Computersoftware als digitales Erbe: Probleme aus Sicht der Technikgeschichte

Michael Friedewald, Timo Leimbach

1 Ein dunkles digitales Zeitalter?

Unser Zeitalter hat so manchen klingenden Namen bekommen: Informations-, Kommunikations- oder Wissenszeitalter. Ganz falsch ist „Informationszeitalter“ sicher nicht, versteht man unter Information nicht Informiertheit und zählt alles dazu, was an Schriften, Bildern, Klängen hervorgebracht wird, alle Äußerungen menschlichen Geistes und menschlicher Geistlosigkeit. Aber wenn wir mitten in einer beispiellosen Informationsexplosion leben, so gleichzeitig in einer Zeit des beispiellosen Informationsverfalls. Bereits 1997 stellte der Informationswissenschaftler Terry Kuny die Frage, ob wir nicht in einem neuen dunklen, digitalen Zeitalter leben, das künftigen Generationen nicht viel an verwertbaren historischen Quellen hinterlässt (Kuny 1998).

Dies gilt für alle Arten von digitalen Dokumenten, egal ob diese durch die Digitalisierung analoger Originale oder aber um von vorneherein digital ("digital born") entstanden sind. Bei letzteren denkt man normalerweise an Texte, die am Computer geschrieben wurden, an digitale Fotos oder an Webseiten im Internet. Daneben gibt es aber auch Texte bzw. Dokumente, die im ureigensten Sinne digital entstanden sind, nämlich die Programme bzw. Software. Erst durch sie wird aus dem Digitalcomputer ein nützliches und universell einsetzbares technisches Gerät; dies ist wiederum die Voraussetzung für die beispiellose Erfolgsgeschichte des Computers in praktisch allen Bereichen des gesellschaftlichen Lebens. Software ist deshalb seit einigen Jahren verstärkt Gegenstand der historischen Forschung (z.B. Broy et al. 2002; Hashagen et al. 2002; Campbell-Kelly 2003).

Bislang spielt Software bei den vielfältigen Aktivitäten zur Sicherung des kulturellen Erbes noch keine Rolle. Das seit 1992 laufende UNESCO-Programm „Gedächtnis der Menschheit“ geht zwar Fragen des vielfältigen Dokumentenerbes nach, in dem allerdings Software als besondere Form digital entstandener Dokumente bislang keine Rolle spielt. Dabei besteht in diesem Nischenbereich bereits heute höchst konkret die Gefahr, dass die Quellen aus mehreren Jahrzehnten der Technikgeschichte unwiederbringlich verloren gehen.

Im folgenden wollen wir kurz skizzieren, (1) was Computer und Software zu Artefakt machten, die mit den traditionellen Methoden der Geschichtswissenschaft nur unzureichend adressiert werden kann, (2) was die Ursachen für die Gefahr des Verlusts der Quellen darstellt und (3) wie man diesen Gefahren zu begegnen versucht. Wenngleich sich unsere Ausführungen auf Computersoftware als eine spezielle Form digitaler Dokumente konzentrieren, gelten die meisten Erkenntnisse cum grano salis auch für andere digitale „Texte".

2 Computer und Software als gegenständliche Quellen in der Technikgeschichte

Unter gegenständlichen Quellen versteht man „unmittelbar überlieferte Gegenstände bzw. Überreste, aus denen historische Informationen über die Zeit ihrer Entstehung und Verwendung gewonnen werden können" (Stadtmu□ller 1999). In der Technikgeschichte haben solche Quellen einen hohen Stellenwert, da sie einem unmittelbaren Eindruck über die Funktionsweise, die verwendeten Materialien, aber auch Produktionsverfahren und ästhetische Vorstellungen geben (Troitzsch et al. 1980). Darüber hinaus sind sie oft die einzigen erhaltenen Quellen, da technische Dokumente lange Zeit nicht als historisch nicht bedeutsam erachtet und weggeworfen wurden oder im Zuge von Unternehmensauflösungen verloren gingen.

Im Prinzip handelt es sich bei den Sachquellen um eine Quelle wie jede andere – auch sie unterliegen daher den Bedingungen der Quellenkritik und Quelleninterpretation. Stahlschmidt (1977) resümiert deshalb: „Wertvolle Informationen kann die eigene Beobachtung und Untersuchung von gegenständlichen Quellen … liefern, sofern Herkunft und Datierung einwandfrei belegt sein" und formuliert damit bereits einige der Schwierigkeiten des Computers als gegenständlicher Quelle: Im Gegensatz zu mechanischen Artefakten entzieht sich der Computer und in noch stärkerem Maße die Software der Beobachtung durch den Historiker. Darüber hinaus sind Belege für die Datierung und zum Teil auch für die Herkunft häufig nicht zu ermitteln.

Beobachtbarkeit: Technische Artefakte aus der jüngeren Vergangenheit und besonders elektronische Geräte sind trotz ihres häufig guten Erhaltungszustands wenig zugänglich , weil ihre Funktionsweise im Gegensatz zu den meisten mechanischen oder elektromechanischen Geräten nicht mehr unmittelbar sichtbar ist. Die Funktionsweise einer elektronischen Schaltung erschließt sich häufig einfacher aus dem Stromlaufplan als durch Untersuchung der realen Schaltung. Dabei weisen Computers eine weitere

Besonderheit auf, die darin besteht, dass man über die technische Funktionsweise und den logischen Aufbau selbst bei Vorhandensein eines noch funktionstüchtigen Gerätes und entsprechender Software keine Aussage treffen kann. Ursache ist die Tatsache, dass die Verwendung elektronischer Bauteile und deren fortschreitenden Miniaturisierung den Computer zu einer Art „Black Box“ werden lassen. Damit ist uns das Design, in dem sich das technische Wissen und die praktischen Anforderungen widerspiegeln, nicht mehr zugänglich. Folglich verlieren Computer viel von ihrem Quellenwert als technisches Artefakt.

Authentizität: Gegenständliche Quellen weisen zwar per se eine hohe Authentizität auf, sind in der Regel aus ihrem ursprünglichen Funktionszusammenhang entfernt, so dass dieser erst mühsam rekonstruiert werden muss. Im Fall des Computers betrifft dies nicht nur die Vielzahl an peripheren technischen Einrichtungen die zu ihrem Betrieb nötig waren (z.B. Klimatisierung), sondern auch die Einbettung in organisatorische Zusammenhänge (Betrieb in Rechenzentren vs. persönliche Computer). Darüber hinaus Darüber hinaus handelt es sich bei Software um ein nicht-materielles technisches Artefakt, das auf materiellen Datenträgern gespeichert ist und nur auf einer bestimmten, ebenfalls materiellen Hardware läuft. Dies erschwert die Festlegung, was nun das relevante, für eine bestimmte historische Situation authentische Artefakt eigentlich darstellt.

Software als hybride Artefakte: Vor allem in Deutschland wird seit wenigstens drei Jahrzehnten in der Informatik-Community darüber diskutiert, ob die Informatik eine Ingenieurwissenschaft, eine Formalwissenschaft der Information oder gar eine Kulturtechnik ist (Eulenhöfer 1999; Coy 2004). Ähnliche Fragen stellen sich auch Kuratoren, wenn Sie ihre Aufgabe vorrangig darin sehen, die *materielle Objekte* zu sammeln und zu konservieren. So kann man denn zu der Auffassung gelangen, dass es ausreicht, sich auf die materielle Datenträger zu konzentrieren, während die darauf gespeicherten Informationen eher abstrakte *Ideen* und damit nicht-technisch seien (Swade 2002). Grundlegende Algorithmen (z.B. für das Sortieren) sind zwar abstrakte Handlungsvorschriften zur Lösung eines Problems, die prinzipiell auch ohne eine technische Realisierung dokumentiert werden können. Typische Computerprogramme haben aber durchaus einen technischen Charakter, weil sie in der Regel konkrete technische Probleme adressieren und diese unter Zuhilfenahme von technischen Mitteln zur Informationsverarbeitung lösen. Diese Charakterisierung legt zwar nahe, dass Software nach systematischen ingenieurwissenschaftlichen Methoden entsteht, doch ähnlich wie in anderen Bereichen spielen aber auch Kreativität und Ästhetik eine

wichtige Rolle bei der Softwareerstellung. Am deutlichsten wird dies vielleicht bei Computerspielen, bei denen die visuellen und narrativen, mithin künstlerischen Elemente entscheidend sind.[1] Software ist also stärker als andere industrielle Produkte ein hybrides kognitiv-technisches Artefakt, die herkömmliche Prozesse der informatischen Produktion als digitale Kopie replizieren und diese mittels der digitalen Grundstruktur zu neuen Wissenstechniken zusammenführen. Dies impliziert, dass es sich bei der Software um das entscheidende Element einer Informationsgesellschaft handelt, dass entsprechend schützenswert ist. Gleichzeitig wird deutlich, dass sich Software nur mit einem multiperspektivischen und transdisziplinären Ansatz historisch analysieren und bewerten lässt (Shustek 2006; Mahoney 2008).

