhaltung und Etablierung von Standards und Methoden, eine nachvollziehbare Dokumentation des Weges zu Ergebnissen, die öffentliche Zugänglichkeit von Ergebnissen und die Archivierung der notwendigen Materialien, die zu Ergebnissen geführt haben.

Einige der Kriterien, die in den DFG-Leitlinien explizit und implizit anklingen, werden im Folgenden näher ausgeführt: Die in der Erläuterung zu Leitlinie 13 genannten FAIR-Prinzipien, die Prinzipien der Offenen Wissenschaft (Open Science) und die in den Leitlinien 2 und 10 angeführten CARE-Prinzipien. Weitere Hinweise zur Umsetzung der DFG-Leitlinien in Bezug auf Forschungssoftware sind unter forschungsdaten.info (2020) zu finden.

FAIR-Prinzipien

Die FAIR-Prinzipien wurden 2016 als *FAIR Data Principles* veröffentlicht (Wilkinson et al., 2016). Sie zielen primär auf Forschungsdaten und deren Metadaten ab und fordern, dass diese auffindbar

(Findable), zugänglich (Accessible), interoperabel (Interoperable) und nachnutzbar (Reusable) sind. Ein Fokus ist dabei die maschinelle Lesbarkeit. Die Prinzipien lassen sich auch auf Forschungssoftware und deren Metadaten übertragen (Goedicke & Lucke, 2020; Lamprecht et al., 2019) und bieten somit Kriterien zur Beurteilung der Qualität von Softwarepublikation, Codedokumentation und Programmierstandards.

Software wird durch die Speicherung zusammen mit ihren Metadaten in dedizierten Softwarerepositorien (z. B. GitHub, GitLab) auffindbar (Findable). Eine Einbindung in fachwissenschaftliche Forschungsinfrastrukturen erhöht die Auffindbarkeit, da diese explizit an Aufbau, Pflege und Anwendung von kontrollierten Indizes und Metadatenstandards arbeiten (Anzt et al., 2020; Lamprecht et al., 2019).

Mit durch Repositorien vergebenen persistenten Identifikatoren (z. B. URIs) und durch die Nennung der Autorenschaft wird das Zitieren von Forschungssoftware ermöglicht. Da Software im Gegensatz zu herkömmlichen Publikationen ständig weiterentwickelt wird, sollte sie in unterschiedlichen Versionen gesichert werden, welche individuell zitiert werden können (forschungsdaten.info, 2020).

Öffentliche Repositorien und dauerhafte Identifikatoren stellen ebenfalls ein Mindestmaß an Zugänglichkeit (Accessible) sicher. Idealerweise sollten Daten und Metadaten dabei auch maschinell über ein offenes und standardisiertes Kommunikationsprotokoll abrufbar sein (Lamprecht et al., 2019). Weitere die Zugänglichkeit erleichternde Aspekte sind die Verfügbarkeit für verschiedene Betriebssysteme und technische Anforderungen, die von aktuell verbreiteten Geräten geleistet werden können. Auch eine verständliche Bedienungsanleitung verbessert die Zugänglichkeit.

Interoperabilität (Interoperable) von Software kann sich zum einen auf die Kompatibilität der Ein- und Ausgabeformate mit anderen Programmen in einem Arbeitsprozess (horizontale Dimension) beziehen, zum anderen verweist sie auch auf die Zusammenarbeit der verwendeten Komponenten in der Software selbst (vertikale Dimension) (Lamprecht et al., 2019, 46 f.). Die Verwendung von Standards ermöglicht beides, da sie die Zusammenarbeit von Softwarekomponenten auch über Betriebssystemgrenzen hinweg ermöglicht.

Die Nachnutzbarkeit (Reusable) von Software hängt von mehreren Komponenten ab. Metadaten und eine umfassende Dokumentation der Software sollten es anderen ermöglichen, Ergebnisse zu reproduzieren sowie eigene Daten und veränderte Anwendungsfälle zu prozessieren. Eine geeignete Lizenz, die auch die abhängigen Softwarekomponenten berücksichtigt, gibt zudem darüber Auskunft, welche Regeln für die Nutzung und die Weiterentwicklung des Codes gelten (Lamprecht et al., 2019, 48-49).

Open Science

Open Science (Offene Wissenschaft) bezeichnet eine Wissenschaftspraxis, deren Ziel ein transparenter, reproduzierbarer und kollaborativer Forschungsprozess ist (Bezjak et al., 2018, ‚Open Concepts and Principles'). Open Science setzt sich aus mehreren Prinzipien zusammen, die unterschiedliche Stadien im Forschungsprozess betreffen. So fordert Open Science nicht nur die Öffnung der Ergebnisse (Open Access), sondern auch die der zugrunde liegenden Daten (Open Data), Methoden (Open Methodology) und der verwendeten Forschungssoftware (Open Source). Für offene Forschungssoftware gilt, dass ihr Quellcode zugänglich und mit einer Lizenz versehen sein muss, welche die Weiterentwicklung erlaubt (Bezjak et al., 2018, ‚Open Research Software and Open Source').

Mit der Forderung nach der Offenlegung des Quellcodes (Open Source) wird auch dem Gedanken der Nachnutzbarkeit entsprochen. Die Offenlegung wird erweitert um die Forderung, den Code frei nutzbar zur Verfügung zu stellen (Free Source) (Stallman, 2001). Letzteres wird als FOSS (Free/Libre Open Source Software) bezeichnet und in Deutschland z. B. von anwendungsbezogenen Wissenschaftlern und Entwicklern aus der Geoinformatik (FOSSGIS e.V., https://www.fossgis.de [23.10.2020]) oder auch von Research-Software-Engineers in der *„de-RSE e.V."* (https://de-rse.org/de/association.html [23.10.2020]; Anzt et al., 2020) vertreten.

In den Leitlinien der DFG klingt die Öffnung von Software-Quellcode im Zusammenhang mit der Qualitätssicherung (DFG, 2019, 14-15) und der Herstellung von öffentlichem Zugang zu Forschungsergebnissen (DFG, 2019, 19) an.

CARE-Prinzipien und Ethos

In den Leitlinien der DFG werden in Leitlinie 2 und 10 ethische Aspekte im Forschungsprozess angesprochen, die unserer Meinung nach auch im Bereich der Forschungssoftware Beachtung finden sollten.

Privilegien und Ungleichheiten in der altertumswissenschaftlichen Forschung werden unbewusst auch in der Forschungssoftware fortgeschrieben.

Teure, proprietäre Software benachteiligt Forscher, die mit weniger Finanzmitteln auskommen müssen, da ihnen so der Zugang zu zeitgemäßen Analysemethoden verwehrt bleibt. In weiterer Folge könnte der Zugang zu hoch bewerteten Zeitschriften erschwert werden und die studentische Ausbildung in den entsprechenden Methoden stagnieren, was bestehende Ungleichgewichte weiter ausprägt. Auch Sprachbarrieren tragen zu einem Gefälle bei: Wenn in internationalen Teams z. B. eine Datenbank verwendet wird, die nicht auch in der Landessprache verfügbar ist, werden indigene Forscher unter Umständen an einer selbstständigen Analyse und Auswertung der Daten sowie an einer Weiterqualifizierung gehindert.

