AR 3203

Daniel Röwenstrunk

Langzeitverfügbarkeit von wissenschaftlicher Software im Bereich historisch-kritischer Musikedition

Zusammenfassung: Die Langzeitverfügbarkeit von Forschungsergebnissen und der dafür häufig notwendige langfristige Erhalt der Lauffähigkeit von Software ist eine gemeinschaftliche Herausforderung für Forschung, Softwareentwicklung und Gedächtnisinstitutionen. Es stehen verschiedene Maßnahmen wie Standardisierung, Emulation oder auch die Weiterentwicklung durch Open-Source-Communities zur Verfügung, die in unterschiedlichen Kontexten für unterschiedliche Arten von Software zum Einsatz kommen können.

Schlüsselwörter: Forschungssoftware, Langzeitverfügbarkeit, Digitale Musikedition

Long-term Availability of Research Software in the Field of Historic Critical Music Editions

Abstract: Long-term availability of research findings is a challenge for research, software development, and memory institutions altogether. This often includes the necessity to keep research software running over a long time. There are various methods and actions like standardization, emulation, or the development of research software by an open source community which can be taken to achieve this goal.

Keywords: Research Software, Long-term Availability, Digital Music Editions

1 Einleitung

Bereits in den Anfängen der Digitalisierung historisch-kritischer Musikeditionen wurde durch die Entwicklung der Software *Edirom*[1] eine Besonderheit im Vergleich zu vielen anderen Forschungsgebieten[2] deutlich: Nicht die Entwicklung eines Werkzeugs für die Erarbeitung der Edition stand zuerst im Vordergrund, sondern eine Software für die *Darstellung* der Editions*ergebnisse* – in diesem Fall die Ergebnisse der Carl-Maria-von-Weber-Gesamtausgabe.[3] Besonders in der Vermittlung

[1] Vgl. https://edirom.github.io/Edirom-Online/.

[2] Eine ähnliche Situation ist sicherlich auch in anderen Forschungsgebieten zu finden, wenn z.B. die Forschungsergebnisse aufgrund der Menge der Daten nicht ohne Visualisierung erfassbar sind.

[3] Die erste Version der *Edirom*-Software entstand für das 2005 veröffentlichte Klarinettenquintett op. 34 von Carl Maria von Weber.

der komplexen Editionszusammenhänge schien ein großes Potential des digitalen Mediums zu liegen. So wurde die Notwendigkeit für den Einsatz digitaler Medien unter anderem in dem Wunsch deutlich, die Lesarten in Kritischen Berichten an der Stelle sichtbar werden zu lassen, an der sie relevant sind – im Notentext selbst. Erst durch das digitale Medium war dieser Wunsch erfüllbar.[4]

Allein dieser Umstand veranschaulicht, dass die Betrachtung der Langzeitverfügbarkeit wissenschaftlicher Software eine genauere Differenzierung des Begriffs *Software* benötigt. Im Folgenden sollen *Forschungssoftware* und *Vermittlungssoftware* für Forschungsergebnisse für den Kontext dieses Artikels definiert werden, um eine klare Abgrenzung der Aufgaben und Anforderungen vornehmen zu können. Aber auch die *Forschungsdaten* müssen in dieser Betrachtung berücksichtigt werden, da sie eigene Aufgaben und Anforderungen im Rahmen der langfristigen Verfügbarkeit von Forschungsergebnissen haben.

Forschungssoftware ist Software, die durch Generierung, Prozessierung und Analyse von Daten zur Herleitung bzw. zum Aufbau von Forschungsergebnissen eingesetzt und in der Regel im akademischen Umfeld entwickelt wird.[5]

Forschungsdaten sind Daten, die die Ergebnisse der wissenschaftlichen Forschung beinhalten. „Die Vielfalt solcher Daten entspricht der Vielfalt unterschiedlicher wissenschaftlicher Disziplinen, Erkenntnisinteressen und Forschungsverfahren. Zu Forschungsdaten zählen u.a. Messdaten, Laborwerte, audiovisuelle Informationen, Texte, Surveydaten, Objekte aus Sammlungen oder Proben, die in der wissenschaftlichen Arbeit entstehen, entwickelt oder ausgewertet werden.“[6]

Vermittlungssoftware für Forschungsergebnisse ist Software, die für die Publikation, Darstellung, Aufbereitung oder das Verständlichmachen von Forschungserkenntnissen eingesetzt wird oder gar notwendig ist.