Die Technikgeschichtsschreibung hat verschiedene Ansätze gefunden, mit diesen Problemen umzugehen. So hat der Mathematiker Richard Hamming bereits 1976 gemahnt, dass Historiker über die Nutzung vorhandener schriftlicher und gegenständlicher Quellen hinausgehen sollten, da diese die wissenschaftliche Praxis und Kultur nur unzureichend dokumentierten (Hamming 1980). Eine Reaktion auf diese Erkenntnis war die verstärkte Nutzung der „Oral History“, also der Befragung von Zeitzeugen. Trotz gewisser methodischer Probleme (Geppert 1994) hat dies in den vergangenen Jahren eine ganze Reihe beeindruckender Studien zur Computer- und Softwaregeschichte hervorgebracht (z.B. Norberg et al. 1996).

3 Probleme bei der Erhaltung digital entstandener Texte

Die geschilderten Charakteristika von Software sind ein Grund, warum bislang unklar ist, ob es sich dabei überhaupt um kulturelles Erbe handelt und wer für dessen Sicherung verantwortlich sein sollte.[2] Darüber hinaus hat die Vergänglichkeit des materiellen Trägers auch Konsequenzen für die Verfügbarkeit bzw. langfristige Bewahrung der Software als dem eigentlichen digitalen Text. Neben der altbekannten Frage nach der Lebensdauer des Datenträgers (vgl. die Thematik des Papierzerfalls bei Büchern) werden

[1] Diese künstlerisch-visuelle Seite des Entwurfs technischer Artefakte ist freilich nicht auf Computer und Software beschränkt. Eugene Ferguson hat darauf hingewiesen, dass die nicht-verbalen und intuitiven Elemente des Design ein wichtiges Element technischen Handels ist, das insbesondere in den USA ein wichtiges Element der Technikkultur darstellt (Ferguson 1993; König 1999).

[2] So ist durchaus unklar, ob Software in (objektorientierten) Museen bewahrt werden sollte oder ob dafür eher Archive und Bibliotheken zuständig sein sollten.

wir im Folgenden auch auf die weiteren Probleme technischer „Texte“ eingehen, nämlich die Frage der Codierung, der Hardwareabhängigkeit und der Authentizität.

3.1 Erstes Problem – Lebensdauer der Datenträger

Abstrakt betrachtet sind Computerprogramme - sei es als Quellcode oder in ausführbarer Form - eine spezielle Form von Information, die stets an einen materiellen Datenträger gebunden ist. Während die Information selbst immateriell und damit theoretisch nicht zerstörbar ist, ist es das Speichermedium sehr wohl.

Wenn man davon absieht, dass sich Computerprogramme im Klartext auch als Ausdrucke auf Papier oder in Form von Diagrammen dokumentieren lässt, wurden im Laufe der Geschichte der elektronischen Datenverarbeitung ist eine ganze Reihe von Speichermedien für Massendaten verwendet worden: Die Spanne reicht von der papiernen Lochkarte, über das Magnetband, Diskette und Festplatte bis zu optischen Speichermedien wie der CD und ihren Abkömmlingen. Für jedes dieser Speichermedien kann man zwei Haltbarkeitsdauern angeben: die physische Haltbarkeitsdauer und die Zeit bis zur Veraltung des Datenträgers.

Auch die Trägermaterialien digitaler Speichermedien unterliegen natürlichen Alterungsvorgängen. Darin liegt eine Gefahr der digitalen Speicherung verborgen, denn es ist weder durch Augenschein erkennbar noch genau prognostizierbar, wann das Kopieren der Daten auf einen neuen Datenträger notwendig wäre. So wurde in den 1990er Jahren häufig darüber berichtet, dass Archiven große Datenbestände u.a. dadurch verloren gingen, dass die Datenträger nicht mehr lesbar waren (Klein 1995; Zimmer 1999). Der Internetkritiker Clifford Stoll resümierte sogar: „Elektronische Medien sind nicht archivierbar“ (Stoll 1996).

Tatsächlich unterliegen digitale Datenträger einer ganzen Reihe von schädlichen Einflüssen: (1) mechanische Einflüsse, (2) chemische Einflüsse, (3) thermische Einflüsse sowie (4) externe Magnetfelder. Die Datenträger altern, über die Zeit akkumulieren sich Fehler und die gespeicherte Information wird in Mitleidenschaft gezogen. Die Bits können schließlich nicht mehr gelesen werden, oder sie werden verändert, und die Information geht verloren.

Die größte Gefahr für magnetische Speichermedien ist dabei nicht etwa die schleichende Entmagnetisierung, wie häufig kolportiert wird. Problematisch ist vielmehr das Trägermaterial und seiner Beschichtung. In früheren Jahrzehnten wurden neben Polyvinylchlorid (PVC) vor allem Celluloseacetat verwendet, das sich bei zu feuchter und

zu warmer Lagerung unter Essiggeruch selbst zersetzt. Später verwendete man zwar auch stabile Polyesterbänder, aber die in den 1970er und 1980er Jahren gern verwendete Bindemitteln, mit denen das magnetisierte Metalloxidpulver auf das Band aufgetragen wurde, neigt dazu, sich bei zu hoher Luftfeuchtigkeit selbst aufzulösen und den Abtastkopf zu verschmieren. Den heute verwendeten Datenbändern billigen Fachleute allerdings eine Lebenserwartung von zehn bis zwanzig Jahren zu.

Bei den optischen Speichermedien herrschte bis vor einigen Jahren noch erhebliche Unsicherheit über die Lebensdauern von CDs und DVDs. Grund hierfür war die Verwendung von Aluminium als Spiegel- bzw. Speicherschicht. Diese konnte bei nicht optimaler Verarbeitung bzw. bei Beschädigung der hauchdünnen schützenden Acryllackschicht rasch stumpf werden. Deshalb rechnete man zunächst mit einer Haltbarkeit von höchstens fünf Jahren (Friedewald 1999). Diese Kinderkrankheiten sind mittlerweile behoben und auch die meisten CDs aus den 1980er Jahren können auch nach mehr als 20 Jahren immer noch problemlos gelesen werden. Heute werden meist Legierungen verwendet, die sehr viel langsamer oxidieren, sodass der CD inzwischen eine Lebensdauer von 50 bis 80, vielleicht sogar 100 Jahren versprochen wird. Bei Beschichtungen aus Gold, Silizium oder Silber-Legierungen scheinen sogar 200 Jahre möglich zu sein. Wie die verwendeten Werkstoffe im Verbund aber tatsächlich altern, weiß kein Mensch.

