NACHHALTIGE SOFTWAREENTWICKLUNG: CODE4RESEARCH

von Raman Ganguly

Zusammenfassung: *Mit dem Schritt ins digitale Zeitalter werden in Forschungsprozessen zunehmend Daten in digitaler Form verarbeitet, die mit Hilfe von Software erzeugt, analysiert und dargestellt werden. Das Angebot an kommerziellen Lösungen stößt bei den Anforderungen aus der Forschung an ihre Grenzen und so wird immer öfters im Rahmen eines Forschungsprojekts Software entwickelt. In vielen Fällen sollen die entwickelten Programme nach Projektende weiter betrieben werden, da sie als Grundlage für weitere Forschung dienen bzw. werden die Ergebnisse in alternativer Form publiziert. Damit werden viele Wissenschaftlerinnen und Wissenschaftler vor neue Herausforderungen gestellt, die mit ihrer eigentlichen Aufgabe, dem Forschen, nur am Rande etwas zu tun haben. Softwareentwicklung ist nur eine von vielen weiteren Herausforderungen, die mit der zunehmenden Digitalisierung einhergehen. Dieser Artikel fokussiert sich auf den Entwicklungsprozess von Software unter Berücksichtigung der Nachhaltigkeit und den Aspekt der Archivierung. Er stellt unter anderem auch die Frage, ob und wie die Forschung von zentraler Stelle unterstützt werden kann und welche Rolle das Forschungsdatenmanagement dabei einnimmt.*

Schlüsselwörter: *Softwareentwicklung; Forschungsdatenmanagement; Archivierung; Nachhaltige Software*

SUSTAINABLE SOFTWARE DEVELOPMENT: CODE4RESEARCH

Abstract: *With the step into the digital age, research processes increasingly process data in digital form, which are generated, analyzed and presented with the help of software. The range of commercial solutions is reaching its limits when it comes to research requirements, and software is increasingly being developed as part of a research project. In many cases, the developed programs will continue to run after the project ends as they serve as a basis for further research or the results will be published in an alternative form. This poses new challenges for many scientists, who have little to do with their actual task, research. Software development is just one of many other challenges that come with increasing digitization. This article focuses on the development process of software taking into account sustainability and the aspect of archiving. Among other things, it raises the question of whether and how research can be supported centrally and what role research data management plays in this.*

Keywords: *software development; research data management; archiving; sustainable software*

DOI: *https://doi.org/10.31263/voebm.v71i1.2019*

Inhalt

1. Einleitung

Im Zeitalter der digitalen Revolution und am Ende der Gutenberg Galaxis wird von Daten als das neue Öl[1] gesprochen. Um dieses Öl zu fördern und nutzbar zu machen, ist der Einsatz von Software notwendig. Software ist jenes Werkzeug, das den Daten Sinn gibt. Auch in Forschungsprozessen nimmt nicht nur die Bedeutung der Daten, sondern auch die der dazugehörigen Software, zu.

Die Anforderungen an Software im Rahmen von Forschungsprozessen sind komplex und nur wenig bis gar nicht standardisiert. Der Softwaremarkt orientiert sich am Bedarf der Märkte und entwickelt standardisierte Produkte, die den Bedürfnissen den Forschung und der Forschenden oftmals nicht genügen. So ist es immer häufiger notwendig, dass im Rahmen von wissenschaftlichen Projekten Software entwickelt wird. Je nach Anforderung hat das entwickelte Produkt über die eigentliche Projektdauer hinweg mehr oder weniger Bedeutung.

Dieser Artikel widmet sich dem Prozess der Softwareentwicklung und der Herausforderung für die Forschung, sowie den Aspekten, was nach Ende des Projekts mit der Software geschehen soll.

2. Software und ihr Umfeld

Ganz allgemein ist mit Software eine Vielzahl von unterschiedlichen Computerprogrammen gemeint, die ein IT-System zum Laufen bringt. Es können die Bereiche Hardware, Betriebssystem, Anwendung und User unter-

schieden werden, die jeweils mit spezialisierten Programmen bedient werden. Diese Teile greifen ineinander und haben Abhängigkeiten zueinander. Auch Hardware und User sind aus Sicht der IT ein integraler Bestandteil des gesamten Systems, da es sich um Schnittstellen zur physischen Welt handelt. Es gibt spezialisierte Software, die für die Interaktion mit der Hardware (z.B. Driver) oder mit den Usern (User Interface) geschrieben ist.