Die Sicherung der langfristigen Verfügbarkeit von wissenschaftlicher Software wird häufig auf die Konzepte, Algorithmen und Methoden innerhalb der Software selbst bezogen. Geht es um diese, ist die Sicherung von Beschreibungen, Quellcode und Abstraktionen des Aufbaus und der Funktionsweise der Software in Form von Diagrammen und Modellen sinnvoll. Eine Betrachtung der Software als Forschungsdaten, wie es die DFG in ihren „Leitlinien zum Umgang mit Forschungsdaten“ vornimmt:

---

[4] Vgl. Bohl, Benjamin; Kepper, Johannes; Röwenstrunk, Daniel: Perspektiven digitaler Musikeditionen aus der Sicht des Edirom-Projekts. In: *Die Tonkunst*, 3 (5) (2011) 275.

[5] Vgl. Hettrick, Simon: Research Software Sustainability. Report on a Knowledge Exchange Workshop. 2016, 7.

[6] Deutsche Forschungsgemeinschaft: Leitlinien zum Umgang mit Forschungsdaten. 2015, 1. http://www.dfg.de/download/pdf/foerderung/antragstellung/forschungsdaten/richtlinien_forschungsdaten.pdf.

> „Software und Simulationen können ebenfalls zentrale Ergebnisse wissenschaftlicher Forschung darstellen und sollten daher ebenfalls unter den Begriff Forschungsdaten gefasst werden“[7]

ist dann legitim und richtig, da die Software aufgrund der *statischen* Daten, die sie beschreiben, nachvollziehbar und letztlich reproduzierbar wird. Die Software muss nicht lauffähig sein, um ihre Funktion und somit ihren wissenschaftlichen Wert nachvollziehen zu können. Die Langzeitarchivierung von statischen Daten bezieht sich im Wesentlichen auf eine Herausforderung hinsichtlich der Speichermedien und der Software zum Lesen und Schreiben der Daten.[8] Durch die weitestgehend vollständige Nutzung reiner Textformate wie XML und gängiger Bild-, Audio- und Video-Formate für die im Kontext der digitalen Edition anfallenden Daten ist diese Frage im Rahmen dieses Artikels zu vernachlässigen.

An dieser Stelle steht aber nicht die Software als Ergebnis wissenschaftlicher Forschung in Form statischer Daten, sondern die *Nutzung* der Software und somit das Erhalten der Lauffähigkeit der Software im Rahmen der langfristigen Verfügbarkeit von Forschungsergebnissen im Fokus. Denn, wenn die Forschungsdaten erst durch die Software verständlich und nachvollziehbar werden,[9] kann nur von einer tatsächlichen Verfügbarkeit der Forschungsergebnisse gesprochen werden, wenn eben diese Software lauffähig bleibt und somit zur Vermittlung der Forschungsergebnisse nutzbar ist. Im Rahmen dieses Artikels wird also nicht die Langzeitverfügbarkeit von *Forschungssoftware*, sondern die langfristige Lauffähigkeit von *Vermittlungssoftware* im Kontext der Langzeitverfügbarkeit von Forschungsergebnissen betrachtet – aus der sicherlich eher Anforderungen stellenden Sicht der Forschung.

2 Software für Publikation und Vermittlung

Für die Publikation und Vermittlung von digitalen Forschungsergebnissen wird Software an unterschiedlichen Stellen eingesetzt. Es lassen sich drei Ebenen mit je eigenen Anforderungen und Besonderheiten identifizieren: 1. die Bereitstellung von Forschungsdaten, 2. der maschinenlesbare Zugang zu Forschungsergebnissen und 3. die Vermittlung von Forschungsergebnissen für Menschen.

---

[7] Deutsche Forschungsgemeinschaft (Anm. 6) 1.

[8] Hettrick, Simon (Anm. 5) 8.

[9] Im Kontext digitaler Musikeditionen wird es schnell ersichtlich, dass z.B. die codierten Notentexte nicht ohne eine Darstellungssoftware erfasst werden können. Hier können schnell mehrere hunderttausend Zeilen Quelltext für einige hundert Takte Orchestermusik zusammenkommen. Wenn dann noch mehrere Varianten innerhalb der Codierung erfasst sind, steigt die Komplexität noch einmal drastisch an. Aber auch Zusammenhänge zwischen Notenmaterialien und Audioaufnahmen lassen sich kaum ohne entsprechende Abspielsoftware erfassen.

Die öffentlich zugängliche und dauerhafte **Bereitstellung** von Forschungsdaten ist die Grundlage für die Verfügbarkeit von Forschungsergebnissen. Sie stellt sicher, dass die *statischen* Daten, die sämtliche Forschungsergebnisse beschreiben, zugänglich bleiben und somit das darin enthaltene Wissen grundsätzlich nicht verloren geht. Aufgrund der bereits beschriebenen Notwendigkeit der Nutzung von Software zur Vermittlung der in den Daten steckenden Erkenntnisse kann die reine Bereitstellung für die langfristige Sicherung des Wissens aber nicht hinreichend sein – nur notwendig.