Meist kommt vor dem physischen Verfall der gespeicherten Daten aber die Unzugänglichkeit durch die Einführung neuer, inkompatibler Datenträger. Ein gutes Beispiel hierfür ist die rasche Folge der Formate bei den Disketten. Die ersten 8-Zoll-Disketten wurden 1969 von IBM eingeführt, bereits 1976 folgte als Nachfolger die 5¼-Zoll-Diskette und 1981 die 3½-Zoll-Diskette. Das Lesen von Disketten macht zwar (noch) keine Probleme, weil Laufwerke noch zahlreich vorhanden und meist auch noch funktionsfähig sind. Mit etwas schaltungstechnischem Aufwand lassen diese sich an heutige Hardware anschließen und mit Hilfe von reichlich im Internet verfügbarer Treibersoftware auch unter modernen Betriebssystemen ansprechen. Es muss aber darauf hingewiesen werden, dass die mechanischen Komponenten, und hier vor allem Plattenlaufwerke alter Computersysteme erfahrungsgemäß als erstes ausfallen (Zabolitzky 2002).

Zusammenfassend besteht als auf Grund der begrenzten Lebensdauer von Datenträgern und Laufwerken ein dringender Handlungsbedarf, als erhaltenswert erachtete Datenbestände in moderne digitale Archive zu überführen. Ein solcher Transfer und die Pflege eines Archivs ist freilich mit erheblichen Kosten verbunden, weshalb dies

zumindest für solche Dokumente kaum praktiziert wird, die für eine Softwaregeschichte benötigt werden. Während die Industrie den Verlust ihres organisatorischen Gedächtnis dabei billigend in Kauf nimmt, haben staatliche Archive zunehmend Probleme, ihren gesetzlich formulierten Aufgaben nachzukommen. So richtete bereits 1990 das amerikanische National Bureau of Census einen Hilferuf an Regierung und Öffentlichkeit als klar wurde, dass große Teile der Rohdaten der Volkszählung von 1960 nicht mehr lesbar war (U. S. House of Representatives Committee on Government Operations 1990; Hedstrom 1991). Dieser Bericht sowie der Aufsatz von Jeff Rothenberg (1995) im Scientific American mobilisierten in den USA erstmal nennenswerte staatliche und private Fördergelder zur Erhaltung des digitalen Erbes.

3.2 Zweites Problem – Codierung, Programmiersprachen und Formate

Das Problem der Datenträger ist aber nicht einmal das gravierendste für den Technikhistoriker. Selbst wenn die Information auf einem Speichermedium noch vorhanden und lesbar ist, und es gelingen sollte, für das Speichermedium ein geeignetes Laufwerk und Leseprogramm verfügbar zu haben, müssen die Daten zusätzlich richtig interpretiert werden.

Im Grunde macht Software, die in Maschinensprache - also in Bitfolgen, die unmittelbar von der Hardware verarbeit werden können - theoretisch die geringsten Schwierigkeiten. Sie ist auf der passenden Hardware sofort lauffähig. Da aber die Hardware ihrerseits ein Problem darstellt, muss in den meisten Fällen auf den Programmcode, also die symbolische Formulierung der Software in einer (höheren) Programmiersprache zurückgegriffen werden.

Liegt dieser Quellcode nicht in gedruckter Form, sondern als elektronisch gespeichertes Dokument vor, hat man es mit den allgemeinen Problemen solcher Dokumente zu tun. Sie liegen als Bitstrom auf einem Datenträger vor, der erst in die vom Menschen lesbare symbolische Notation zurückübersetzt werden muss. Dazu ist Wissen über die verwendete Codierung notwendig. Abbildung 1 macht deutlich, wie sich eine bestimmte Bitfolge interpretieren lässt. Neben der Interpretation als alphanumerisches Zeichen (hier „u“) lässt sich die Bitfolge auch als ganze oder Gleitkomma-Zahl interpretieren. Sie kann allerdings auch als Bild oder Ton interpretiert werden.

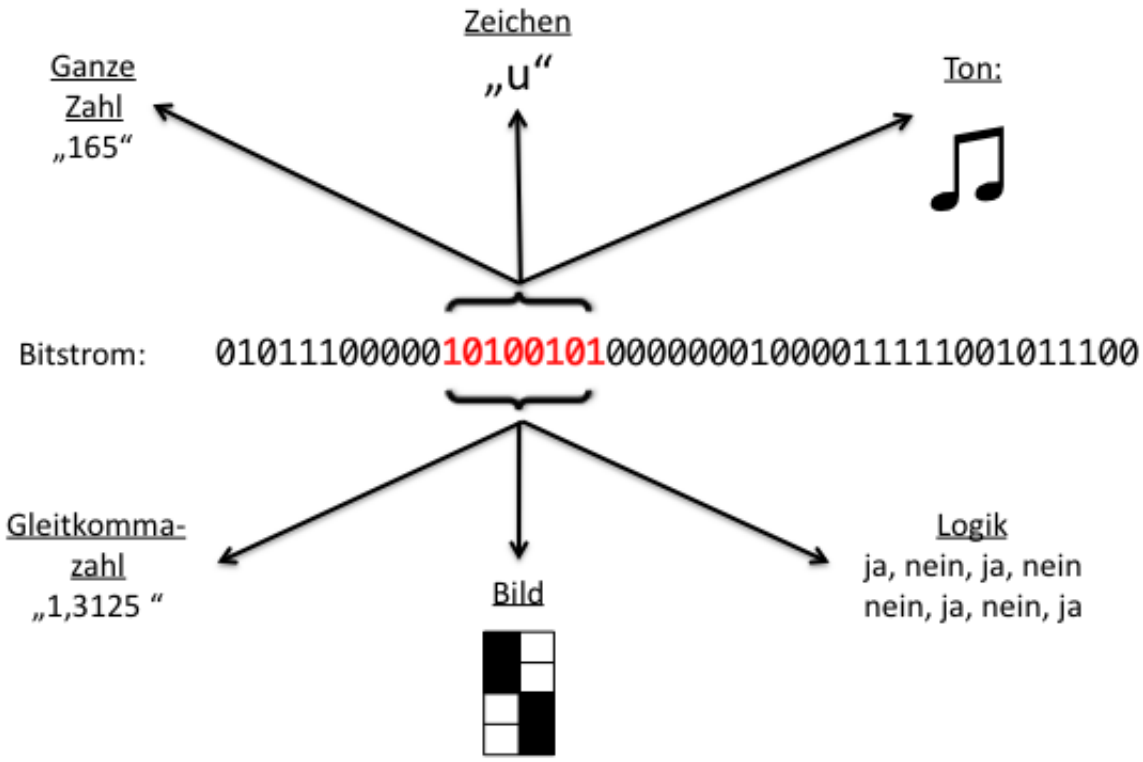

Abb. 1: Mögliche Interpretationen einer Bitfolge (Rothenberg 1995)

Die Codierung von insbesondere alphanumerischen Zeichen war gerade in der Frühzeit der Computerentwicklung von Hersteller zu Hersteller unterschiedlich und hat sich auch im Laufe der Zeit - zusammen mit der Hardwarearchitektur - häufig verändert. Zwar haben sich bereits relativ früh zwei De-Facto-Standards herausgebildet, der verbreitete ASCII-Code und der von IBM verwendete EBCDIC-Code, dennoch sind auf einem Datenträger abgespeicherte Bitfolgen ohne Kenntnis der verwendeten Codierung kaum wieder zu entschlüsseln. Darüber hinaus wurde – zumindest zeitweise – die Nutzung anwendungsspezifischer Dateiformate als Gefahr für die Dokumentenüberlieferung empfunden (Rothenberg 1995). Obwohl dies für die Bestände aus der frühen Computergeschichte weiterhin ein Problem bleibt, ist dies bei neueren Datenbeständen nahezu irrrelevant geworden. Nicht nur Webseiten, sondern auch wichtige Anwendundungsprogramme nutzen mittlerweile standardisierte offene Formate wie XML für die Strukturierung der darzustellenden Inhalte (Rosenthal 2009).

Als Lösung für dieses Problem ist es freilich nicht ausreichend, das Kodierungsschema mit auf dem Datenträger abzuspeichern, weil die gleichen Probleme natürlich auch für

diese Information gelten. Im Extremfall muss die Kodierungsinformation für jeden Datenträger - als eine Art Stein von Rosette des Computerzeitalters - in dauerhaft lesbarer Form angegeben werden.