Im Fokus steht die sogenannte Anwendungssoftware, die eine spezielle Nutzung der Daten gewährleistet. Dies kann sein, dass Daten in einer bestimmten Form abgelegt und strukturiert werden, dass diese in Beziehung gesetzt werden oder mit Hilfe von Algorithmen analysiert werden. Die Daten können auch grafisch aufbereitet und dargestellt oder mit weiteren Daten aus anderen Quellen (also anderen Anwendungen) in Beziehung gesetzt werden, um so neue Erkenntnisse zu gewinnen. Auch können Daten mit Hilfe von mathematischen Modellen in eine virtuelle Realität versetzt werden, um mit Hilfe von Simulationen diverse Vorhersagen für eine mögliche Zukunft treffen zu können. Die Möglichkeiten sind sehr vielfältig und der Computer erlaubt nicht nur neue Blickwinkel, sondern schafft es eine hohe Anzahl von Daten in kurzer Zeit zu analysieren, die manuell nicht zu bewältigen wäre.

Limitierende Faktoren der Möglichkeiten sind Aufwand und Kenntnisse, um Software zu entwickeln. Das notwendige Wissen entsteht aus einem Zusammenspiel der IT und der fachlichen Domäne. Die fachliche Domäne liefert das Wissen über den Prozess, was mit den Daten geschehen soll, und die IT das Handwerkszeug, um diesen in eine für den Computer verständliche Form zu bringen. Um erfolgreich zu sein, braucht es von beiden Seiten ein hohes Maß an Verständnis für das Gegenüber. Je besser sich beide Seiten verstehen, umso effizienter und effektiver kann Software erstellt werden.

3. Aspekte für die Archivierung von Software

Was soll nun nach dem Projekt mit der Software passieren? Um diese Frage beantworten zu können, geht man zurück zum Beginn und schaut sich die Zielsetzung an, warum die betreffende Software überhaupt geschrieben wurde. Es können sich daraus drei Möglichkeiten ergeben:

Die Software diente zur Erhebung, Analyse, Auswertung oder Darstellung von bestimmten Daten, die nur einmal benötigt wurden und sie hat nach dem Projekt keinen weiteren Nutzen mehr. Die Software selbst dient nicht der Reproduzierbarkeit des Forschungsergebnisses und hat sonst auch keinen dokumentarischen Zweck.

Sie hatten einen ähnlichen Zweck wie im ersten Fall, nur mit dem Unterschied, dass sie für die Dokumentation oder die Nachvollziehbarkeit der Ergebnisse von Bedeutung ist.

Sie stellt die Forschungsergebnisse dar beziehungsweise ist die Grundlage für weitere Forschungen.

Im ersten Fall ist es sehr einfach zu entscheiden, was getan werden muss. Die Software kann außer Betrieb genommen und gelöscht werden. Es muss nichts weiter berücksichtigt, sondern lediglich vielleicht ein Vermerk in der Dokumentation gemacht werden.

Auch im zweiten Fall kann die Software außer Betrieb genommen werden. Der Unterschied besteht allerdings darin, dass die Software nicht gelöscht, sondern der Source Code, also die textuelle Repräsentation der Software in einem geeigneten Archivsystem abgelegt wird, damit sie zitiert werden kann. Darüber hinaus muss das Umfeld berücksichtigt werden: Welche Dokumentation ist notwendig, um das Programm wieder in Betrieb zu setzen und um den Code zu verstehen? Alle wesentlichen Abhängigkeiten müssen sich in dieser wiederfinden.

Der dritte Fall stellt ein großes Problem dar, da die Software nicht außer Betrieb genommen werden kann. Dies bedeutet in mehrerlei Hinsicht das Betreiben eines gewissen Aufwands: Zum einen in technischer Hinsicht, da sich das Umfeld, also das Betriebssystem usw. immer wieder ändert, weil regelmäßig neue Versionen eingespielt werden. Diese Änderungen können dazu führen, dass die Software nicht mehr lauffähig ist. Zum anderen ergibt sich ein nicht technischer Aufwand, der durch die AnwenderInnen entsteht. Im Lauf der Nutzung von Software entwickeln sich Fragen und Wünsche oder es werden Fehler gefunden. Der Betrieb muss auf diese Dinge Rücksicht nehmen, da die Software mit der Zeit sonst nicht mehr genutzt wird.