In den Archäologien, in denen häufig an fremden Kulturen und Kulturhinterlassenschaften geforscht wird, sollten die indigenen und ortsansässigen Bevölkerungsgruppen die Möglichkeit zur Mitbestimmung über die Erforschung bekommen. Um diesem Ideal gerecht zu werden, eignet sich die Orientierung an den CARE-Prinzipien (https://www.gida-global.org/care [23.10.2020]). Sie wurden von der Global Indigenous Data Alliance (GIDA) (https://www.gida-global.org [23.10.2020]) und der Research Data Alliance (RDA) (https://www.rd-alliance.org [23.10.2020]) im Jahr 2018 auf Basis der UN Declaration on the Rights of Indigenous Peoples (UNDRIP) (https://www.un.org/development/desa/indigenouspeoples/declaration-on-the-rights-of-indigenous-peoples.html [23.10.2020]) erarbeitet und konzentrieren sich – wie die FAIR-Prinzipien – explizit auf Forschungsdaten. Ihre vier Grundpfeiler kollektiver Nutzen (Collective Benefit), Souveränität (Authority to Control), Verantwortung (Responsibility) und Ethik (Ethics) sind unserer Meinung nach ebenso für Forschungssoftware relevant.

Mögliche Vorgehensweise zur Rezension archäologischer Forschungssoftware

Die Rezension archäologischer Forschungssoftware setzt die Sichtung von Dokumentation und Publikationen ebenso voraus wie die praktische Erprobung der Software selbst. Unserem Verständnis nach sollte eine Softwarerezension, so wie es eine herkömmliche Besprechung einer wissenschaftlichen Publikation leistet, die Software zunächst mit zusammenfassenden Eckdaten vorstellen. Anschließend sollte der Kontext erläutert werden und eine kritische Beurteilung erfolgen.

Zu den Eckdaten, welche tabellarisch dargestellt werden können, gehören unter anderem Angaben zur Version, zu den Entwicklern und der Lizenz. Dies ermöglicht z. B. eine schnelle Übersicht darüber, ob die begutachtete Software mit der eigenen technischen Umgebung kompatibel ist. Ein Vorschlag zu den Inhalten einer solchen Tabelle befindet sich am Ende dieses Beitrags.

Mit Kontext ist die Einordnung in ein archäologisches Forschungsfeld und Angaben zum möglichen Zusammenhang mit Forschungsprojekten, Arbeitsgruppen oder Institutionen gemeint. Auch der ‚Kontext' der rezensierenden Person sollte dargestellt werden, denn eine realistische und transparente Einschätzung der eigenen Kompetenzen und des eigenen Nutzungsinteresses stellt einen wichtigen Anhaltspunkt für die heterogene Leserschaft dar. Wurde die Rezension rein aus Sicht eines Anwenders oder auch aus Entwicklersicht geschrieben? Angaben zur verwendeten Testumgebung, also des zum Testen der Software genutzten Computers, können ebenfalls von Bedeutung sein, wie z. B. Angaben zu Betriebssystem, RAM, Prozessor, Grafikkarte oder Bandbreite.

Für die kritische Beurteilung von Forschungssoftware aus verschiedenen Blickwinkeln folgt hier im Anschluss ein in drei Bereiche gegliederter Fragenkatalog. Die Fragen aus dem Bereich *„Einsatz in der Archäologie und wissenschaftlicher Zweck"* sollten unserer Ansicht nach in jeder Rezension behandelt werden. Als sehr wichtig erachten wir die Beantwortung der Fragen zu Installation, Tutorials und der tragenden Community, sowie zu Ein- und Ausgabeformaten, Programmierschnittstellen und den Möglichkeiten zur Beteiligung an der Weiterentwicklung der Software. Allein diese Fragen ziehen weitere nicht unwichtige Folgefragen nach sich. Beispielsweise sind für die Installation auch Informationen dazu, ob es sich um eine eigenständige (stand-alone) Software oder um eine Webanwendung handelt, ob die Installationsvoraussetzungen klar dokumentiert sind und ob die Dokumentation vollständig und aktuell ist, wichtig. Die unserer Ansicht nach besonders relevanten Fragen sind im Katalog mit vorangestellten ✚ (sehr wichtig), + (wichtig) gekennzeichnet.

Die Ergebnisse der Begutachtung sollten in einer Stellungnahme zusammengefasst werden, welche die Software hinsichtlich ihrer Nützlichkeit, ihrer Bedienbarkeit, ihrer handwerklichen Qualität und ihrer Position in Relation zu den bereits angesprochenen Idealen guter Forschungssoftware beurteilt.

Der im Anschluss vorgestellte Fragenkatalog ist umfangreich und muss nicht in allen Einzelheiten abgearbeitet werden. Seine Komplexität und Tie-

fe ist ein guter Indikator dafür, dass die Entwicklung von Forschungssoftware eine vollwertige, wissenschaftliche Tätigkeit sein kann. Wir raten dazu, die eigenen Kompetenzen für die Rezension kritisch zu reflektieren und die Evaluierung gegebenenfalls in einem Team durchzuführen. Denn so wie Rezensionen von herkömmlichen Publikationen im Idealfall von mit dem Thema betrauten Experten erstellt werden, sollte dies auch für Forschungssoftwarerezensionen gelten.

Fragenkatalog zur Beurteilung von Software

Im Folgenden stellen wir einen kommentierten Fragenkatalog mit Kriterien zur Beurteilung von archäologischer Forschungssoftware vor. Drei Bereiche bündeln jeweils Fragen aus verschiedenen Kompetenzbereichen. Die ersten beiden Bereiche beschäftigen sich mit dem wissenschaftlichen Anwendungsfeld sowie der Anwendung und Bedienbarkeit aus Nutzersicht. Der dritte Bereich konzentriert sich auf Fragen, die insbesondere für Entwickler und IT-Administratoren relevant sind. Abgeschlossen wird der Katalog durch eine Liste von Merkmalen, welche in tabellarischer Form die Rezension ergänzen, jedoch meist nicht kritisch beurteilt werden können.

Wie bei der Besprechung einer wissenschaftlichen Publikation ist die Zusammensetzung und Gewichtung der einzelnen Merkmale bei der Begutachtung selbst zu bestimmen und in Relation zum Anwendungskontext zu setzen. Entsprechend verstehen wir unseren kommentierten Fragenkatalog als Maximalversion, die als Hilfsmittel für die Begutachtung und die Einschätzung der eigenen Kompetenzen dienen kann.

Einsatz in der Archäologie und wissenschaftlicher Zweck

Bei der Beurteilung der wissenschaftlichen Qualität von Forschungssoftware sind zunächst zwei wesentliche Dimensionen zu unterscheiden. Dies ist zum einen – das ist für Forschungssoftware vorrangig – die Frage, ob die Software einen sinnvollen Beitrag zur Bearbeitung der archäologischen Fragestellung liefert. Zum anderen ist zu beurteilen, ob Software eine korrekte Umsetzung der angestrebten Arbeit leistet. Beide Fragen sind mitunter nicht leicht zu beantworten.