Daten, die in Repositorien vorgehalten werden, müssen – stabil und verlässlich – langfristig in ihrem Zustand bestehen bleiben (Persistenz), sie müssen eindeutig identifizierbar und referenzierbar sein (Persistent Identifier, PID) und ihre Provenienz muss jederzeit erkennbar bleiben.[10] Die so oft als Vorteil der digitalen Welt angeführte Möglichkeit, bereits veröffentlichte Daten im Nachhinein zu verändern, zu korrigieren oder zu erweitern, kann nur dann in Einklang mit der hier geforderten Unveränderlichkeit und Eindeutigkeit der Daten gebracht werden, wenn das Repository über Konzepte der Versionierung von Daten verfügt. Die geforderte Persistenz muss für jede Version eines Datensatzes sichergestellt werden; auch muss jede Version mit Hilfe eines Persistent Identifiers eindeutig bestimmt werden und ihre Provenienz abgelesen werden können. Als Beispiele für die Umsetzung solcher Repositorien können das TextGrid-Repository[11] und auch das DARIAH-DE Repository[12] gelten.

Im Rahmen des durch das Bundesministerium für Bildung und Forschung (BMBF) zwischen 2012 und 2015 geförderten Projekts „Freischütz Digital – Paradigmatische Umsetzung eines genuin digitalen Editionskonzepts“[13] zum Beispiel sind die Daten in das TextGrid-Repository mit dem PID *hdl:11378/0000-0000-16BA-6*[14] eingespeist worden. Innerhalb des Release 0.8.0 sind die Projektergebnisse[15] inkl. der zu den Codierungen gehörenden Schemata unter einer Creative Commons Lizenz veröffentlicht.

Aber schon auf der zweiten Ebene, der Ebene des maschinenlesbaren Zugangs zu Forschungsergebnissen durch **Schnittstellen,** wird deutlich, dass das Ablegen der Forschungsdaten in Repositorien wie denen von TextGrid oder DARIAH allein nicht ausreicht. Ist die reine Bereitstellung

[10] Dies ist nur ein sehr kleiner Ausschnitt der Anforderungen an Datenrepositorien. Weitere Aspekte wie z.B. der Authentifizierung oder der Urheber- und Verwertungsrechte können im Rahmen dieses Artikels unberücksichtigt bleiben.

[11] Vgl. https://textgridrep.org/.

[12] Vgl. https://de.dariah.eu/repository.

[13] Vgl. http://freischuetz-digital.de/.

[14] Aufgelöst werden kann der PID über den Link: http://hdl.handle.net/hdl:11378/0000-0000-16BB-5.

[15] Es konnten nicht alle Daten im Repository abgelegt werden; lediglich die Musikedition, die Textedition und die Metadaten zu den im Projekt erstellten Aufnahmen der Nummern 6, 8 und 9 des Freischütz', die in Kooperation mit der Hochschule für Musik Detmold entstanden sind, konnten berücksichtigt werden. Die Audiodaten selbst waren – zumindest zu dem damaligen Zeitpunkt noch – zu umfangreich.

von Forschungsdaten noch relativ leicht ohne die Kenntnis der Strukturen der Daten selbst möglich,[16] benötigen Schnittstellen, sogenannte *Application Programming Interfaces* (API), genaue Kenntnis über eben diese semantischen Informationen, [17] um eine Suche, Aggregation oder eine Analyse der Daten auf einer feingranularen, aber dateiübergreifenden und somit Zusammenhänge berücksichtigenden Ebene zu ermöglichen. Forschungsnahe Repositorien können die Struktur der enthaltenen Daten in manchen Fällen voraussetzen und Schnittstellen anbieten; dies ist aber nur für Daten oder Teilbereiche von Daten möglich, die sich standardisieren lassen. Eine Einigung eines Faches auf ein Codierungsschema z.B. der *Music Encoding Initiative* (MEI)[18] ist Voraussetzung für eine solche Standardisierung. Die Flexibilität der Auslegung der Guidelines der MEI und die daraus resultierende unterschiedliche Nutzung der Strukturen und Möglichkeiten für einzelne Phänomene aber auch die – für die Wissenschaft notwendige – Erweiterbarkeit des Formats haben zur Folge, dass Schnittstellen – wenigstens zu einem Teil – projektspezifisch gedacht werden müssen.