4. Nachhaltige Softwareentwicklung

In weiterer Folge betrachten wir den langfristigen Betrieb von Software. Der Aufwand im Betrieb wird reduziert, wenn bei der Planung und Entwicklung bereits auf Nachhaltigkeit geachtet wurde.

Eines der wichtigsten Dinge bei der Entwicklung von nachhaltiger Software ist die Dokumentation. Es kann niemals alles dokumentiert werden, und jede Dokumentation ist lückenhaft. Die Dokumentation soll einem Zweck folgen; in diesem Fall jenen, einen möglichst langfristigen Betrieb zu gewährleisten. Die wichtigsten Fragen, die es zu beantworten gilt, sind dabei:

- Was ist der Zweck der Software?
- Wie wird die Software gebaut? (Architektur und Programmiersprache)
- Warum wurde die Architektur gewählt?
- Warum wurde diese Sprache gewählt?
- Welche Abhängigkeiten gibt es?
- Welche Softwarekomponenten werden benötigt?
- Welche Voraussetzungen sind notwendig, um die Software in Betrieb zu halten?
- Was wurde umgesetzt?
- Was war geplant und wurde nicht umgesetzt?
- Was sind kritische Punkte im Betrieb?
- Wie werden die Daten gespeichert?

Die Punkte der Dokumentation können auch so betrachtet werden, dass sie einen Leitfaden für die Umsetzung darstellen. Alles was dokumentiert werden muss, sollte im Vorfeld auch geplant worden sein bzw. bewusst entschieden werden.

Für die Umsetzung werden drei große Blöcke unterschieden: Planung, Umsetzung und Betrieb. In der Planung muss geklärt werden, warum, was und wie umgesetzt werden soll. Bei der Umsetzung ist die sogenannte Architektur wichtig. Diese sollte so umgesetzt werden, dass ein späterer Betrieb möglichst einfach gestaltet werden kann. Der letzte Punkt ist der Betrieb selbst, der aus Sicht der Softwareentwicklung den Abschluss bildet. Im besten Fall gibt es eine offizielle und dokumentierte Übergabe an den Betrieb. Hier zeigt sich in der Praxis ofters ein Problem, da am Beginn nicht definiert wurde wer den Betrieb leistet. An dieser Stelle tritt plötzlich die Frage auf: wer ist das für verantwortlich?

4.1. Planung

Bei der Zielsetzung ist die zentrale Frage das Warum. Softwareentwicklung bedeutet einen erheblichen Aufwand und bindet viele Ressourcen. Daher ist es wichtig, im Vorfeld genau zu definieren, warum eine neue Software gebraucht wird und warum keine bereits bestehende Software verwendet werden kann. Im Rahmen eines Forschungsprojekts ist diese Frage sicher rasch zu beantworten, dennoch sollte sie gestellt werden. Eine gute und nachvollziehbare Antwort hilft bei der Beschaffung der notwendigen Mittel. Ist das Warum geklärt, ist es notwendig das Was zu eruieren. An diesem Punkt kommt das sogenannte Requirements Engineering[2] ins Spiel, auf das in diesem Artikel nicht weiter eingegangen wird.

Die Praxis zeigt, dass am Beginn der Überlegungen oft das Was nicht richtig geklärt werden kann. Methoden, wie die agile Softwareentwicklung, begegnen diesem Problem, indem sie die Entwicklung in mehrere kleine Schritte teilen und nach jedem Schritt prüfen, ob der eingeschlagene Weg der richtige ist. Daher ist es wichtig, sich ein Bild vom Endprodukt zu machen, auch wenn es noch verschwommen und in seiner Gesamtheit nicht völlig erfasst ist, das Ziel muss bekannt sein. Auch bei einem agilen Ansatz ist ein gewisses Maß an vorangestellter Planung notwendig. Geht man vom Warum zum Was, ist es leichter ein grobes Bild zu zeichnen, da die zukünftigen Ergebnisse (Was) auf einem Fundament (Warum) aufsetzen.