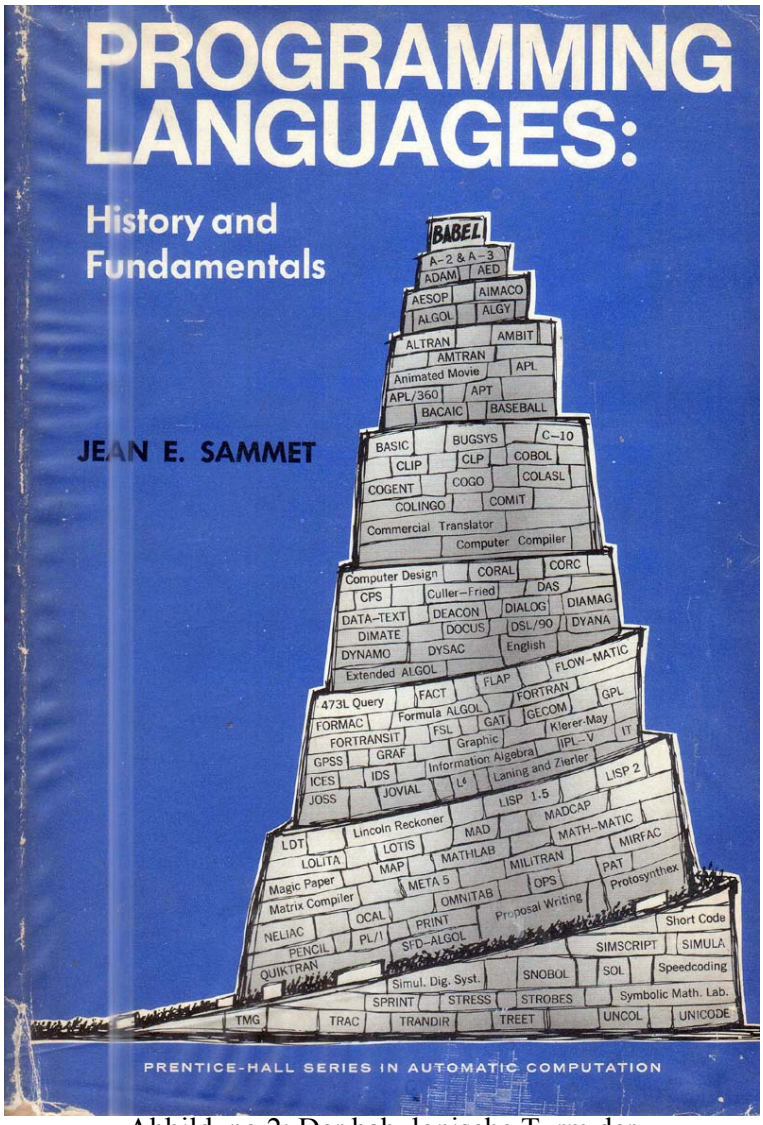

Abbildung 2: Der babylonische Turm der Programmiersprachen (Sammet 1969)

Ist es dann gelungen den elektronisch vorhanden Quelltext einer Software mit der richtigen Textkodierung auszulesen steht man vor dem nächsten Problem. Die meisten Programme sind spätestens ab den 1960er Jahren in einer höheren Programmiersprachen geschrieben. Für die richtige Interpretation des Programmtextes ist daher auch die Kenntnis über die Programmiersprache und das verwendete Übersetzungsprogramm notwendig. Im Laufe der Computergeschichte sind allerdings Hunderte verschiedener Programmiersprachen entwickelt worden – so dass Jean Sammet schon 1969 in ihrem Standardwerk „Programming Languages“ das Bild einer babylonischen Sprachverwirrung bemühte (Abb. 2). Und auch die populäreren Sprachen haben sich im Laufe der Jahre weiterentwickelt und wurde darüber hinaus auch an die Leistungsfähigkeit der verfügbaren Hardware angepasst (Wexelblat 1981; Bergin et al. 1996).

Dies ist einer der wichtigsten Gründe dafür, warum Software ein hybrides Artefakt darstellt, in der zwar abstrakte Verfahren zur Verarbeitung von Daten niedergelegt werden, sich aber implizit immer auch auf die materielle Basis der Berechnung beziehen. Diese Gemengelage macht die Bewahrung von Computerprogrammen in einer für die

weitere Nutzung zugänglichen Form und vor allem deren Analyse und Interpretation schwierig.

Auch wenn die Codierung von Zeichen mit dem Unicode-Standard der Internationalen Standardisierungsbehörde ISO 1991 (wenigstens theoretisch) vereinheitlicht wurde und auch die Zahl der praktisch relevanten Programmiersprachen deutlich abgenommen hat, ist das beschriebene Problem für den Archivbestand von Software aus den vergangenen Jahrzehnten immer noch konzeptionell wie technisch ungelöst.

3.3 Drittes Problem – Authentizität digitaler Dokumente

Authentizität ist eine der zentralen Forderungen, die die Geschichtswissenschaften an Quellen stellt. Authentizität bedeutet in diesem Zusammenhang, dass ein Artefakt tatsächlich von der Person, dem Autor oder Quelle stammt, von der es vorgibt zu stammen, also weder Fälschung noch Fehlzuschreibung ist. Eng damit verbunden ist die Frage nach der Datierung der Quelle. Hier stellt sich das Problem, dass bei Software die traditionellen Verfahren der Quellenkritik weitgehend versagen: Verfahren der äußeren Quellenkritik greifen kaum, da die Speichermedien selbst keinen sicheren Aufschluss über die Authentizität und das Alter der auf ihnen gespeicherten Daten zulassen.

Das eigentliche Problem bei der Feststellung und Sicherung der Authentizität von hat allerdings seine Ursache darin, dass der Programmtext (als eigentlich interessierendes Artefakt) als Bitstrom auf dem Datenträger gespeichert ist, der sich problemlos verändern und kopieren lässt. Dies führt unter anderem dazu, dass im Laufe des Lebenszyklus von Software in der Regel eine ganze Reihe von Versionen entstehen, mit denen die Funktionalität erweitert, Fehler behoben oder Anpassungen auf neue Hardware vorgenommen werden. Nur wenn solche Änderungen genau dokumentiert und eindeutig einem bestimmten Programmtext zugeordnet sind, können historische Programme richtig gelesen, interpretiert und in den Entstehungskontext (der sich aus anderen Dokumenten erschließen lassen muss) eingeordnet werden.[3] Erschwerend kommt hinzu, dass neuere Programme zu den komplexesten technischen Artefakten überhaupt gehören, die aus vielen Millionen Zeilen Code (engl. Lines of Code, LoC) bestehen, deren Auswertung ein

[3] Ein umfangreiche Dokumentation ist zwar mittlerweile ein zentrales Element von Softwarequalität, allerdings war und ist die Dokumentation von Programmcode (bzw. das Schreiben von selbsterklärendem statt kryptischem Code) bei den Entwicklern eine nicht sonderlich beliebte Tätigkeit, so dass die Dokumentation häufig wenig aussagekräftig ist.

extrem aufwendiges Unterfangen darstellt, wie beispielsweise der Rechtsstreit um die Rechte an Teilen des Unix-Quellcodes illustriert.[4]

Da sich die Bits der unmittelbaren Inaugenscheinnahme entziehen, benötigt der Historiker auch stets ein Werkzeug, das den Programmtext liest und in eine für ihn zugängliche Form übersetzt. Abgesehen davon, dass dieses Werkzeug Informationen über die Codierung des Bitstroms benötigt (s.o.) muss auch sichergestellt werden, dass durch diese Übersetzung keine Verfälschungen entstehen (Lynch 2000). Dass darüber hinaus digitale Dokumente (und damit auch Programme) auch bewusst manipuliert werden können, zeigt der Fall der digitalen Bildverarbeitung sehr anschaulich und erschwert zusätzlich die Bewertung der Authentizität.