Soll eine Schnittstelle nicht nur als einmaliger Lieferant einer Teilmenge der Forschungsdaten für eine Nachnutzung dienen, sondern den permanenten und referenzierbaren Zugang zu Informationen auf der Ebene der Maschinenlesbarkeit darstellen, müssen die gleichen Anforderungen an die API gestellt werden, die auch an die Repositorien gestellt werden müssen: die Existenz und stabile Lauffähigkeit der API muss langfristig gesichert werden (Persistenz) und sie muss unter gleichbleibender Adresse erreichbar sein (Persistent Identifier). Versionierung hingegen muss im Kontext von Schnittstellen aus zweierlei Perspektiven bedacht werden: 1. die Schnittstelle muss gezielten Zugriff auf sämtliche Versionen der Daten erlauben und 2. muss die Implementierungs- bzw. Release-Historie der Schnittstelle selbst unter Versionsverwaltung liegen. Ohne die Verfügbarkeit sämtlicher Versionen der Daten ist ein stabiles Referenzieren über die API nicht möglich, aber auch ohne die Kenntnis der ggf. unterschiedlichen Funktionsweisen der Schnittstelle über ihre Versionen hinweg können Ergebnisse bei erneutem Aufruf verfälscht werden. Die Dokumentation und Definition der Schnittstelle ist wiederum Teil der statischen Daten und sollte genauso wie z.B. die Codierungsmodelle in einem Daten-Repository veröffentlicht werden.

[16] Das TextGrid-Repository geht bei den im Rahmen des Projekts „Freischütz Digital“ veröffentlichten musikalischen Daten in der von der *Music Encoding Initiative* vorgegebenen Struktur davon aus, dass es sich um Dateien in der Struktur der *Text Encoding Initiative* handele. So stellt es zumindest die Oberfläche des Repository am 30. November 2017 dar.

[17] Es ist prinzipiell möglich, eine Schnittstelle so zu entwerfen, dass ein Zugriff entweder nur auf Dateiebene erfolgen kann oder eine Anfrage an innere Strukturen von der API nur „durchgereicht“ werden, aber dann muss zwangsläufig eine Verschiebung des Wissens über die inneren Strukturen der Daten an den Abfragenden geschehen; ansonsten unterscheidet sich die API nicht von dem Download einer Datei aus einem Repository.

[18] Vgl. http://www.music-encoding.org.

Als Beispiel für eine musikeditionspezifische Schnittstelle kann die noch in den Kinderschuhen steckende API des *Virtuellen Forschungsverbunds Edirom* (ViFE) gelten,[19] die einen semantischen Zugriff auf Editionsdaten in den Codierungsstandards der MEI und TEI, auf Personen in den Editionskontexten und weiteren projektspezifischen Forschungsergebnissen bieten will. Die ViFE-API ist mit Hilfe der *OpenAPI Specification* und dem dazugehörenden Framework *Swagger*[20] definiert und beschrieben und wird neben einem projektübergreifenden Kern projektspezifische Erweiterungen[21] enthalten.

Der Zugang zu den Forschungsergebnissen für Menschen, der durch den Einsatz von **Vermittlungssoftware** in Form von Visualisierungen o.ä. realisiert werden muss, ist sicherlich der aus Sicht der dafür notwendigen Software der umfangreichste und vielfältigste. Vermittlungssoftware nutzt im Idealfall die auf der zweiten Ebene verfügbaren Schnittstellen, um eine graphische Aufbereitung der zugrundeliegenden Daten vorzunehmen und so Strukturen, Zusammenhänge und Details der Forschungsergebnisse zu vermitteln, die ohne eine visuelle Aufbereitung, allein aus den Daten heraus nur mit großem Aufwand oder eventuell gar nicht[22] erkennbar wären. Auch für Vermittlungssoftware müssen somit die Anforderungen an eine persistente und stabil referenzierbare Existenz gelten. Veröffentlichte Versionen der Software (Releases) müssen nachvollziehbar und dokumentiert und somit reproduzierbar sein. Die Lauffähigkeit der Software muss sichergestellt werden; im Idealfall sogar die Lauffähigkeit sämtlicher Releases.

Für Visualisierungen sollte ebenso wie für Daten in aller Regel eine Standardbildung angestrebt werden. Für die Darstellung von in MEI codierten Notenmaterialien hat sich die Notensatzbibliothek *Verovio*[23] als de facto Standard herauskristallisiert. *Verovio* stellt einen hohen Anteil der in MEI codierbaren Musik dar und kann somit für einen großen Teil der in dieser Form vorliegenden musikalischen Materialien genutzt werden. Aber auch *Verovio* kann die Mehrdeutigkeit, Flexibilität in der Nutzung und Erweiterbarkeit von MEI nicht auflösen – ebenso wenig wie eine Standard-MEI-API es könnte. Die vom Zentrum Musik – Edition – Medien (ZenMEM) gepflegte Darstellungssoftware für digitale Musikeditionen *Edirom Online* ist ebenfalls ein quasi Standard in diesem Bereich. Aber viel

[19] Vgl. dazu https://github.com/Edirom/ViFE-API und Hartwig, Maja; Herold, Kristin; Röwenstrunk, Daniel; Stadler, Peter: Moves like swagger – look into my API and you'll own me. Poster auf der Music Encoding Conference 2017 in Tours (siehe unter http://zenmem.de/confluence/display/ZMEM/2017/05/18/Poster+at+MEC+2017).