- **Welche Aufgabenstellung versucht die Software zu lösen?** Diese Frage ist mit dem deskriptiven Charakter einer Rezension verknüpft: Welche Aufgaben werden mittels der Software bei der Erfassung, der Verarbeitung oder der Analyse von Daten bearbeitet? Wie relevant sind die Aufgaben in einem archäologischen Kontext und wie häufig werden sie gestellt? Dieser Punkt verdeutlicht, warum es von besonderem Wert ist, wenn Archäologen selbst Softwarerezensionen für sich und ihre Kollegen verfassen.
- **Wie löst die Software eine gegebene (technische) Aufgabe?** Eine detaillierte Beantwortung dieser Frage ist nur aus Entwicklerperspektive möglich. Kernbestandteile lassen sich jedoch meist leicht identifizieren. Wie ist die grundsätzliche Funktionsweise der Software konzipiert? Was sind die wesentlichen technischen Bestandteile im Nutzerinterface und in den dahinter liegenden Datenverarbeitungsmodulen? Handelt es sich zum Beispiel um eine Webanwendung, die als Schnittstelle zu einer Datenbank fungiert? Oder ist die Software ein schlichtes, monolithisches Kommandozeilenprogramm?
- **Wie funktioniert der wissenschaftliche Arbeitsablauf, der in der Software implementiert wurde?** Hier geht es weniger um die konkrete technische Implementierung, sondern vielmehr um die generelle Methodik. Welche wesentlichen Schritte durchlaufen Daten, um eine bestimmte Aufgabe zu lösen? Welche statistischen Werkzeuge kommen zum Einsatz? Gibt es Vergleichsdaten? Ein Beispiel ist etwa die Bereinigung von Eingabedaten, deren anschließende Klassifizierung durch einen Algorithmus und zuletzt ihre Visualisierung.
- **Ist die Behauptung, eine bestimmte wissenschaftliche Fragestellung mit dem gewählten Arbeitsablauf beantworten zu können, korrekt?** Forschungssoftware verspricht meist – implizit oder explizit – die Beantwortung wissenschaftlicher Fragestellungen zu ermöglichen oder zumindest zu vereinfachen. Eine GIS-Applikation zur Analyse von Punktmustern kann etwa für den Zweck konzipiert sein, menschliches Siedlungsverhalten aufzuzeigen. Die Frage, ob der angewendete Algorithmus dazu überhaupt in der Lage ist, kann den Umfang einer Rezension bei Weitem übersteigen. Dennoch sollte der Rezensent versuchen, eine Einschätzung der Plausibilität vorzunehmen oder zumindest das Problem der Einschätzbarkeit selbst offenzulegen, um die Leser zu sensibilisieren. Zur Beurteilung kann es helfen sich zu vergegenwärtigen, welche Schlüsse aus den rohen, unverarbeiteten Eingabedaten überhaupt theoretisch ableitbar sein könnten.

- ✚ **Sind die Algorithmen korrekt implementiert worden?** Die Korrektheit der Ergebnisse einer Software ist mitunter schwer zu beurteilen, da beispielsweise ein umfangreicher Blackbox-Test durchgeführt werden muss. Ein Mangel an vergleichbarer Software mit der zu testenden Funktionalität erschwert dies zusätzlich. Werden die in der Dokumentation genannten Algorithmen fehlerfrei in Programmcode ausgedrückt? Ist das wissenschaftliche Ergebnis trotz aller technischen Unterschiede vergleichbar mit anderen Softwarelösungen? Sind die verwendeten Algorithmen dokumentiert und hinreichend wissenschaftlich belegt? Auch diese Frage mag im Rahmen einer Rezension nicht abschließend zu beantworten sein. Ein wichtiges Indiz für eine fehlerhafte Implementierung ist eine mangelnde Robustheit – ein Aspekt, der später noch besprochen wird.
- ✚ **Gibt es für die Archäologie relevante Projekte/Anwendungen, in denen die rezensierte Software bereits angewendet wurde?** Software wird oft von und für bestimmte Forschungsprojekte entwickelt. Zur Beurteilung ihrer Qualität und Relevanz kann es also hilfreich sein, Forschungsprojekte selbst genauer anzusehen. Wie plausibel sind die erzielten Resultate, die mit der Software zusammenhängen? Wie wurden sie von der wissenschaftlichen Community aufgenommen?
- ✚ **In welcher Form ist die Software publiziert?** Idealerweise sollte Forschungssoftware auch wissenschaftlich publiziert sein. Dies erleichtert die Zitierbarkeit. Gibt es ein Benchmarking-Paper, in dem das Werkzeug explizit vorgestellt und mit alternativen Produkten verglichen wird? Sind einzelne Versionen der Software auch als solche referenzierbar? Ist zum Beispiel ein Digital Object Identifier (DOI) und somit ein persistenter Link vorhanden? Wurde die Software in einer Fachzeitschrift oder einem anderen Medium veröffentlicht? Gab es dazu einen expliziten Software-Peer-Review-Mechanismus?

Bedienbarkeit und Zielgruppenorientierung

Die Bedienbarkeit von Forschungssoftware ist von zentraler Bedeutung, da sie als Flaschenhals die Interaktion zwischen Nutzer und Software bestimmt. Dazu gehören z. B. die Komplexität des Installationsprozesses, das Nutzerinterface und maschinelle Schnittstellen. Hierbei ist z. B. zu beachten, dass eine grafische Nutzeroberfläche Vorteile für Gelegenheitsnutzer bietet, während eine Bedienung per Kommandozeile für die Prozessierung von größeren Datensammlungen für erfahrene Nutzer vorteilhafter ist. Hilfestellungen (Foren, FAQs, Tutorials) und die Größe und Aktivität der Nutzer- und Entwicklergemeinschaft sind entscheidend für die praktische Bedienbarkeit. Die Größe und Aktivität der Gemeinschaften kann ein wichtiges Indiz für die Zukunftsfähigkeit einer Software und damit ihre Eignung für den Einsatz auf institutioneller Ebene in langfristigen Vorhaben sein.

Die folgenden Fragen nehmen Bezug auf technische Merkmale, welche die Nutzererfahrung (User Experience, UX) beeinflussen. Weitere technische Merkmale aus der Entwicklerperspektive betrachten wir weiter unten.

INSTALLATION

Die Installation ist häufig der erste Schritt in der Bedienung der Software. Schon an dieser Stelle können Punkte auffallen und/oder für die Leser einer Rezension von Bedeutung sein.

- ✚ **Wie funktioniert die Installation und wo wird die Software vorgehalten?** Je einfacher die Installation einer Software ist, desto größer ist ihr potenzieller Nutzerkreis. Die Vielseitigkeit von Computersystemen macht es aus Entwicklersicht oft schwer, eine einfache Installation für alle Nutzer zu gewährleisten. Für die Rezension kann geprüft werden, ob ein Installationsskript, -Wizard oder -Paket bereitsteht. Gegebenenfalls ist die Software nur als Quellcode verfügbar, der zunächst selbst kompiliert werden muss. Installationsskripte oder -pakete ermöglichen einen viel breiteren Anwenderkreis, wohingegen das Kompilieren es geübten Nutzern erlaubt, die Installation auf verschiedenen Endgeräten durchzuführen. Wenn die Software für die Rezension selbst installiert wird, können konkrete Probleme und Schwachstellen beim Installationsprozess erkannt und in der Rezension dokumentiert werden.
- ✚ **Handelt es sich um eine eigenständige (standalone) Software oder um eine Webanwendung?** Nicht jede Software muss lokal installiert werden, um genutzt werden zu können. Webanwendungen, die entweder dynamisch im Browser oder auf einem Server ausgeführt werden, sind heute zu beliebig komplexen Operationen in der Lage. Daher stellt sich die Frage, ob die Anwendung eher als lokale Installation oder als Webanwendung zum Einsatz in der archäologischen Praxis taugt. Webanwendungen können beispielsweise nicht hinreichend performant sein, da sie den Transfer großer Datenmengen

über das Internet erfordern. Sie sind im Gelände gegebenenfalls nicht ausführbar und daher nicht für alle Arbeitsbedingungen sinnvoll.

+ **Sind grundlegende Voraussetzungen in Form von Hardware und Betriebssystem klar dokumentiert?** Gerade in der Projektplanung ist es wichtig, technische und finanzielle Anforderungen korrekt identifizieren zu können. Eine Spezifikation bzw. Hinweise zur benötigten Hardware und Betriebssystem sind z. B. für die Einbindung von vorhandenen Infrastrukturen bei größeren Vorhaben oder Instituten von Relevanz.