Beim letzten Schritt der Planung steht schon die Umsetzung im Fokus. Mit dem Wie werden grundlegende Paradigmen festgelegt und Entscheidungen für die eigentliche Entwicklung getroffen: an diesem Punkt wird über Architektur und Spracheentschieden. Auch dabei wird in der Praxis oft nicht das Was berücksichtigt und die Entscheidung des Wie auf Grund von subjektiven und ungeprüften Vorlieben getroffen.

4.2. Umsetzung

Die Umsetzung selbst ist ein rein technischer Akt. Die Größe, Zeit und Ressourcen bestimmen, wie viele EntwicklerInnen an der Software arbeiten. Pragmatismus ist hier gefragt. Der Weg zu einer funktionierenden Software ist nicht einfach. Immer wieder stößt man auf Dinge, die nicht einkalkuliert worden sind. Das liegt nicht an einer schlechten Planung, sondern ist bedingt durch die Komplexität der Materie. Eine laufende Beobachtung des Prozesses ist wichtig, um Irrwege frühzeitig zu erkennen und aus den Fehlern zu lernen. Rasche Entscheidungen zur Korrektur sind notwendig, die von den EntwicklerInnen oft alleine getroffen werden können. Kurze Kommunikationswege zwischen Entwicklung und fachlichen Ansprechpersonen sind ein wesentlicher Erfolgsfaktor. Je enger beide Bereiche zusammenarbeiten, desto besser funktioniert es. Der Weg zu einer guten Software führt über viele Fehler – das Motto ist: „Fail fast, fail often."

Um den nachhaltigen Betrieb zu erleichtern, gilt: Trennung von Daten und Logik. Das erleichtert die Wartung, hilft den Überblick zu bewahren und Abhängigkeiten so gering wie möglich zu halten sowie autonome Sinneinheiten zu bauen. Der Software-Code ist an sich schon ein Teil der Dokumentation, wenn er sauber und konsistent geschrieben wurde. Die Verwendung von Versionskontrolle unterstützt den Dokumentationscharakter gut; natürlich nur dann, wenn die Kommentare schlüssig und sinnvoll sind.

4.3. Betrieb

Ist eine Software fertig, wird sie in den Betrieb überführt. Dieser umfasst mehrere Aspekte, um das eigentliche Ziel zu erreichen: Die Software soll von einer definierten Anzahl von BenutzerInnen[3] verwendet werden, die grundsätzlich sehr wesentlich ist, denn je größer diese ist bzw. je schlechter diese Gruppe zu erreichen ist, desto kritischer ist die Einrichtung eines sogenannten Helpdesks. Zielgruppendefinitionen helfen bei der Bestimmung dieser Gruppen. Auch bei Web-Anwendungen, die theoretisch von jedem Menschen mit Zugang zum Internet benutzt werden können, wird dies nie der Fall sein. Je besser die Zielgruppe definiert werden kann, um so leichter ist es den Helpdesk zu automatisieren[4].

Die NutzerInnen einer Software stellen über kurz oder lang Fragen, und dessen muss sich der Betrieb bewusst sein. Will man nun auf diese bestmöglich eingehen, so ist es notwendig, ein Konzept für die Kommunikation zu entwickeln. Dabei ist es von Vorteil, wenigstens eine minimale Kommunikation mit den NutzerInnen oder zumindest ein Monitoring über die Nutzung zu führen, damit erkannt werden kann, ob es Sinn macht den Betrieb aufrecht zu erhalten. Eine Software, die nicht mehr benutzt wird, kostet unnötig Ressourcen, wenn diese weiterhin am Laufen gehalten wird.

Über eine geregelte Kommunikation erhält man auch Informationen über Fehler der Software. Eine Software ist nie fehlerlos und erst in der Benützung werden viele Mängel sichtbar. Auch gut getestete Anwendungen können viele Fehler nicht erkennen, da sie sich erst bei einem längeren Betrieb zeigen.