[20] Vgl. https://swagger.io/.

[21] Ein Beispiel einer projektspezifischen Implementierung der ViFE-API ist die API der Carl-Maria-von-Weber-Gesamtausgabe. Vgl. dazu http://weber-gesamtausgabe.de/de/Hilfe/API_Dokumentation.html.

[22] Da die Daten sämtliche Erkenntnisse der Forschung beinhalten sollten und kein Wissen in der Software „versteckt" werden sollte, das so in den Daten nicht vorhanden ist, kann grundsätzlich davon ausgegangen werden, dass die Daten theoretisch für das Verständnis der Ergebnisse ausreichen, praktisch ist das in vielen Fällen aufgrund der Komplexität oder Menge der Daten nicht möglich.

[23] Vgl. http://www.verovio.org/.

mehr noch als bei der sehr spezifischen und klar definierten Aufgabe der Anzeige der Noten durch *Verovio* steht eine Standardisierung der Vielfalt der Editionskonzepte und -methoden und der Eigenarten der Editionsgegenstände gegenüber. Bereits in den Anfängen der Entwicklung der *Edirom*-Software im DFG-Projekt „Entwicklung von Werkzeugen für digitale Formen wissenschaftlich-kritischer Musikeditionen“[24] wurde stets die Verallgemeinerung der Darstellungskonzepte für eine möglichst breite Nutzung in möglichst vielen unterschiedlichen Projekten angestrebt, aber es ist im Laufe der seitdem vergangenen zwölf Jahre kein Editionsprojekt ohne spezifische Anpassungen der Software ausgekommen. Die Modifikationen erstrecken sich von einfachen Einstellungen in der Software über graphische und funktionale Anpassungen durch Plugins bis hin zu tiefgreifenden Modifikationen der Funktionsweise der Software selbst. Zum Teil konnten diese Erweiterungen in die allgemeine *Edirom* zurückgeführt und integriert werden, was aber aufgrund der Spezifik der Anpassungen im Kontext des Projekts nicht in jedem Fall sinnvoll ist. Es entstehen dadurch neben den Versionen der allgemein verfügbaren *Edirom Online* noch projektspezifische Derivate, die wiederrum eine eigene Versionshistorie mitbringen.[25]

Im Projekt *Freischütz Digital* ist eine starke Erweiterung der Darstellungssoftware *Edirom Online* vorgenommen worden.[26] Die Modifikationen sind, soweit sie für andere Projekte adaptierbar waren, zurück in die allgemeine *Edirom Online* überführt worden, wobei ein eigenes Derivat der *Edirom Online* unabdingbar blieb. Darüber hinaus sind sogenannte *Demonstratoren* entwickelt worden, die gezielt einzelne Konzepte des Editionsmodells oder einzelne Ergebnisse hervorheben sollten;[27] diese Softwarekomponenten sind sehr spezifisch für den Kontext des Projekts, stellen aber auch in vielen Fällen genau die Innovationen des Editionsmodells dar und sind somit von besonderer Relevanz.

Software wird im Kontext der Publikation und Vermittlung der Forschungsergebnisse auf unterschiedlichen Ebenen, zu unterschiedlichen Zwecken und in mehreren Versionen und sogar Derivaten eingesetzt. Eine Standardbildung ist sowohl für die Datenformate als auch für die Visualisierungen anzustreben, wobei immer davon ausgegangen werden muss, dass dies nur zu einem Teil möglich ist und projektspezifische Anpassungen notwendig sind und bleiben.

[24] Vgl. http://www.edirom.de/edirom-projekt/.

[25] Über den Mechanismus der sogenannten *Forks* (vgl. https://help.github.com/articles/about-forks/) können Repositories in GitHub so kopiert werden, dass ein Rückschluss auf das ursprüngliche Repository erhalten bleibt. Im Falle der *Edirom Online* ist das Repository mehrfach von Editionsprojekten kopiert worden (vgl. dazu https://github.com/Edirom/Edirom-Online/network).

[26] Veröffentlich auf GitHub unter https://github.com/Freischuetz-Digital/Edirom-Online/

[27] Vgl. http://freischuetz-digital.de/demos.html.