INTERFACE

Die Nutzbarkeit von Software wird von den Möglichkeiten der Kommunikation von Mensch und Programm bestimmt, also der Benutzeroberfläche (User Interface, UI). Diese ist in manchen Fällen grafisch gestützt (GUI), in anderen Fällen erfolgt die Bedienung über eine Befehlseingabe in der Konsole. Eine gemischte Bedienung ist ebenfalls nützlich, um verschiedene Nutzergruppen anzusprechen. Nutzergruppen sind oft vielfältig und können nie scharf umgrenzt werden, wie etwa Grabungstechniker oder Landesarchäologen, doch für die Nutzbarkeitsanalyse können mit ihrer Hilfe geteilte Anforderungen an die Software evaluiert werden. Die Gestaltung von Oberflächen und die Nutzerführung in Menüstrukturen sind ein eigenes Aufgaben- und Forschungsfeld in der Softwareentwicklung. Gut an den Nutzer angepasste Lösungen sind das Ergebnis einer genauen Kenntnis der Zielgruppe und ihrer Gewohnheiten und Bedürfnisse, fügen sich also beispielsweise nahtlos in den archäologischen Forschungsprozess ein. Ein gutes Interface unterstützt das fehlerfreie und effiziente Arbeiten. So sind z. B. die Verständlichkeit der Menü-Einträge oder der Kommandos, aber auch die Aussagekraft von Fehlermeldungen Aspekte, die in der Rezension betrachtet werden sollten. Barrierefreiheit ist bislang ein unzureichend abgedecktes Kriterium, das aber ebenfalls die Effizienz und die mögliche Nutzergruppe adressiert.

+ **Passt das User Interface zur Nutzergruppe?** Jede Nutzergruppe, auch die zu der sich der Rezensent als zugehörig zählt, wird bestimmte Erwartungen an die Bedienbarkeit der Anwendung haben. Für die einen ist eine Kommandozeilenanwendung sehr gut zu bedienen, während andere damit Probleme haben werden. Viele Softwarelösungen haben mehrere Nutzerschnittstellen. Beispielsweise können viele Webanwendungen sowohl über ein Suchfeld und Filterfunktionen in der grafischen Oberfläche einer Webseite als auch code-basiert über eine REST-Schnittstelle angesteuert werden.
+ **Ist eine archäologische Nutzung vorgesehen?** Die Frage nach der Nutzergruppe sollte auch speziell für die Archäologie gestellt werden: Entspricht der archäologische Einsatz den Anwendungsszenarien, die die Entwickler der Software vor Augen hatten? Das hat oft großen Einfluss auf das Nutzerinterface. Beispielsweise ist für von Archäologen genutzte CAD-Software häufig für Architektur- oder Maschinenbauanwendungen konzipiert und konfrontiert Archäologen und Grabungstechniker mit einem überwältigenden Funktionsumfang.
– **Orientiert sich die Menüführung an bestimmten Vorbildern?** Wenn sich die Menüführung oder Tastenkürzel an bekannter, in der Community verbreiteter Software orientiert, wird eine schnellere Einarbeitung ermöglicht. Manche Tools verwenden deswegen z. B. bewusst Eingabemasken, die weitverbreiteten Tabellenkalkulationsprogrammen nachempfunden wurden.
– **Ist das Programm mehrsprachig bzw. in welchen Sprachen wird es angeboten?** In Abhängigkeit von der Nutzergruppe, sollte das Programm eventuell mehrsprachig sein. Üblicherweise wird man zumindest eine englische Fassung erwarten. Bei der Mehrsprachigkeit ist darauf zu achten, dass das Layout in jeder Sprache nutz- und lesbar bleibt. Beispielsweise können sich Beschriftungen von Buttons in der Textlänge stark unterscheiden und in manchen Sprachen zu einer abgeschnittenen Anzeige führen.
– **Sind die Fehlermeldungen für die Rezensenten gut verständlich?** Eine Fehlermeldung an den Benutzer kann diesem ermöglichen, eine Aktion zur Behebung des Fehlers zu ergreifen. Nützliche und hilfreiche Fehlermeldungen sind entsprechend verständlich formuliert und sichtbar in der Anwendung platziert. Darüber hinaus kann eine Fehlermeldung ggf. ein Feedback an die Entwickler der Anwendung liefern. Dies muss die Fehlermeldung dem Benutzer kommunizieren und auch einen brauchbaren Bericht zur Behebung des Fehlers an die Entwickler schicken. Stack Traces (Hinweise auf die Stellen des Programmcodes, an denen ein spezifischer Fehler auftrat) oder aus der Ausführungsumgebung übernommene Fehlermeldungen sind für die meisten Nutzer unverständlich.

Performanz und Robustheit

Performanz und Robustheit beeinflussen, wie mit dem Programm gearbeitet wird. Die Performanz spielt insbesondere bei der Bearbeitung von großen Datensätzen eine Rolle, die Robustheit beeinflusst das Sicherungs- und Speicherverhalten der Nutzer.

– **Ist die Implementierung performant?** Die Performanz einer Anwendung ist je nach Anwendungstyp von mehr oder weniger Wichtigkeit. Ein Rezensent sollte hier bewerten, ob die Software ihre Aufgabe in angemessener Zeit erfüllt. Gegebenenfalls kann die Ausführungszeit weiterer verwandter Softwareimplementierungen mit der zu testenden Implementierung verglichen werden. Die Gründe für eine mangelnde Performanz der Software sind oft nicht leicht zu erkennen. Für Webapplikationen und Plugins ist zu überprüfen, ob sie responsiv und über die verschiedenen Browser hinweg performant sind.

+ **Ist die Software robust?** Die Robustheit einer Software setzt im Wesentlichen voraus, dass regelmäßig Zwischenstände der Aufgaben gesichert werden, um bei einem unvorhergesehenen Abbruch wieder an dem Letztstand anknüpfen zu können. Ein Fehlen dieser Funktion kann oft, z. B. bei einem Stromausfall, den Verlust von Einstellungen, Daten und/oder die gesamte Neuinitialisierung einer Berechnung bedeuten. Ein Beispiel für eine solche Robustheit ist das regelmäßige automatisierte Zwischenspeichern in einem Textverarbeitungsprogramm. Das Textdokument ist somit im Falle eines Absturzes der Software wiederherstellbar. Abhängig vom Anwendungsfall kann es auch vorteilhaft sein, dass die Software eine Historie der Benutzeränderungen vorhält und diese ebenfalls wiederherstellen kann. Dies können die Spracheinstellungen oder auch die Anpassung von Maßeinheiten, Vorgaben zum Speicherort und Ähnliches sein.

Hilfefunktionen, Tutorials und Community

Neben Hilfefunktionen und Tutorials ist es von großer Bedeutung, ob die Software von einer Community getragen wird. Die Zahl der aktiven Nutzer und Entwickler eines Softwarewerkzeugs ist entscheidend dafür, ob man bei Problemen Hilfe in Foren findet oder, bei einem kleinen Nutzerkreis, nur im persönlichen Austausch Hilfe bekommt. Kleine Nutzernetzwerke können jedoch den großen Vorteil bieten, dass konkrete Fragen von den Entwicklern aufgegriffen werden.

✚ **Gibt es ausreichend Tutorials für das Erlernen der Software?** Tutorials sind essenziell, um sowohl Benutzer als auch Softwareentwickler anzusprechen. Nutzer erwarten üblicherweise ein leicht verständliches, auf das Wesentliche konzentrierte Anwendungsbeispiel, um eine Idee für die typische Verwendung zu bekommen. Für Entwickler ist entscheidend, dass ein Tutorial ggf. vorhandene Schnittstellen (APIs) erläutert. Gute Tutorials erläutern das benötigte Vorwissen und weisen auf Quellen zu dessen Aneignung hin. Auch Hilfestellungen in einem FAQ oder einem Troubleshooting-Bereich erhöhen die Qualität eines Tutorials. In welchen Sprachen die Tutorials vorliegen ist ebenfalls von Relevanz.

+ **Gibt es für die Software Testdatensätze?** Diese Frage ist eng mit der Frage nach Tutorials verbunden, da letztere oft mit Übungsdaten arbeiten. Diese Übungsdaten oder Testdatensätze sollten sich an der wissenschaftlichen Praxis orientieren, jedoch ohne spezifische Vorkenntnisse verständlich sein. Sie sollten frei zur Verfügung stehen und möglichst ohne Registrierung nutzbar sein.