Es lässt sich freilich die Frage stellen, ob sich der Historiker für die Analyse von Software und Computersystemen auf die Ebene des Programmcodes begeben muss (Grier 2001). Für die ersten Jahre der Computertechnik ist dies gewiss notwendig, da Programme zu jener Zeit noch sehr maschinennah und ohne etablierte Methoden entwickelt wurden. Mit dem Aufkommen des so genannten „Software Engineering“ wurde ab etwa 1970 eine Trennung von Architektur bzw. Entwurf (Programmieren im Großen) und Implementierung (Programmieren im Kleinen) eingeführt (z. B. Royce 1970), wobei letztere mit Hilfe von Werkzeugen zu einem gewissen Maße automatisiert wurde. In vielen Fällen ist deshalb eine Analyse der Architektur ausreichend, allerdings handelt es sich dabei ebenfalls um digitale Texte mit allen genannten Schwierigkeiten.

Moderne Software ist außerdem vielfach mit technischen Mechanismen ausgestattet, die ein Kopieren verhindern sollen. Beim Kopieren der Software von den ursprünglichen Datenträgern in digitale Archive kommt es darauf an, dass die Programme nicht wie beim „cracken“ (dem gewaltsamen Entfernen des Kopierschutzes) üblich, verändert werden, sondern dass der Kopierschutz selbst mitkopiert wird. Hier stehen den Archivaren bisher nur rudimentär geeignete Werkzeuge zur Verfügung. Rechtlich betrachtet wäre es für die Bewahrung von Software notwendig, dass eine Schranke im Urheberrecht implementiert wird, die es Archiven erlaubt, für konservatorische Zwecke Kopierschutzmechanismen zu umgehen. In den USA wurde ein entsprechender Passus bereits 2003 in den „Digital Millenium Copyrights Act“ eingefügt (Hirtle 2004).

[4] Das Unternehmen SCO Group behauptete 2003 sämtliche Rechte an den Methoden und Konzepten des Betriebssystems UNIX sowie einen Anspruch auf Geheimhaltung des Codes aller auf Unix basierenden Projekte der Unix-Lizenznehmer von AT&T zu besitzen und hat in der Folge Firmen wie IBM, Novell, AutoZone und DaimlerChrysler wegen Verletzung dieser Rechte verklagt. Im Zuge der Beweisaufnahme mussten beispielsweise 900 Millionen Codezeilen der IBM-Betriebssysteme AIX und Dynix untersucht werden.

Schließlich muss man berücksichtigen, dass sich die Qualitäten von Software erst in der Interaktion zwischen Mensch und Maschine erschließen lassen und diese deshalb bei der historischen Analyse mit berücksichtigt werden müsste (Oudshoorn et al. 2003; Hellige 2008).

4 Modelle für die Bewahrung des digitalen Erbes

Den Königsweg für die Bewahrung von Software als digitalen Erbes, der die Anforderungen aller Interessengruppen gerecht wird, gibt es nicht; es werden vielmehr verschiedene Wege eingeschlagen werden müssen.

4.1 Erstes Weg: Musealisierung

Der erste Weg ist das des Technikmuseum: Neben den Daten selbst müssten auch die Geräte und die Programme aufbewahrt werden, mit denen sie erzeugt wurden.

Es wurde bereits erwähnt, dass die Existenz eines physischen Objekts aus der Forschungsperspektive einen erkenntnistheoretischen Mehrwert aufweist, der sich nicht aus Büchern, Websites oder anderen Repräsentationen des Objekts erzielen lässt. Dies gilt besonders für so ein schwieriges und wenig anschauliches Objekt wie einen Computer (und die darauf laufende Software). Dies ist der Grund warum historische Rekonstruktionen eine wichtige Rolle einnehmen und warum die Frage der Authentizität hier so entscheidend ist.

Die auf den ersten Blick naheliegende Art der Konservierung ist also die Erhaltung des kompletten Computersystems im Originalzustand: Dabei würde Software auf den originalen Speichermedien gespeichert, von den originalen Laufwerken gelesen und schließlich auf der passenden Hardware mit einem dazugehörigen Betriebssystem ablaufen. Eine solche Alternative mag zwar vom Standpunkt des Ingenieurs interessant sein, ist aber aus historiographischer wie konservatorischer Sicht nicht praktikabel und nicht nachhaltig. Abgesehen davon, dass für orthodoxe Kuratoren und Konservatoren das originale technische Artefakt sakrosankt ist und nicht verändert werden darf, ergeben sich eine Vielzahl von praktischen Problemen: Zum einen ist es nicht möglich, die ganze Vielfalt an Computern, die in den letzten 75 Jahren als Einzelstücke, in Klein- oder Großserie hergestellt worden sind, in einem möglichst intakten Zustand zu sammeln und zu dokumentieren. Zum anderen ist es enorm aufwändig und kostspielig, historische Computer betriebsbereit zu erhalten:

Die für die Instandhaltung bzw. Reparatur der Geräte notwendigen Ersatzteile sind heute vielfach nicht mehr verfügbar.[5] Die Verwendung von funktionsgleichen Ersatzteilen stellt hingegen eine ggf. akzeptable Verfälschung des originalen Artfakts dar. Da die Halbleiterherstellung ein komplizierter und kapitalintensiver technologischer Prozess ist, ist eine Neuproduktion historischer Bauelemente in der Regel nicht möglich (Mahoney 1988; Rothenberg 1995).

Computersysteme bestanden normalerweise nicht nur aus der Zentraleinheit (CPU) und einer mehr oder weniger großen Zahl von Peripheriegeräten, sondern darüber hinaus auch noch aus umfangreichen Hilfsaggregaten, vor allem für die Stromversorgung und Klimatisierung. Der Leistungsaufnahme einer solchen Installation liegt (bei Computern der ersten Generation) bei deutlich über 100 kW.

Auch wenn die materiellen Elemente eines Computersystems funktionsfähig sind, ist das für den Betrieb notwendige stille Wissen (z.B. beim Testen der Hardware oder bei der Fehlersuche) heute nicht mehr (oder nur noch in den Köpfen weniger Zeitzeugen) verfügbar (Swade 2002; Spicer 2005).[6]

Eine Rekonstruktion ist ohnehin nur bei isolierten Computersystemen sinnvoll. Im Zuge der immer stärkeren Vernetzung von Computern ist der objektorientierte Ansatz ohnehin immer weniger zielführend, weil die oftmals globalen kommunikativen Verflechtungen nicht physisch rekonstruierbar sind (Schwens et al. 2004). Dies ist beispielsweise eines der zentralen Schwierigkeiten bei der Archivierung von Internet-Inhalten, bei denen zwar lokale Seiten gespeichert werden können, nicht aber die Verflechtung mit anderen Elementen im Netz (Drösser 2008).[7]

Nichtsdestotrotz gibt es Bereiche, in denen es zur Musealisierung keine Alternative zu geben scheint. Bei Software mit Echtzeitanforderungen (z.B. Computerspiele) ist es

---

[5] Vor einigen Jahren wurde berichtet, dass die NASA für die Instandhaltung ihrer Space Shuttle-Flotte bei Online-Auktionshäusern wie ebay ausgediente Intel 8086-CPUs und passende Mainboards einkaufen musste, weil diese Ersatzteile anderweitig nicht mehr erhältlich waren (Broad 2002).

[6] Dies zeigte sich beispielsweise bei der Rekonstruktion des britischen Pioniercomputers „Colossus“ (1943), der nach dem zweiten Weltkrieg vollständig demontiert wurde. Die einzigen Konstruktionspläne waren 1960 verbrannt. Aus Geheimhaltungsgründen waren die Entwickler jahrelang zum Schweigen verpflichtet (Sale 2005).