Aktive Kommunikation mit den NutzerInnen ist nicht nur als Aufwand zu sehen, sondern vor allem als Erfolgsfaktor für die Software. Eine nachhaltige Software wird stetig weiterentwickelt. Nur eine Software mit einer aktiven NutzerInnengruppe kann die notwendigen Mittel generieren, um eine Weiterentwicklung zu finanzieren.

4.4. Zusätzliche ExpertInnen

Kommt es zu der Anforderung, eine Software zu entwickeln und auch nach Abschluss des Projekts nachhaltig zu betreiben, sind zusätzliche ExpertInnen notwendig, die vorrangig nichts mit dem eigentlichen Forschungsprozess zu tun haben oder hatten.

Die offensichtlichsten Fachleute, die miteinzubeziehen sind, sind TechnikerInnen für Entwicklung und Betrieb. Letzterer hat ebenfalls eine starke Kommunikationskomponente, die zusätzlich geplant werden sollte, um

erfolgreich zu sein[5]. Die Planung und Umsetzung von Software läuft parallel zum eigentlichen Forschungsprozess: Die Software muss zum richtigen Zeitpunkt fertig gestellt und die Kosten, die dafür eingeplant wurden, dürfen nicht überschritten werden. Auf Grund der schlechten Planbarkeit der Softwareentwicklung muss auf Kosten und Zeit geachtet werden, damit am Ende nicht halbfertige Software entsteht, die niemand verwenden kann.

Soll die Software am Ende klassisch archiviert werden, also nicht laufen, sondern als zusätzliche Forschungsdaten dienen, dürfen die Kosten für die Entwicklung trotzdem nicht vergessen werden. In Zusammenhang mit der aktuellen Forschungsförderung wird auf diese Problematik nur zu einem geringen Maße eingegangen und es gibt oft keine ausreichende Finanzierung für die Entwicklung von Software. Darüber hinaus sollte sich die zu entwickelnde Software auch in einem guten Datenmanagementplan wiederfinden.

Selbst dann, wenn die Entwicklungskosten finanziert sind, ist mitunter noch nicht geklärt, wie Forschungsprojekte geeignete SoftwareentwicklerInnen finden. Der Arbeitsmarkt für diese Personen ist von der privatwirtschaftlichen Seite hart umkämpft, da auch dort eine hohe Nachfrage nach guten EntwicklerInnen herrscht. Diese Situation bedingt, dass diese Fachleute meist rar und deshalb besonders teuer sind. Vor allem sind EntwicklerInnen nicht daran gewöhnt, nur für ein einzelnes Projekt angestellt zu werden. Sie arbeiten meist in IT-Abteilungen, die ihnen ein personelles Umfeld für den fachlichen Austausch bietet.

5. Gruppe code4reseach

An der Universität Wien hat sich 2017 eine Gruppe von ForscherInnen, BibliothekarInnen und IT-ExpertInnen gebildet, um Lösungen für diese oben beschriebenen Probleme zu finden. Die Gruppe „code4research“[6] will eine Plattform bilden, um die Anforderungen unterschiedlicher Forschungsgruppen zu erheben und pragmatische Lösungen für die dringendsten Probleme zu finden.

Die Gruppe an der Universität Wien dient auch zum Austausch von BestPractice-Methoden, um sich so gegenseitig zu unterstützen. Ein drittes Ziel ist es, breitenwirksam auf diese Probleme aufmerksam zu machen, damit nachhaltige Lösungen gefunden und finanziert werden können.

Die Finanzierung aus Mitteln, die indirekt aus Forschungsprojekten kommen, ist notwendig. Am besten sollte nur ein zentrales Service für eine Institution aufgebaut werden, da so Synergien geschaffen werden können,

und somit kostengünstiger gearbeitet werden kann. Dieses Service kann als kooperatives Service aus unterschiedlichen Dienstleistungseinrichtungen der Institution gestaltet werden, wie zum Beispiel als Kooperation zwischen Zentraler IT und Bibliothek. Es kann auch an den Fakultäten angesiedelt sein, mit einer zentralen Koordinationsstelle. Dies kann nur im Rahmen einer Organisationsentwicklung einer Institution gestaltet werden.