– **Sind weitere Informationen zur Software leicht zu finden?** Verweisen die Informationsseiten der Software oder deren Tutorials auf weitere Materialien? Wird auf Publikationen der Entwickler selbst, wie auch auf Rezensionen dazu hingewiesen?

✚ **Wird die Software von einer Community getragen? Ist diese Community möglicherweise ganz oder teilweise altertumswissenschaftlich geprägt?** Beispiele für Softwareentwicklung, die zunächst aus einer altertumswissenschaftlichen Community heraus betrieben wurde und sich inzwischen fachlich erweitert hat, sind Pelagios Commons (https://pelagios.org/ [23.10.2020]) und die Webanwendung Recogito (https://recogito.pelagios.org/ [23.10.2020]). Engagierte Nutzergruppen mit einem Fokus auf archäologischen Fragestellungen haben sich auch z. B. innerhalb der Communities von Softwarepaketen wie QGIS oder R entwickelt.

+ **Gibt es archäologische Best Practices oder Publikationen, die auf die rezensierte Software verweisen?** Während Foren, Blogs und verwandte Angebote einen direkten und oft auch raschen Austausch mit Nutzern und ggf. Entwicklern bieten, ist die Einbindung und Empfehlung von Programmen in Best Practices und Publikationen ein weiteres Indiz für deren Verbreitung, Umfang und Verlässlichkeit.

Dateningest, Interoperabilität und Schnittstellen
Eingabe- und Ausgabedatenformate beeinflussen die Kompatibilität zu anderen Anwendungen und sollten in einer Rezension erwähnt werden. In vielen Fällen können die unterstützten Dateiformate über einen entsprechenden Menü-Eintrag (z. B. *„Speichern unter ...“*) gefunden werden. Auch die verschiedenen Möglichkeiten des Einlesens der Daten und vorhandene Schnittstellen sind relevant.

+ **Welche Datenformate werden wie eingelesen?** Sind alle relevanten Datenformate für die Aufgabe, die die Software im typischen Anwendungsfall lösen soll, einlesbar? Werden offene Datenformate unterstützt? Können Datenformate auch aus gängigen Repositorien eingelesen werden (z. B. Webservices, Git, Cloud Services)?
+ **Welche Datenformate werden ausgegeben?** Das Programm sollte eine Ausgabe von Datenformaten bieten, die eine Weiterverarbeitung in anderen (auch Open Source) Softwarepaketen zulassen. Es sollte also zumindest ein offen spezifiziertes Format angeboten werden. Sollten nur Exporte in ein proprietäres, ggf. von der Software definiertes Format möglich sein, muss dies von den Entwicklern gut begründet sein.
+ **Wie können Daten eingelesen werden? Ermöglicht die Software eine Stapelverarbeitung?** Für sehr viele Anwendungsszenarien ist die Durchführung eines einmal definierten Workflows auf einen Dateistapel oder eine Datenreihe wichtig, wie z.B. die gleichen Transformationsschritte für alle Bilder in einem Ordner.
+ **Gibt es eine Programmierschnittstelle (API)?** Neben der Bedienbarkeit durch einen Menschen ist auch eine maschinelle Ansteuerung der Software wichtig. Nur so können beispielsweise komplexe Prozesse mit mehreren Softwarekomponenten komplett automatisiert werden. Eine API gewährleistet dies. Sie sollte möglichst offen und ggf. mittels eines Standards wie OpenAPI (https://www.openapis.org [23.10.2020]) dokumentiert sein. Ein Beispiel für APIs sind die Schnittstellen von Wikidata, an die man automatisierte Datenabfragen stellen kann.

Konformität mit Regelungen zum Datenschutz und der Datensparsamkeit
Für den Einsatz in der universitären Forschung und der Lehre entscheidet die Frage nach dem Datenschutz vielfach darüber, ob die Software überhaupt genutzt werden darf. Eine Einschätzung der Regelungen, sofern sie sich nicht eindeutig auf die europäischen Rahmenrichtlinien beziehen, ist teilweise jedoch kaum möglich und Gegenstand von juristischen Diskussionen. Auch hier kann die Rezension bereits durch den Hinweis auf das Thema einen wichtigen Service für die Leser bringen. Dies gilt auch für Datensparsamkeit und das Hinterfragen von Registrierungsvorgängen, Cookies und Ähnlichem.

— **Werden die Gesetze (z. B. zu Datenschutz, Kartendarstellungen etc.) des Einsatzlandes durch die Software eingehalten?** Es muss damit gerechnet werden, dass die Software in verschiedenen Regionen und Ländern verwendet wird, welche jeweils spezifische Gesetze haben, die sich auf die Ausführung/Installation der Software auswirken. Ein Beispiel sind Cloud-Anwendungen, die für einen nordamerikanischen Nutzerkreis entwickelt wurden und mit der EU-weit gültigen Datenschutzgrundverordnung nicht kompatibel sind.
— **Welche Daten speichert die Anwendung zu welchem Zweck und wie lange? Werden Daten an Dritte übertragen?** In vielen Softwareanwendungen werden Nutzerdaten zur Verbesserung der Anwendungen anonymisiert erfasst. Manche Softwareanbieter sammeln jedoch weitaus mehr Daten und übertragen diese z. B. auch an Drittanbieter. Aus der Softwaredokumentation heraus und vor allem beim Erstbesuch einer Webanwendung sollte ersichtlich werden, welche Daten erhoben und wie lange sie gespeichert werden sowie ob die Zustimmung des Benutzers eingeholt wird. In einer Rezension könnte zusätzlich hinterfragt werden, ob der Zweck der Datenerfassung gerechtfertigt scheint, wie etwa zur Verbesserung der Software, oder anderen Interessen dient.

Entwickler-Perspektive
Anwendungsentwickler sind zwar auch Nutzer einer Software, haben jedoch wegen zusätzlicher Interessen und weiterer Anwendungsszenarien einen anderen Blick auf Software. Eine gute Zusammenfassung der Entwicklerperspektive auf die Softwarequalität bieten Jung et al. (2004). Dort werden die Vorgaben des ISO/IEC 9126 Standards (https://de.wikipedia.org/wiki/ISO/IEC_9126 [23.10.2020]) in Beispielen ausgeführt.

Die folgenden Fragen fokussieren sich auf den Programmcode und die Softwarearchitektur. Diese können nur beurteilt werden, wenn der Quellcode offen einsehbar (Open Source) ist, was bei proprietärer Software meist nicht oder nur eingeschränkt möglich ist.

Dokumentation und Tests

Eine umfangreiche Dokumentation bietet ein umfassendes Verständnis der Intention, der Ausgereiftheit und des aktuellen Entwicklungsstands der Software. Sie ist zudem essenziell, um die Software zu erweitern. Eine solide Dokumentation wird schließlich auch eine größere Community von potenziellen Entwicklern ansprechen.

Man unterscheidet mehrere Bestandteile der Dokumentation, und zwar die Dokumentation:

des *Quellcodes*, d. h. von Klassen oder einzelnen Methoden,

des *Build-Prozesses*, d. h. wie die Software aus dem Quellcode erzeugt wird,

des *Softwaretestprozesses*, d. h. welche Testfälle von der Software berücksichtigt wurden.

Schließlich gibt es eine Entwicklerdokumentation mit Benutzungsbeispielen.

Software Repositorien wie GitHub (https://github.com [23.10.2020]) oder GitLab (https://gitlab.com [23.10.2020]) bieten oft Vorlagen oder Best Practices, um diese Anforderungen in verschiedenen Programmiersprachen umsetzen zu können.