[7] Gänzlich sichtbar wird dies bei den soziale Netzwerken der vergangenen Jahre, die durch das Studium der dafür genutzten Software nicht erschlossen werden können, weil das Charakteristische dieser Anwendungen erst durch die Kommunikation zwischen den Nutzern entsteht.

beispielsweise notwendig, dass das Interaktionsverhalten erhalten bleibt, das durch das Zusammenspiel von Hardware, Betriebssystem und Anwendungssoftware bestimmt wird. Für die Rekonstruktion von Aussehen und Handhabung (engl. Look and Feel) kommt es darüber hinaus auch auf die Eigenschaften der originalen Peripheriegerät (z.B. Bildschirmauflösung) an (Lange 2009).

4.2 Zweiter Weg: Emulation und Simulation

Eine weiterer Weg zur Erhaltung der Langzeitverfügbarkeit digitaler Objekte liegt in der Emulation bzw. Simulation. Dieser Ansatz sieht vor, dass die Systemvoraussetzungen, die zur Nutzung älterer digitaler Objekte notwendig sind, durch spezialisierte Software auf aktuellen marktgängigen Systemen nachgebildet (emuliert) werden können. Die digitalen Objekte selbst werden dabei möglichst unverändert erhalten (Schwens et al. 2004).

Die Emulation setzt voraus, dass ausreichend detaillierte technische Metainformation über die veralteten Systeme vorhanden ist, so dass die Ausgangslage rekonstruiert werden kann. Da dies – wie bereits im Zusammenhang mit der Musealisierung erwähnt – nicht immer der Fall ist, kann die Emulation nur dort praktiziert werden, wo ein sehr großer Datenbestand den Aufwand lohnt. Abbildung 3 zeigt, dass viele Elemente zusammenwirken müssen, bis ein Dokument lesbar gemacht werden können. Neben der Lesbareit des Datenträgers und der Funktionsfähigkeit eines entsprechenden Lesegeräts muss nicht nur ein passender Hardware-Emulator entwickelt werden, sondern es müssen auch das originale Betriebssystem und die Originalsoftware auf der emulierten Hardware zum Laufen gebracht werden.

Die Emulation ist eine sinnvolle Strategie vor allem für den musealen Bereich, wo es, wie im Bostoner Computermuseum, darauf ankommt, dem Besucher möglichst effizient einen lebendigen Eindruck von historischer Software zu geben. Emulation kann allerdings nie mehr als einen „Eindruck“ von historischer Software vermitteln. Wenn Software auf einem emulierten System läuft, dass wie bei modernen Computern mit Bildschirm, Tastatur und ggf. Maus ausgestattet ist, mag dieser Eindruck nicht weit vom Original entfern sein (wie etwa bei der Emulation beliebter Homecomputer aus den 1980er Jahren).

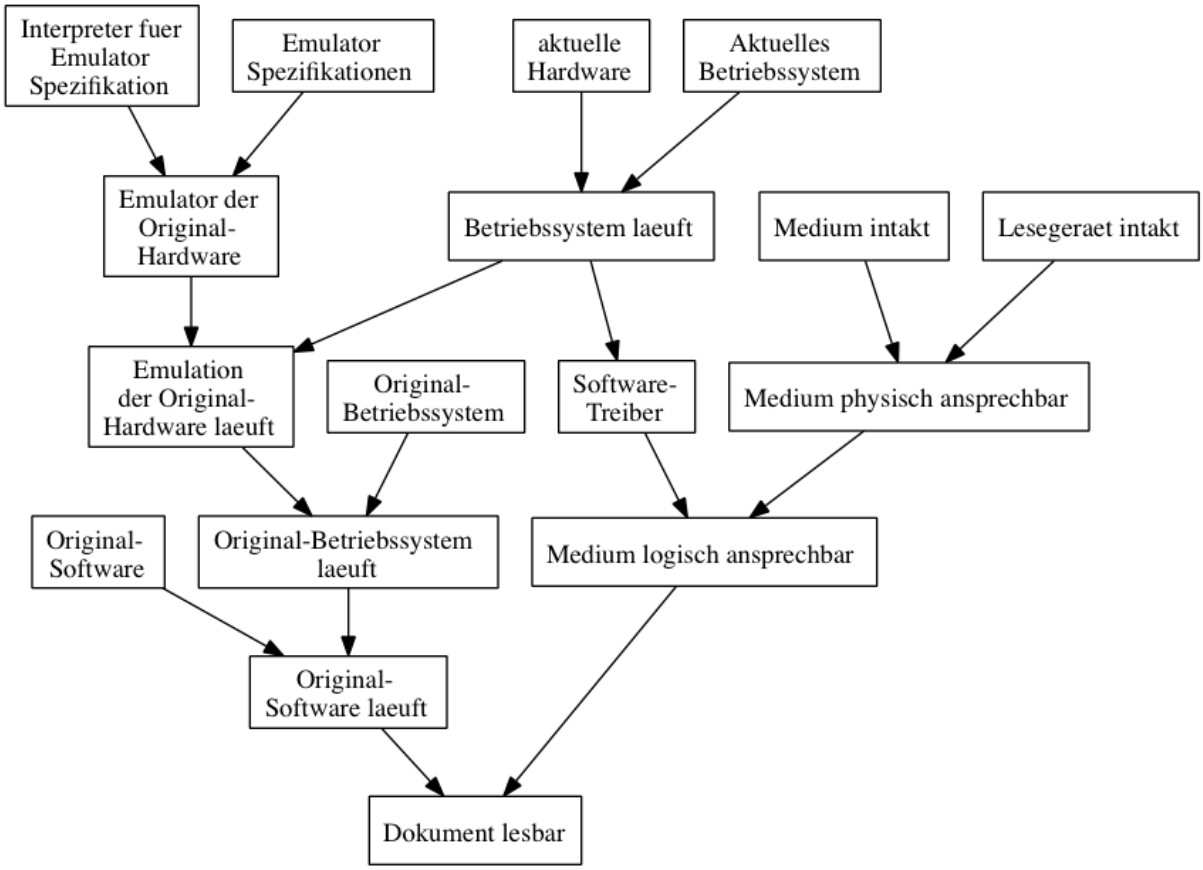

Abbildung 3: Lesen obsoleter digitaler Dokumente mittels Emulation (Rothenberg 1999)

Anders sieht dies bei älteren Computern aus, die ein gänzlich anderes Bedienkonzept hatten, das im Zuge der Emulation mehr oder weniger vollständig und überzeugend auf den Bildschirm gebracht werden muss (Abb. 4). Es besteht sogar die Gefahr, dass eine solche Mischung aus Historizität und moderner Technik zu völlig falschen Schlüssen verleitet. Insbesondere kann die Emulation nur in Ausnahmefällen das reale dynamische Verhalten abbilden, dass bereits im Zusammenhang mit Computerspielen erwähnt wurde.