3 Lauffähigkeit von Software

Warum ist der Einsatz von Software generell und von vielen verschiedenen projektspezifischen Varianten im Besonderen eine Herausforderung für die Langzeitverfügbarkeit von Forschungsergebnissen? Die Antwort ist einfach: *Software altert*. Wie Engels et al. in ihrem Beitrag „Design for Future – Legacy-Probleme von morgen vermeidbar?“ beschreiben, altert Software relativ zu ihrer Umgebung, in der sie eingesetzt wird.[28] Die Umgebung – dazu gehören Betriebssystem, Laufzeitumgebungen, genutzte Bibliotheken, clientseitig ggf. Browser, aber auch Hardware und Infrastruktur – wird kontinuierlich weiterentwickelt.[29] „Wird die Software nicht ebenfalls kontinuierlich weiterentwickelt, so altert sie rasch relativ zu ihrer Umgebung.“[30] Verändert sich also in dem Gesamtgefüge der Abhängigkeiten aus Software, Hardware und Infrastruktur um die zu erhaltende Software herum etwas, kann die Lauffähigkeit der Software eingeschränkt oder ggf. gar nicht mehr gegeben sein.

Für den Umgang mit alternder Software sind unterschiedliche Ansätze und Maßnahmen vorstellbar, die in verschiedenen Kontexten zu unterschiedlichen Zeiten sinnvoll sind. Viele von ihnen sollen die Abhängigkeiten reduzieren, um so das Problemfeld zu beschränken. Es ist aber offensichtlich, dass immer einige Abhängigkeiten bestehen bleiben werden.[31] Nachfolgend werden ausgewählte Ansätze und Maßnahmen,[32] die speziell für den Bereich der digitalen Musikedition relevant sind, kurz dargestellt.

Die **Standardbildung** sowohl auf der Ebene der Daten (MEI, TEI, JPG, etc.) als auch zur Beschreibung von Softwarefunktionen (*Unified Modeling Language*[33]), der Definition von Schnittstellen (*Swagger*[34]), aber auch bei der Wahl der einzusetzenden Software (*Verovio*, *Edirom Online*, etc.) bietet gerade in Bezug auf die Reduktion der Abhängigkeiten ein enormes Potential, wenn man die Gesamtheit der langfristig zur Verfügung zu haltenden Forschungsergebnisse betrachtet. Wenn die Darstellung von Notenmaterialien z.B. grundsätzlich durch eine Kombination aus in MEI codierten Daten und der Notensatzbibliothek *Verovio* vorgenommen werden würde, könnten zur Verfügung stehende Ressourcen an genau dieser Stelle gebündelt werden, um die Lauffähigkeit genau dieser

[28] Engels, G.; Goedicke, M.; Goltz, U. et al.: Informatik Spektrum (2009) 32: 393. https://doi.org/10.1007/s00287-009-0356-3.

[29] Unter Umständen werden einzelne genutzte Bibliotheken oder Softwarebestandteile Dritter auch nicht weiterentwickelt, was die Komplexität des Problems zusätzlich erhöhen kann.

[30] Engels, G., Goedicke, M., Goltz, U. et al. (Anm. 28) 393.

[31] Wie schon am Beispiel der Demonstratoren des Projekts *Freischütz Digital* gezeigt wurde.

[32] Weitere Maßnahmen finden sich u.a. in Hong, Neil Chue; Crouch, Steve; Hettrik, Simon; et al.: Software Preservation Benefits Framework. Software Sustainability Institute, 2010.

[33] Vgl. http://www.uml.org/.

[34] Vgl. https://swagger.io/.

Software sicherzustellen.[35] Auch wenn die Software nicht weiter gepflegt oder an die Umgebung angepasst werden könnte,[36] wäre lediglich der Austausch oder die Neuentwicklung *einer* Softwarekomponente notwendig, um den Zugang zu den Daten nicht zu verlieren. Die Infrastrukturen, die für die standardisierten Daten und Softwarekomponenten aufgebaut werden, müssen die Anteile an den Daten oder auch Visualisierungen, die nicht durch ein Standard-Codierungsformat oder eine Standardsoftware abgebildet werden können, zulassen und ggf. mit Hinweis auf deren Existenz ignorieren, um eine möglichst hohe Kompatibilität gewährleisten zu können. Darüber hinaus müssen für genau diese Anteile andere Strategien gefunden werden, um Funktionalität nicht einschränken und den Zugang zu spezifischen Forschungsergebnissen nicht zu verlieren.