Ein wichtiger Begriff in dem Zusammenhang von Dokumentation und Tests ist auch der der *Continuous Integration* (CI; Kontinuierliche Integration). Damit wird die fortlaufende Zusammenfügung der Einzelkomponenten einer Anwendung bezeichnet. Hierfür werden Routinen erstellt, die vielseitige Aufgaben durchführen. So kann beispielsweise automatisiert die Erstellung einer Quellcodedokumentation, die Durchführung eines Softwaretests oder die Erzeugung einer ausführbaren Datei aus dem Quellcode (Releasefile, EXE-Datei) angestoßen werden. Dabei werden diese Routinen meist nach jeder Änderung des Quellcodes erneut ausgeführt, wodurch alle Programmkomponenten aktuell bleiben.

– **Ist eine Quellcodedokumentation vorhanden und ggf. eine HTML-Variante davon verfügbar?** Best Practices sind hier beispielsweise Quellcodedokumentationen mit Doxygen (https://www.doxygen.nl/index.html [23.10.2020]), JavaDoc (http://www.oracle.com/technetwork/java/javase/documentation/javadoc-137458.html [23.10.2020]), JsDoc (https://jsdoc.app [23.10.2020]) oder das für Python beliebte ReadTheDocs (https://readthedocs.org [23.10.2020]). Alle genannten Tools erzeugen eine HTML-Repräsentation der Dokumentation, welche idealerweise auch online bereitgestellt werden sollte, etwa als GitHub-Page.

– **Ist der Build-Prozess dokumentiert und ggf. mittels Buildingscripts automatisiert?** Neben dem grundlegenden Verständnis der Programmarchitektur ist das Wissen um die Bauanleitung, dem Build-Prozess, der Software wichtig. Eine Dokumentation dieses Prozesses sollte Informationen über den funktionierenden Erstellungsprozess der Software enthalten. In der Vergangenheit wurde mit Anleitungen in README-Dateien oder ähnlichen natürlichsprachlichen Beschreibungen gearbeitet. Inzwischen ist es etablierter Standard, maschinenlesbare Bauanleitungen (*Buildscripts*) bereitzustellen. Diese beschreiben eindeutig und maschinenlesbar, wie die Software erstellt wurde und erlauben oft das Starten des Build-Prozesses mit einem einzigen Skript. Abhängigkeiten der Software in Bezug auf verwendete Libraries (Programmbibliotheken) werden mit Buildscripts ebenfalls dokumentiert. Beispiele für solche Skripte finden sich z. B. in Apache Maven (https://maven.apache.org [23.10.2020]) oder Gradle (https://gradle.org [23.10.2020]).

\+ **Ist die Dokumentation aktuell und adressiert sie alle Funktionen des Programms?** Das letzte Bearbeitungsdatum einer Dokumentation ist meist einem Zeitstempel zu entnehmen, wenn die Dokumentation mit einem der oben genannten Dokumentationswerkzeuge erstellt wurde. Dieses Datum sollte mit dem Veröffentlichungsdatum der aktuellen Software übereinstimmen oder aktueller sein. Werden im besten Fall die Prinzipien der Continuous Integration angewendet, so ist die Erzeugung und Bereitstellung der Dokumentation direkt mit im Entwicklungsprozess der Anwendung integriert, wie beispielsweise das GitHub Repositorium des SPARQLing-Unicorn-QGIS-Plugins (https://github.com/sparqlunicorn/sparqlunicornGoesGIS [23.10.2020]). Hier erstellt ein automatisierter Prozess die Dokumentation bei jeder Veränderung des Quellcodes neu und publiziert sie auf der GitHub Seite des Repositoriums (https://sparqlunicorn.github.io/sparqlunicornGoesGIS/ [23.10.2020]).

– **Gibt es eine Entwicklerdokumentation, welche eine evtl. Softwareerweiterung fördert?** Eine Entwicklerdokumentation bietet einem Entwickler den Einstieg in die Beschäftigung mit der Erweiterung einer Software über die einfache Nutzung hinaus. Eine aussagekräftige README-Datei, die kurz die Verwendung des Programms mit den Standardeinstellungen demonstriert, ist eine Mindestanforde-

rung. Ein Beispiel dafür ist Bibtex_JS (https://github.com/pcooksey/bibtex-js [23.10.2020]). Idealerweise werden zusätzlich Beispieldaten für ein besseres Verständnis des Programmablaufs beigelegt und ggf. weitere häufig verwendete Anwendungsfälle der Software in Beispielen vorgestellt. Für das gerade genannte Beispiel trifft dies zu: Bibtex_JS-Beispiele (https://github.com/pcooksey/bibtex-js/tree/master/test [23.10.2020]). Abhängig von der Komplexität der Software kann es sinnvoll sein, ein ggf. auch von einer Nutzercommunity gepflegtes Wiki bereitzustellen, um fortgeschrittene Optionen zu erläutern (siehe Bibtex_JS-Wiki: https://github.com/pcooksey/bibtex-js/wiki [23.10.2020]).

– **Enthält der Quellcode Softwaretests, um die Kernfunktionen zu testen und diese für andere Entwickler aufzuzeigen?** Jede Software kann durch Tests auf ihre antizipierte Funktionalität hin überprüft werden. Entsprechend geben Softwaretests Auskunft über die angedachten Anwendungsfälle und als Testergebnis, welche Funktionen stabil, fehleranfällig oder verbesserungsbedürftig sind. Als Teil eines *Continuous-Integration*-Prozesses können Softwaretests nach jeder Änderung am Quellcode automatisch ausgeführt werden.
– **Wird es dem Entwickler einfach gemacht, die Software zu testen (z. B. Virtuelle Maschine, Dockercontainer, Installer)?** Entwickler sind meist in der Lage, eine Software zu bauen und auszuführen, doch das kann mehr oder weniger aufwändig sein. Tatsächlich werden Softwares, welche leicht installierbar und vor allem testbar sind, sowohl bei Entwicklern, die einen schnellen Einblick in die Funktionalität bekommen wollen, als auch bei anderen Benutzern einen besseren Anklang finden. Es liegt in der Natur der Sache, dass es hierbei, je nach Art der Anwendung, unterschiedliche Ausführungsformen gibt. Eine Webanwendung kann als Beispiel im Internet gehostet sein und dem Benutzer einen Testaccount zur Verfügung stellen (z. B. CWRCWriter; https://cwrc.ca [23.10.2020]), eine Desktopanwendung kann bereits eine installierbare Anwendung oder ein Installerpaket mitliefern. Serveranwendungen können für einen Test als Images von virtuellen Maschinen oder – in letzter Zeit üblicher – als Dockerimages in Portalen wie Dockerhub (https://hub.docker.com [23.10.2020]) für ein einfaches Testen hinterlegt werden.
– **Sind die Entwickler gut erreichbar?** Oft stellt sich bei der Bedienung von Software heraus, dass Funktionalitäten fehlen oder Fehler im Programm vorhanden sind. Dies kann die Verwendung der Software für bestimmte Anwendungsfälle behindern oder deren Ergebnisse signifikant verschlechtern. In solchen Fällen ist die Verfügbarkeit der Entwickler und deren Rückmeldungen auf Supportanfragen ein entscheidendes Kriterium. Es sollte eindeutig kommuniziert werden, wie und wo Fehlerberichte eingereicht werden können und welche Informationen für eine rasche und präzise Bearbeitung erforderlich sind. Ob die Entwickler einen aktiven Austausch pflegen, kann man beispielsweise in dem Issue-Bereich des Softwarerepositoriums (z. B. GitHub oder GitLab) ablesen. Haben die Entwickler Anfragen dort zeitnah und zufriedenstellend beantwortet? Wie ist das Verhältnis von offenen Issues zu bereits geschlossenen? Wie lange dauerte es, bis eine Änderung eingepflegt wurde?
+ **Wird an der Software aktiv gearbeitet?** Ein Indiz dafür ist, dass die Software regelmäßig mit Updates versorgt wird. Welche Änderungen demnächst für das Programm geplant sind und welche Issues für das nächste Release bearbeitet werden sollen, können Entwickler proaktiv in einer Roadmap kommunizieren.
✚ **Ist es möglich, die Softwareentwicklung zu unterstützen?** Damit externe Softwareentwickler die Software erweitern können, ist eine *Contribution Guideline* hilfreich. Sie enthält Angaben dazu, unter welchen Umständen und wie Änderungen an der Software von Dritten angenommen und integriert werden (z. B. https://projectacrn.github.io/latest/developer-guides/contribute_guidelines.html [23.10.2020]).