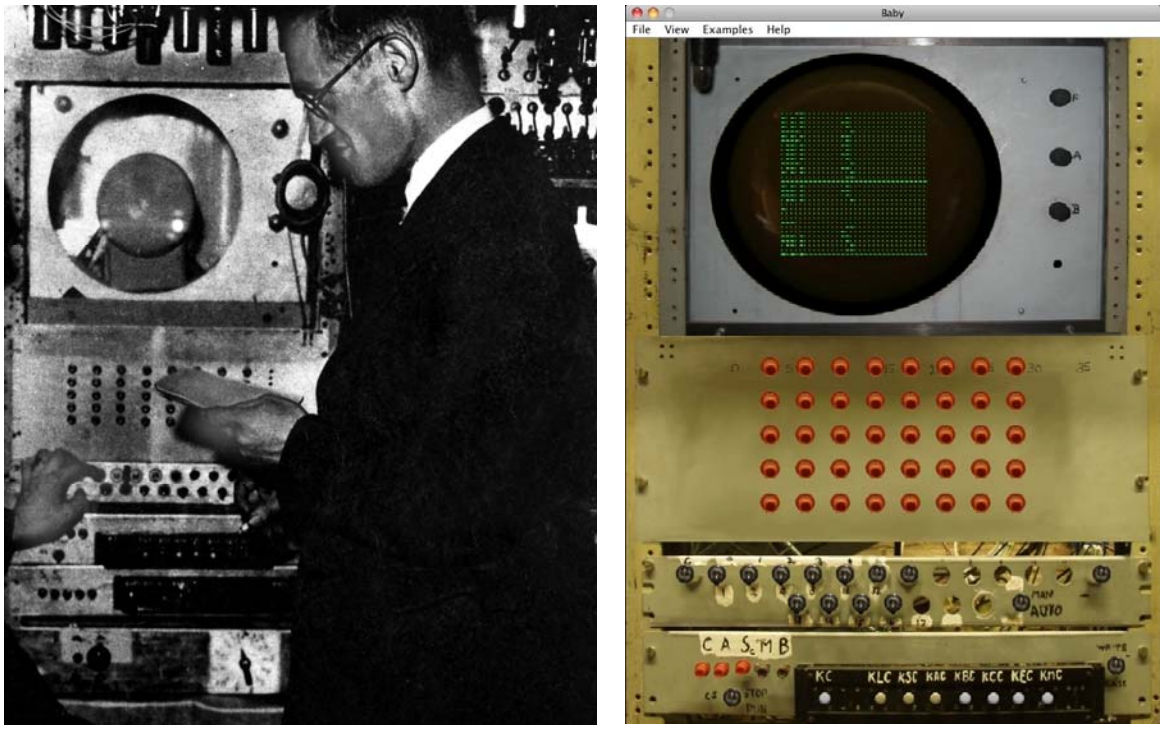

Abbildung 3: Bedienkonsole des Manchester „Baby“ Computers (1948) und dessen Simulation (Quelle: http://www.cs.manchester.ac.uk/Digital60/)

4.3 Dritter Weg: Datenmigration

Der dritte Weg ist die Migration, also das regelmäßige Umkopieren von einem Datenträger auf den anderen. Sie wird momentan von vielen Archivaren als einzig realistische Möglichkeit der Langzeitsicherung beurteilt. Aber auch sie setzt entweder voraus, dass mit den Informationen die Programme aufgehoben werden, mit deren Hilfe sie erzeugt wurden und wieder entziffert werden können; und dass diese vor den Informationen selbst auf die neueren Maschinen migrieren. Oder aber dass sich maschinenunabhängige offene Standardformate entwickeln, die die Chance hätten, auch in die weitere Zukunft zu überleben. Die ewige Festschreibung solcher Standards scheint aber den bisherigen Lebensgesetzen der Informationstechnikindustrie zu widersprechen. Aber es besteht immerhin die Aussicht, dass einige von ihnen solche Verbreitung finden und solche Mengen an Information regieren, dass kein Entwickler es sich mehr leisten kann, den Standard leichthin fallen zu lassen.

Aber vielleicht liegt die Zukunft der Datenmigration auch in der Verlagerung von Speicherkunktionen ins Netz, also im sogenannte Cloud Computing (Lin et al. 2009).[8] Bei einer Speicherung von Daten in der „Wolke“ statt auf indivuellen Datenträgern verliert das Problem der Lebensdauer einzelner Datenträger zunehmend an Bedeutung. In den hinter der Wolke stehenden riesigen Speicherbanken der Service-Anbieter werden die Festplatten regelmäßig ausgetauscht und Daten umkopiert, um den Anforderungen der Nutzer gerecht zu werden. Für elektronische Dokumente aus der Vor-Netzwerk-Zeit, die der Nachwelt erhalten bleiben sollen und (nur) auf Lochkarten, magnetischen und optischen Datenträgern gespeichert sind, ist es allerdings dringend geboten, diese in moderne digitale Archive zu überführen.

Aus methodischer Sicht ist aber auch die Datenmigration nicht die perfekte Lösung. Zum einen können durch die Konvertierung von Daten in standarisierte Formate Verfälschungen entstehen und die Daten im schlimmsten Fall unbenutzbar machen. Auch die Authentizität der Dokumente kann so nicht garantiert werden.

5 Fazit

Software ist für die Geschichte der Informationstechnik eine wichtige Sachquelle, deren Bewahrung als digitales Erbe in jeden Fall wichtig ist. Dabei stellt Software eine besondere „Gattung“ digital entstandener Texte dar mit besonderen Eigenschaften dar. Diese Besonderheiten haben Ihre Ursache im hybridem Charakter von Software (materielles vs. immaterielles Artefakt, abstrakte Idee vs. technische Lösung).

Ein wichtiges Problem bei der Konservierung von Software ist die Lebensdauer der materiellen Datenträger und Laufwerke, die vor allem für Software relevant ist, die auf Magnetbändern und Disketten gespeichert sind (also vor allem Software, die zwischen Mitte der 1950er Jahre und Ende der 1980er Jahre entwickelt wurde). Ebenso wichtig ist die „babylonische Sprachverwirrung“ mit einer fast unüberschaubaren Zahl an Kodierungsschemata, Dateiformaten und Programmiersprachen, die es erschweren, historische Software physisch und intellektuell zu lesen. Zum anderen gibt es schließlich das grundsätzliche Problem, dass sich digitale Dokumente problemlos manipulieren lassen, ohne dass dies für den (späteren) Betrachter unmittelbar erkennbar ist und somit die Feststellung der Authentizität erschwert ist.

[8] Das Cloud Computing ist ein neues Modell von IT-Outsourcing, bei dem abstrahierte IT-Infrastrukturen (z.B. Speicherkapazität oder Rechenleistung) von einem Dienstleister über das Netzwerk zur Verfügung gestellt werden.

Die bislang im Bereich der Langzeitarchivierung diskutieren Ansätze (Musealisierung, Emulation, Migration) sind bislang aus Sicht des Historikers noch nicht zufriedenstellend.

So hat die von den meisten Experten aus guten Gründen als nicht praktikabel und zu teuer abgelehnte Musealisierung durchaus seine Vorteile. Neben einer höheren Authentizität besteht diese vor allem darin, dass die Software soweit als möglich in ihrem realen Umfeld analysiert werden kann.

Die Migration von Daten bis hin zur Auslagerung ins Internet stellt das andere Extrem dar. Hier sind die Kosten im Vergleich zur Musealisierung deutlich niedriger, dafür muss sich die Analyse der technischen Artefakte allerdings auf reine „Trockenübungen" beschränken. Verloren geht dabei das Element der Anschaulichkeit, das gegenständliche Quellen gegenüber schriftlichen Quellen auszeichnet.

Die Emulation historischer Computersysteme scheint ein Kompromiss zwischen Migration und Musealisierung zu sein, weil es das Element der Anschauung beibehält, ohne sich zusätzlich auf die Probleme einer verfallenden Hardware einlassen zu müssen. Emulation ist aber stets nur eine Annäherung an die historische Realität, weil heutige Annahmen über das zu emulierende System in den Entwurf des Emulator einfließen. Bei der Interpretation von Erkenntnissen, die man mit diesem Ansatz gewonnen hat, muss man besonders vorsichtig sein.

6 Literatur

Bergin, J., Thomas J.; Gibson, J., Richard G. (Hrsg.) (1996): *History of Programming Languages–II*. Reading, Mass.: Addison-Wesley.

Broad, W. J. (2002): For Old Parts, NASA Boldly Goes ... on eBay. In: *New York Times* vom 12 Mai 2002. http://www.nytimes.com/2002/05/12/technology/ebusiness/12NASA.html

Broy, M.; Denert, E. (Hrsg.) (2002): *Software Pioneers: Contributions to Software Engineering*. Berlin und Heidelberg: Springer.

Campbell-Kelly, M. (2003): *From Airline Reservation to Sonic the Hedgehoc: A History of the Software Industry*. Cambridge, Mass.: MIT Press.

Coy, W. (2004): Was ist Informatik? Zur Entstehung des Faches an den deutschen Universitäten. In: Hellige, H. D. (Hrsg.): *Geschichten der Informatik: Visionen, Paradigmen, Leitmotive*. Berlin, Heidelberg: Springer, S. 473-498.

Drösser, C. (2008): Das digitale Alexandria. In: *Die Zeit* 4/2008 vom 17. Januar 2008.