Wird im Rahmen der Entwicklung von Software eine strikte **Modularisierung** vorgenommen, lassen sich die Abhängigkeiten einzelner Module ebenfalls stark minimalisieren, da so entstehende Module zum einen besser als Standardwerkzeug für eine bestimmte Aufgabe genutzt werden können, zum anderen aber auch durch Abhängigkeiten notwendig werdende Anpassungen im Idealfall nur ein einzelnes Modul betreffen. Eine Herausforderung für Modularisierung ist sicherlich die Definition sinnvoller und somit modularisierbarer Teilaufgaben und deren Beschreibung, aber auch die Definition der Schnittstellen der Module, da sie in aller Regel miteinander interagieren müssen. Ein Ansatz zur Modularisierung von Software sind *Micro Services*, wie sie z.B. für das Curation Center der California Digital Library eingesetzt werden.[37] *Micro Services* sind unabhängige, kleine Softwarekomponenten, die eine einzige Aufgabe (einen Service) in einer Architektur aus vielen *Micro Services* bereitstellen. Übertragen auf die Funktionalität von *Edirom Online* könnte die Darstellung von Notenfaksimiles mit der Funktion des taktweisen Weiterblätterns ein solcher *Micro Service* sein. Dieser Service könnte in einer digitalen Edition (mehrfach) genutzt werden. Ändert sich nun die Abfragesyntax für den Bildserver z.B. zur Syntax des *International Image Interoperability Framework,*[38] muss lediglich der eine *Micro Service* angepasst werden.

Auch die explizite Definition von Abhängigkeiten durch die Angabe von **Versionen** eingesetzter Software oder Softwarebibliotheken dient der Reduktion der Komplexität der Abhängigkeiten und somit des Wartungsaufwands. Dies ist aus Sicht der adaptierten Software und aus Sicht der adaptierenden Software zu betrachten; die adaptierte Software muss in eindeutig identifizierbaren

[35] Durch einen DFG-Viewer, der in MEI codierte Notenmaterialien mit Hilfe von *Verovio* darstellt, würde eine solche Festlegung auf einen Standard geschaffen und eine stetige Verfügbarkeit der Darstellungskomponente sichergestellt.

[36] Zum Beispiel aus technischen oder auch rechtlichen Gründen.

[37] Vgl. Abrams, Stephen; Kunze, John; Loy, David: An Emergent Micro-Services Approach to Digital Curation Infrastructure. In*: International Journal of Digital Curation*, 5 (1) (2010) 172–86, DOI: http://dx.doi.org/10.2218/ijdc.v5i1.151.

[38] Vgl. http://iiif.io/.

und die Art der Änderung abschätzbaren Versionen[39] veröffentlicht werden und über eine stabile Referenz zugreifbar sein und bleiben. Die adaptierende Software muss explizit die Version[40] der einzusetzenden Bibliothek angeben, die für die korrekte Funktion notwendig ist. Ein Bruch der Abhängigkeit durch das Aktualisieren der adaptierten Software wird somit vorhersehbar – ggf. sogar durch ein Testing-Framework automatisch erkennbar.

Migration wird häufig nur im Kontext von Forschungsdaten gedacht und bezeichnet die Umstellung der Daten auf eine neue Struktur oder in ein neues Format,[41] aber auch Software kann migriert werden. Hierfür wird die Software der sich ändernden Umgebung angepasst, um die Funktionalität in einer aktuellen Infrastruktur, auf aktueller Hardware und mit aktuellem Betriebssystem und Laufzeitbibliotheken etc. aufrecht zu erhalten.[42] Für die Migration von Software ist ein recht detailliertes Wissen über die Entwicklung der Software notwendig, da u.U. die Software selbst angepasst werden muss. Dieser Aufwand ist aber für Software, die weit verbreitet ist und als Standard gelten kann, durchaus lohnend, da so z.B. keine Einbußen in der Performance hingenommen werden müssen.

Ein weiterer Ansatz für den Erhalt der Funktionalität einer Software ist die **Emulation**.[43] Hierfür wird die Umgebung einer Software insofern stabil gehalten, als dass die Hardware bzw. das gesamte System inklusive des Betriebssystems und aller notwendigen Softwarekomponenten virtualisiert wird.[44] Der Wartungsaufwand beschränkt sich dann lediglich auf die Weiterentwicklung und Anpassung des Emulators an die Gegebenheiten neuer Hostsysteme. Durch den Einsatz eines Emulators für den Erhalt vieler Visualisierungen von Forschungsergebnissen reduziert sich der Aufwand erheblich. Wie jeder Ansatz hat auch die Emulation Vor- und Nachteile; so sind emulierte Systeme sehr stark isoliert und nur schwer auf einer funktionalen Ebene in neue Umgebungen einzupassen. Wenn grundsätzlich eine modular aufgebaute Architektur vorliegt, kann die Emulation für z.B. einzelne *Micro Services* die richtige Strategie sein, da die Funktionalität per Definition bereits stark entkoppelt und über Schnittstellen definiert ist, andere – stärker frequentierte – Module können dann weiterhin migriert werden.

---

[39] Hierzu ist es ebenfalls ratsam, sich an einen Standard für die Versionierung von Software und Schnittstellen zu halten: *Semantic Versioning* (vgl. https://semver.org/).