Qualität der Implementierung

Die Qualität der Implementierung beeinflusst, ob und wie sehr sich das Programm zur Anpassung und Weiterentwicklung eignet und damit auch, wie langlebig es vermutlich sein wird.

– **Entspricht die Implementierung dem Stand der Technik?** Diese Frage muss meist anwendungsbezogen beantwortet werden und verlangt ein technisches Verständnis der Abläufe innerhalb der Anwendung. Einige Aspekte können ohne Fachwissen bewertet werden: Ist die Anwendung auf vielen verschiedenen Endgeräten bedienbar (z. B. Handy, verschiedene Betriebssysteme usw.)? Werden veraltete Technologien vermieden, wie z. B. Adobe Flash oder Java Applets bei Webanwendungen?

- **Zeichnen automatische Testsysteme (etwa auf GitHub) offene Sicherheitslücken in der Software aus?**
- **Wird *Continuous Integration* zur Absicherung der Implementierungsqualität verwendet?** Für einen Entwickler ist nicht nur das Vorhandensein und die Dokumentation des Quellcodes entscheidend, sondern er wird meistens auch einen Hinweis zur Funktionalität und Kompilierbarkeit des Quellcodes erwarten. *Continuous Integration* kann, wie schon in den vorherigen Abschnitten angesprochen, ein solches Qualitätsmerkmal darstellen. Der *Continuous-Integration*-Prozess überprüft die Kompilierbarkeit der Software nach jeder Änderung und kann diese am Repositorium mit einer Statusanzeige sichtbar machen. Es ist ein Zeichen von gut gewartetem Quellcode, wenn dieser in der aktuell vorliegenden Version und ggf. in der aktuellen Entwicklerversion kompiliert.

Software-Eckdatentabelle

In der Tabelle **Abb. 1** stellen wir einige Punkte vor, die tabellarisch am Ende oder Anfang einer Rezension genannt werden können, um einen schnellen Überblick über eine Software zu geben, ähnlich wie in Buchrezensionen Titel, Verlag und ISBN wertneutral gelistet werden. Diese Tabelle kann ggf. auch um Hardwareanforderungen ergänzt werden, wenn diese als besonders relevant empfunden werden. Als Beispiel haben wir die GIS-Anwendung QGIS gewählt.

Wie geht es weiter?

Die Diskussion über die Bewertung von Forschungssoftware in der Archäologie und ihren Nachbardisziplinen hat gerade erst begonnen und es wurde bisher kein allgemeiner Konsens erarbeitet. Unsere hier vorgestellten Überlegungen und der Fragenkatalog sollen daher als Diskussionsanstoß und Impuls zur Etablierung eines gemeinsamen Verständnisses der Anforderungen an eine Software-Rezension verstanden werden. Wie eingangs angemerkt, werden Stellungnahmen aus verschiedensten Perspektiven und Disziplinen per GitHub oder E-Mail erbeten. Der Katalog soll daher mitnichten als in Stein gemeißelt verstanden, sondern soll in Zukunft von Archäologen und Research Software Engineers modifiziert und fortgeschrieben werden.

Literatur

Anzt, H., Bach, F., Druskat, S., Löffler, F., Loewe, A., Renard, B.Y. et al. (2020). An Environment for Sustainable Research Software in Germany and Beyond: Current State, Open Challenges, and Call for Action. *F1000Research 9, 295*. https://doi.org/10.12688/f1000research.23224.1.

Bach, F., Druskat, S., Katerbow, M., Loewe, A., Seemann, G. (2019). *Herausforderungen für die nachhaltige Entwicklung, Bereitstellung und Pflege von Forschungssoftware in Deutschland.* Berlin: deRSE. https://de-rse.org/de/conf2019/talk/PVEXDH/ [23.10.2020].

Name:	Der Name der Software, z. B. *„QGIS“*.
Kurzbeschreibung:	Angabe dessen, was die Software leistet, z. B. *„Umfangreiches graphisches Werkzeug zur Raumdatenverarbeitung“*.
Rezensierte Version:	Die Softwareversion, die für die Rezension genutzt wurde, z. B. *„3.10.10 LTR“*.
Plattform:	(Betriebs-) Systeme, auf denen die Software genutzt werden kann, z. B. *„Windows, macOS, Linux, BSD, Android“*.
Webseite:	URL, unter der weitere Informationen abgerufen werden können, z. B. *„https://qgis.org“*.
Lizenzierung:	Unter welcher Softwarelizenz wurde die Software veröffentlicht, z. B. *„Open Source mit GNU General Public License (GPL)“*.
Kosten:	Ggf. anfallende laufende oder einmalige Lizenzgebühren, z.B. *„kostenfrei“*.
Ein- und Ausgabeformate:	Welche Dateiformate die Software verarbeiten kann, z. B. *„Geodatenbanken“* (SpatiaLite, PostGIS, MSSQL, ...), *„Web-Geodaten Dienste“* (WMD/WMTS, Vector Tiles, XYZ Tiles, WFS, ...), *„Geo-Vektordatenformate“* (ESRI Shapefile, Geopackage, ...), *„Geo-Rasterdatenformate“* (GeoTIFF, ...), *„Tabellendaten“* (CSV, TXT, ...) und weitere Datentypen (für QGIS sind ungewöhnlich viele Datenformate zu beachten, deren komplette Listung hier den Rahmen sprengen würde).

Abb. 1 Vorschlag für die Angabe der Eckdaten einer Software am Beispiel von QGIS.

Baxter, R., Chue Hong, N., Gorissen, D., Hetherington, J., Todorov, I. (2012). The Research Software Engineer. *Digital Research 2012*. https://web.archive.org/web/20180202071627/http://digital-research-2012.oerc.ox.ac.uk/papers/the-research-software-engineer/at_download/file [23.10.2020].

Bezjak, S., Clyburne-Sherin, A., Conzett, P., Fernandes, P., Görögh, E., Helbig, K. et al. (2018). Open Science Training Handbook. *Zenodo, 4.4.2018*. https://doi.org/10.5281/zenodo.1212496.

DFG (2019). *Leitlinien zur Sicherung guter wissenschaftlicher Praxis – Kodex*. Bonn: DFG. https://www.dfg.de/download/pdf/foerderung/rechtliche_rahmenbedingungen/gute_wissenschaftliche_praxis/kodex_gwp.pdf [23.10.2020].

Druskat, S., Spaaks, J. H., Hong, N. C., Haines, R., Baker, J. (2017). Citation File Format (CFF). *Zenodo, 4.11.2019*. https://doi.org/10.5281/zenodo.1003149.

forschungsdaten.info (2020). *Softwareentwicklung in der Wissenschaft*. Konstanz: forschungsdaten.info. https://www.forschungsdaten.info/themen/ethik-und-gute-wissenschaftliche-praxis/softwareentwicklung-und-gute-wissenschaftliche-praxis/ [23.10.2020].

Goedicke, M., Lucke, U. (2020). *Nationale Forschungsdateninfrastruktur für und mit Computer Science (NFDIxCS)*. https://www.dfg.de/download/pdf/foerderung/programme/nfdi/nfdi_konferenz_2020/nfdixcs_abstract.pdf [23.12.2020].