Eulenhöfer, P. (1999): *Die formale Orientierung der Informatik: Zur mathematischen Tradition der Disziplin in der Bundesrepublik Deutschland*. Technische Universität Berlin, Dissertation.

Ferguson, E. S. (1993): *Das innere Auge: Von der Kunst des Ingenieurs*. Basel, Boston und Berlin: Birkhäuser.

Friedewald, M. (1999): Zuverlässig nur auf Zeit. In: *Elektronik* Nr. 16, S. 28–30.

Geppert, A. C. T. (1994): Forschungstechnik oder historische Disziplin? Methodische Probleme der Oral History. In: *Geschichte in Wissenschaft und Unterricht* 45, Nr. 5, S. 303-323.

Grier, D. A. (2001): Mapping the History of Computing—Software Issues: A report on the International Computer History Conference 2000, Heinz Nixdorf Museums Forum, Paderborn, Germany. In: *IEEE Annals of the History of Computing* 23, Nr. 2, S. 68-71.

Hamming, R. W. (1980): We Would Know What They Thought When They Did It. In: Metropolis, N. C.; Howlett, J. et al. (Hrsg.): *A History of Computing in the Twentieth Century - A Collection of Essays*. New York: Academic Press, S. 3-9.

Hashagen, U.; Keil-Slawik, R.; Norberg, A. L. (Hrsg.) (2002): *History of Computing: Software Issues*. Berlin, Heidelberg, u.a.: Springer.

Hedstrom, M. (1991): Understandig Electronic Incunabla: A Framework for Research on Electronic Records. In: *American Archivist* 54, Nr. 3, S. 334-354.

Hellige, H. D. (Hrsg.) (2008): *Mensch-Computer-Interface: Zur Geschichte und Zukunft der Computerbedienung*. Bielefeld: Transkript Verlag.

Hirtle, P. B. (2004): Digital Preservation and Copyright. In: *E-Business* 5, Nr. 2, S. 58-64.

Klein, S. (1995): Fröhlicher Wildwuchs. In: *Der Spiegel* 40/1994, S. 228-230.

König, W. (1999): *Künstler und Strichezieher: Konstruktions- und Technikkulturen im deutschen, britischen, amerikanischen und französischen Maschinenbau zwischen 1850 und 1930*. Frankfurt am Main: Suhrkamp.

Kuny, T. (1998): A Digital Dark Ages? Challenges in the Preservation of Electronic Information. In: *International Preservation News* Nr. 17, S. 8-13.

Lange, A. (2009): Save Game: Die Bewahrung komplexer Artefakte am Beispiel von Computerspielen. In: Sieck, J.; Herzog, M. A. (Hrsg.): *Kultur und Informatik: Serious Games*. Boizenburg: Verlag Werner Hülsbusch, S. 189-200.

Lin, G.; Fu, D.; Zhu, J. (2009): Cloud Computing: IT as a Service. In: *IT Professional* 11, Nr. 2, S. 10-13.

Lynch, C. A. (2000): Authenticity and Integrity in the Digital Environment: An Exploratory Analysis of the Central Role of Trust. In: *Authenticity in a Digital Environment*. Washington, D.C.: Council on Library and Information Resources, S. 32-50.

Mahoney, M. S. (1988): History of Computing in History of Technology. In: *Annals of the History of Computing* 10, Nr. 2, S. 113-125.

Mahoney, M. S. (2008): What Makes the History of Software Hard. In: *IEEE Annals of the History of Computing* 30, Nr. 3, S. 8-18.

Norberg, A. L.; O'Neill, J. E.; Freedman, K. (1996): *Transforming Computer Technology: Information Processing in the Pentagon 1962-1986*. Baltimore Md.: Johns Hopkins University Press.

Oudshoorn, N.; Pinch, T. (Hrsg.) (2003): *How Users Matter: The Co-Construction of Users and Technology*. Cambridge, Mass. and London: MIT Press.

Rosenthal, D. S. H. (2009). How Well Are We "Ensuring the Longevity of Digital Documents"?. CNI Spring Task Force Meeting, 6-7 April 2009, Minneapolis MN, Coalition for Networked Information.

Rothenberg, J. (1995): Ensuring the Longevity of Digital Documents. In: *Scientific American* 272, Nr. 1, S. 24-29.

Rothenberg, J. (1999). *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation*. A Report to the Council on Library and Information Resources. Washington: Council on Library and Information Resources.

Royce, W. W. (1970): Managing the Development of Large Software Systems: Concepts and Techniques. In: *Technical Papers of Western Electronic Show and Convention (WesCon) August 25-28, 1970, Los Angeles, USA*. IEEE, S. 1-9.

Sale, A. E. (2005): The Rebuilding of Colossus at Bletchley Park. In: *IEEE Annals of the History of Computing* 27, Nr. 3, S. 61-69.

Sammet, J. E. (1969): *Programming Languages: History and Fundamentals*. Englewood Cliffs, NJ: Prentice Hall.

Schwens, U.; Liegmann, H. (2004): Langzeitarchivierung digitaler Ressourcen. In: Kuhlen, R.; Seeger, T. et al. (Hrsg.): *Grundlagen der praktischen Information und Dokumentation Band. 1: Handbuch zur Einfu□hrung in die Informationswissenschaft und -praxis*, 5., vo□llig neu gefasste Aufl. Mu□nchen: Saur, S. 567- 570.

Shustek, L. J. (2006): What Should We Collect to Preserve the History of Software? In: *IEEE Annals of the History of Computing* 112, 110-111, Nr. 28, S. 4.

Spicer, D. (2005): The IBM 1620 Restoration Project. In: *IEEE Annals of the History of Computing* 27, Nr. 3, S. 33-43.

Stadtmu□ller, W. (1999): Sachquellen. In: Schreiber, W. (Hrsg.): *Erste Begegnungen mit Geschichte. Grundlagen historischen Lernens*, Band 1. Neuried, S. 391-404.

Stahlschmidt, R. (1977): *Quellen und Fragestellungen einer deutschen Technikgeschichte des frühen 20. Jahrhunderts bis 1945*. Göttingen: Vandenhoeck und Ruprecht.

Stoll, C. (1996): *Die Wüste Internet: Geisterfahrten auf der Datenautobahn*. Frankfurt am Main: S. Fischer.

Swade, D. (2002): Collecting Software: Preserving Information in an Object-Centred Culture. In: Hashagen, U.; Keil-Slawik, R. et al. (Hrsg.): *History of Computing: Software Issues*. Berlin, Heidelberg, u.a.: Springer, S. 227-235.

Troitzsch, U.; Wohlauf, G. (1980): Einführung. In: Troitzsch, U.; Wohlauf, G. (Hrsg.): *Technik-Geschichte. Historische Beiträge und neuere Ansätze*. Frankfurt am Main: Suhrkamp, S. 10-42.

U. S. House of Representatives Committee on Government Operations (1990). *Taking a byte out of history: The archival preservation of federal computer records*. House Report 101-978. Washington: U.S. Government Printing Office.

Wexelblat, R. L. (Hrsg.) (1981): *History of Programming Languages*. Boston: Academic Press.

Zabolitzky, J. G. (2002): Preserving Software: Why and How. In: *Iterations* 1. http://www.cbi.umn.edu/iterations/zabolitzky.html

Zimmer, D. E. (1999): Das große Datensterben. Von wegen Infozeitalter: Je neuer die Medien, desto kürzer ist ihre Lebenserwartung. In: *Die Zeit* vom 18. November 1999, S. 45-46.

Angaben zu den Autoren:

Dr.-Ing. Michael Friedewald, *1965, Fraunhofer-Institut für System- und Innovationsforschung, Breslauer Straße 48, 76139 Karslruhe, Michael.friedewald@isi.fraunhofer.de

Timo Leimbach, M.A., *1976, Fraunhofer-Institut für System- und Innovationsforschung, Breslauer Straße 48, 76139 Karslruhe, timo.leimbach@isi.fraunhofer.de