Zurzeit gibt es nur sehr wenige Beispiele wie Universitäten und Institutionen sich mit dem Problem von Nachhaltigkeit Softwareentwicklung in der Forschung begegnen. Meist wird dieses Thema noch den ForscherInnen überlassen. Positiv kann das UCL (University College London) genannt werden. Hier gibt es im Rahmen der Zentralen IT eine Abteilung die sich mit den IT-Bedürfnissen der Forschung auseinandersetzt. Diese Abteilung bietet auch ein Service für Softwareentwicklung an[7].

6. Conclusio

Um die Entwicklung von Software effizient zu gestalten, braucht es eine enge Zusammenarbeit zwischen zentralen Services und domainspezifischen Expertisen. Zentraler Support könnte vom Aufbau von Infrastrukturen zur besseren Unterstützung der Softwareentwicklung in der Forschung bis hin zur Zurverfügungstellung eines Pools von IT-ExpertInnen zur Unterstützung der Forschung reichen. In den Niederlanden wurden solche Services bereits auf nationaler Ebene eingerichtet[8].

Auch der Betrieb macht nur aus zentraler Sicht Sinn: Ob eine Person eine oder mehrere Softwareprodukte betreibt ist egal. Erst ab einem gewissen Schwellenwert müssen zusätzliche Ressourcen aufgebracht werden. Sind Betrieb und Entwicklung nahe beisammen, können auch die Fragen der Architektur bei der Entwicklungsplanung besser berücksichtigt werden.

Mittelfristig ist die Schaffung einer zentralen IT, die den Forschungsprozess begleitet, eine kostengünstigere Variante, als jedes Projekt einzeln mit Ressourcen auszustatten. Darüber hinaus können Synergien mit der Lehre geschaffen werden, indem man die Forschungs-IT auch für die forschungsgestützte Lehre nutzt. Weiteres bietet sich die Möglichkeit, so auch Aus- und Weiterbildungsprogramme in Zusammenarbeit mit dieser zentralen IT zu gestalten und anzubieten.

Forschung und Lehre einer Universität lassen sich über zentrale Services besser im Web darstellen. Die in beiden Bereichen generierten Daten können in gleicher Weise behandelt werden und sind so langfristig und nachhaltig verwendbar.

Softwareentwicklung wird in den nächsten Jahren immer mehr an Bedeutung gewinnen, und ein professioneller Umgang mit dem Thema wird zukünftig sowohl für die Forschung als auch die Lehre ein Erfolgsfaktor sein.

Dipl.-Ing. (FH) Raman Ganguly
ORCID: http://orcid.org/0000-0002-9837-0047
Universität Wien, Zentraler Informatikdienst
E-Mail: raman.ganguly@univie.ac.at

1 Öl gilt als der wichtigste Rohstoff der sogenannten old economy, also den klassischen Wirtschaftsbetrieben. Es wird die Metapher für Daten herangezogen, da Daten genauso wichtig für die new economy sind, wie Öl für die old economy.
2 Vgl. auch Susanne Margareta Blumesberger (2017): Der Umgang mit Requirements-Engineering an wissenschaftlichen Bibliotheken. Master-Thesis (ULG), Universität Wien. Universitätslehrgang Library and Information Studies (MSc). URN: urn:nbn:at:at-ubw:1-12645.74671.225161-2
3 Entspricht der Zielgruppe. Es kann natürlich sein, dass die eigentliche Zahl der BenutzerInnen größer oder kleiner dieser Zielgruppe ist. Für die Planung des Betriebs geht man von der Zielgruppe aus, da im Idealfall die Software genau auf diese Gruppe zugeschnitten wird.
4 Die einfachste Form eines automatisierten Helpdesks ist es, Hilfstexte zu implementieren und Anleitungen zur Verfügung zu stellen.
5 Erfolgreich im Sinne der Zielsetzung, also das die Ziele bestmöglich erreicht werden.
6 Webseite von code4research: https://datamanagement.univie.ac.at/rdm/netzwerk-nachhaltige-softwareentwicklung/
7 Vgl. Webseite UCL: Information Service Division/Reserach IT: https://www.ucl.ac.uk/isd/services/research-it/research-software-development
8 Vgl. Angebote von Netherlands eScience Center: https://www.esciencecenter.nl/ – SURFsara: https://www.surf.nl/en/about-surf/subsidiaries/surfsara/ – DANS: https://dans.knaw.nl/en