Helmholtz Open Science (2019). *Forschungssoftware*. Bremenhaven: Alfred-Wegener-Institut, Helmholtz-Zentrum für Polar- und Meeresforschung. https://os.helmholtz.de/open-science-in-der-helmholtz-gemeinschaft/forschungssoftware/ [23.10.2020].

Hettrick, S. (2016). *A not-so-brief history of Research Software Engineers*. Edinburgh: Software Sustainable Institute. https://www.software.ac.uk/blog/2016-08-17-not-so-brief-history-research-software-engineers-0 [23.10.2020].

Hettrick, S., Antonioletti, M., Carr, L., Chue Hong, N., Crouch, S., De Roure, D., et al. (2014). UK Research Software Survey 2014. *Zenodo, 4.12.2014*. https://doi.org/10.5281/zenodo.14809.

Jung, H.-W., Kim, S.-G., Chung, C.-S. (2004). Measuring software product quality: A survey of ISO/IEC 9126. *IEEE software, 21*, 88-92.

Katerbow, M. & Feulner, G. (2018). Handreichung zum Umgang mit Forschungssoftware. *Zenodo, 27.2.2018*. https://doi.org/10.5281/zenodo.1172970.

Katz, D.S., Niemeyer, K.E., Smith, A.M., Anderson, W.L., Boettiger, C., Hinsen, K. et al. (2016). Software vs. data in the context of citation (No. e2630v1). *PeerJ Preprints, 10.12.2016*. https://doi.org/10.7287/peerj.preprints.2630v1.

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E. et al. (2019). Towards FAIR principles for research software. *Data Science, 3(1)*, 37-59. https://doi.org/10.3233/DS-190026.

Löffler, F. (2020). *Nationale Forschungsdateninfrastruktur für wissenschaftliche Software (NFDI4RSE)*. https://www.rse4nfdi.de/de/index.html [23.10.2020].

Scheliga, K., Pampel, H., Bernstein, E., Bruch, C., zu Castell, W., Diesmann, M. et al. (2017). Helmholtz Open Science Workshop „Zugang zu und Nachnutzung von wissenschaftlicher Software". *#hgfos16: Report Nov. 2016*. https://doi.org/10.2312/lis.17.01.

Schmidt, S. C. & Marwick, B. (2020). Tool-Driven Revolutions in Archaeological Science. *Journal of Computer Applications in Archaeology, 3*, 18-32. https://doi.org/10.5334/jcaa.29.

Stallman, R. (2001). *What is Free Software? The Free Software Definition*. https://www.gnu.org/philosophy/free-sw.html.en [23.12.2020].

Wilkinson, M.D., Dumontier, M., Aalbersberg, Ij.J., Appleton, G., Axton, M., Baak, A. et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data, 3*, 160018. https://doi.org/10.1038/sdata.2016.18.

Über die Autoren
(in alphabetischer Reihenfolge)

Timo Homburg, Hochschule Mainz, studierte Informatik mit den Schwerpunkten Computerlinguistik (Assyrologie), Semantic Web und Sinologie. In den letzten Jahren arbeitete er im Bereich der GIS-Anwendungen in der Geoinformatik. Seine Doktorarbeit beschäftigt sich mit der besseren Integration von Geodaten in ein Semantic-Web-Umfeld. Sein aktuelles Forschungsvorhaben beschäftigt sich mit der Schaffung digitaler Standards im Bereich Assyrologie und der Best-Practice-Dokumentation von Ausgrabungen mit Keilschrifttexten – eine Erweiterung seiner Publikationen zum Thema *„Keilschrift in den Digital Humanities"*.

https://orcid.org/0000-0002-9499-5840

Anne Klammt, Deutsches Forum für Kunstgeschichte Paris, hat über ein landschaftsarchäologisches Thema promoviert und ist seit 2020 als Forschungsleiterin für die Weiterentwicklung der Digitalen Kunstgeschichte am Deutschen Forum für Kunstgeschichte Paris verantwortlich.

https://orcid.org/0000-0003-3697-9241

Hubert Mara, mainzed & Hochschule Mainz, studierte Informatik an der Technischen Universität Wien. Dort befasste er sich bereits mit digitalen Methoden in der Archäologie zur Keramikanalyse. Im Anschluss war er Marie-Curie-Fellow an der Universität Florenz im Rahmen des Cultural Heritage Informatics Research Oriented Network (CHIRON). Er promovierte an der Universität Heidelberg im Interdisziplinären Zentrum für Wissenschaftliches Rechnen (IWR). Dort entstand das GigaMesh Software Framework (https://gigamesh.eu) und das Forensic Computational Geometry Laboratory (FCGL). Seit Juni 2020 ist er Geschäftsführer des mainzed und wissenschaftlich am Institut für Raumbezogene Informations- und Messtechnik der Hochschule Mainz tätig.

https://orcid.org/0000-0002-2004-4153

Clemens Schmid, Max-Planck-Institut für Menschheitsgeschichte Jena, studierte Prähistorische, Historische und Naturwissenschaftliche Archäologie und Informatik in Tübingen und Kiel. Nach seinem Studienabschluss arbeitete er mit computerbasierter Datenanalyse in verschiedenen archäologischen Forschungsprojekten an der Universität Kiel, dem Römisch-Germanischen Zentralmuseum in Mainz und der Universität Bern. Im Augenblick ist er für ein Promotionsprojekt zur quantitativen Nachhersage von Mobilitätsverhalten mit genetischen und historisch-linguistischen Daten in der Abteilung für Archäogenetik des Max-Planck-Instituts für Menschheitsgeschichte in Jena beschäftigt.

https://orcid.org/0000-0003-3448-5715

Sophie Charlotte Schmidt, Deutsches Archäologisches Institut, studierte Altertumswissenschaften und Prähistorische Archäologie an der Freien Universität Berlin. Im Anschluss arbeitete sie als Wissenschaftliche Mitarbeiterin im Bereich Archäoinformatik an den Universitäten Köln und Bonn, derzeit ist sie bei dem Konsortialprojekt NFDI4Objects am Deutschen Archäologischen Institut beschäftigt.

https://orcid.org/0000-0003-4696-2101

Florian Thiery, Römisch-Germanisches Zentralmuseum – Leibniz-Forschungsinstitut für Archäologie, ist studierter Geodät und Research Software Engineer (RSE) im Arbeitsbereich Wissenschaftliche IT, Digitale Plattformen und Tools am Römisch-Germanischen Zentralmuseum in Mainz. Er ist Initiator der Research Squirrel Engineers, einem Netzwerk für RSEs in den Digital Humanities und Communityprojekte mit Fokus auf freie und offene Daten, sowie Linked Open Data. Im Rahmen eines Stipendiums des Wikimedia Fellow-Programm Freies Wissen beschäftigt er sich mit der Modellierung von irischen (Ogham) Steinen und deren Publikation im Wikimedia Universum.

https://orcid.org/0000-0002-3246-3531

Martina Trognitz, Austrian Centre for Digital Humanities and Cultural Heritage, ÖAW Wien, studierte Computerlinguistik und Klassische Archäologie in Heidelberg. 2012 bis 2017 war sie im Projekt IANUS am Deutschen Archäologischen Institut tätig. Seit 2017 leitet sie das digitale Archiv ARCHE des ACDH-CH an der Österreichischen Akademie der Wissenschaften. Sie bearbeitet ein Promotionsprojekt zur maschinellen Analyse von mehrseitigen ägäischen Siegeln der Bronzezeit.

https://orcid.org/0000-0003-0485-6861

Martina Trognitz M.A.
Austrian Centre for Digital Humanities
and Cultural Heritage
ÖAW Wien
Martina.Trognitz@oeaw.ac.at