[40] Wenn eine Software die Versionen nach den Prinzipien des *Semantic Versioning* angibt, kann auch eine Versionsspanne (z.B. 2.x.x) für eine adaptierte Software angegeben werden, da das *Semantic Versioning* sicherstellt, dass Änderungen an der Schnittstelle oder an bestehenden Funktionen mit einer neuen „Major"-Versionsnummer versehen werden müssen.

[41] Delve, Janet; Denard, Hugh; Kilbride, William: Digital Preservation Strategies for Visualisations and Simulations. In: *The Preservation of Complex Objects*, (1) (2012) 127.

[42] Hong et al. (Anm. 32) 67ff.

[43] Vgl. dazu Delve et al. (Anm. 41) 127 oder Hong et al. (Anm. 32) 66.

[44] Vgl. hierzu z.B. das Virtualisierungskonzept von Docker (https://www.docker.com/).

Die aktive Weiterentwicklung einer Software oder eines Datenformats z.B. durch eine **Open-Source-Community** ist sicherlich in vielen Fällen erstrebenswert und u.U. für Standardsoftware oder einen Datenstandard realisierbar, aber für viele Visualisierungen und Erweiterungen von Datenformaten aufgrund der stark auf ein Projekt bezogenen spezifischen Ausrichtung und somit mangelnden Größe oder gar Existenz einer Community undenkbar. Diese im „Software Preservation Benefits Framework" von Hong et al. *Cultivation*[45] genannte Strategie ist z.B. im Kontext digitaler Musikeditionen für den Codierungsstandard der *Music Encoding Initiative* (MEI) gelebte Praxis, die Notenanzeigebibliothek *Verovio* hingegen erhält aufgrund ihrer Komplexität nur wenig Entwicklungsleistung aus der Community und die Anzeigesoftware für digitale Musikeditionen *Edirom Online* hat aufgrund der sehr spezifischen Ausrichtung nur eine so kleine Community, die darüber hinaus über wenig Entwicklungs-Know-How verfügt, dass hier eine Community-basierte Weiterentwicklung nicht realisierbar ist. Weiterentwicklungen werden hier in aller Regel in Forschungsprojekten durch das Zentrum Musik – Edition – Medien koordiniert durchgeführt.

4 Langzeitverfügbarkeit gemeinsam angehen

Forschungsprojekte können demnach die Langzeitverfügbarkeit ihrer Ergebnisse insofern unterstützen, als dass sie möglichst Standards für Daten und Visualisierungen nutzen und sich generell um eine Standardbildung bemühen. Aber auch die explizite Dokumentation der genutzten Formate und Software und die Veröffentlichung der eigenen Ergebnisse mithilfe von Versionen und persistenten Identifikatoren tragen dazu bei. Softwareentwicklung, entweder in Forschungs- oder Infrastrukturprojekten, kann durch Modularisierung und die Reduktion von Abhängigkeiten, standardisierte Beschreibungen der Software und der Schnittstellen und Verbreitung des Wissens über die Software im Sinne der *Cultivation* die Chancen für eine lange Verfügbarkeit der Forschungsergebnisse erhöhen. Gedächtnisinstitutionen können die Standardisierung aktiv mitgestalten, Rahmenbedingungen für z.B. Emulation vorgeben und Strukturen aufbauen, die so modular sind, dass sie standardisierte Formate und Visualisierungen ermöglichen, ohne dadurch die projektspezifischen Lösungen auszuschließen.

Es ist offensichtlich, dass die Herausforderung der Langzeitverfügbarkeit von Forschungsergebnissen eine gemeinsame Aufgabe der Forschung, Softwareentwicklung und Gedächtnisinstitutionen sein muss, wobei diese förderpolitische und strategische Unterstützung durch Drittmittelförderer aber auch durch eine projektunabhängige „Grundfinanzierung durch entsprechend engagierte Bundes- und Landesministerien", wie es Gietz und Hütter in ihrem Aufsatz zum Aufbau der DARIAH-DE

[45] Hong et al. (Anm. 32) 70f.

eHumanities Infrastructure Service Unit gefordert haben,[46] bei der Zusammenstellung dieser Allianz z.B. in Form dauerhafter Stellen bekommen müssen.

Daniel Röwenstrunk

Zentrum Musik – Edition – Medien

Musikwissenschaftliches Seminar Detmold/Paderborn

Hornsche Straße 39

32756 Detmold

daniel.roewenstrunk@uni-paderborn.de

[46] Gietz, Peter; Hütter, Heiko: Der Aufbau einer nachhaltigen technischen Infrastruktur für die Geisteswissenschaften: Die DARIAH-DE eHumanities Infrastructure Service Unit (DeISU). In: *BIBLIOTHEK – Forschung und Praxis*, 40 (2) (2016) 248.