Softwarealterung aus Sicht des IT-Managements – Ergebnisse einer qualitativ-empirischen Analyse in der Finanzindustrie

Wiebke Kappenberg, Paul Drews

Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg
wiebke.kappenberg@gmail.com
drews@informatik.uni-hamburg.de

Abstract: Seit 20 Jahren wird das Phänomen der Softwarealterung in der Forschung überwiegend aus der Perspektive der Softwareentwicklung untersucht. Aus Sicht des IT-Managements ist dies unzureichend, da für das Management komplexer Anwendungslandschaften auch organisatorische, finanzielle und strategische Faktoren zu berücksichtigen sind. In Banken und Versicherungen setzen sich die Anwendungslandschaften aus vielfältigen Anwendungen unterschiedlicher Technologien und Generationen zusammen. Es stellt sich in diesem Zusammenhang die Frage, wie und anhand welcher Indikatoren das IT-Management in der Finanzindustrie das Softwarealter bewerten kann und welche Maßnahmen gegen die Softwarealterung ergriffen werden können. Um diese Fragen zu beantworten, wurden in einer qualitativ-empirischen Studie Experteninterviews mit 28 mit Vertretern des IT-Managements der Finanzindustrie durchgeführt. Die Ergebnisse zeigen, dass ein erweiterter Ansatz zur Bewertung von Softwarealterung erforderlich ist.

1 Einleitung

Seit Mitte der 90er Jahre, nachdem Parnas mit der Metapher Softwarealterung (Software Aging) die Aufmerksamkeit der Forschung auf dieses Phänomen lenkte [Pa94], wurden zahlreiche Methoden zur Bewertung des Softwarealters, zum Refactoring und zur Verjüngung von Software (Software Rejuvenation) [Hu95] entwickelt. Diese betrachten jedoch primär technische Eigenschaften, wie beispielsweise die zunehmende Anzahl an Fehlern, den steigenden Ressourcenverbrauch sowie den damit einhergehenden Performanceverlust [BF09] [Co11]. Diese technische Sichtweise ist geeignet, um Softwarealterung aus der Sicht der Softwareentwicklung oder des Produktmanagements zu bewerten. Aus der Sicht des IT-Managements bedarf es jedoch anderer Methoden, da die Bewertung für eine Vielzahl von Anwendungen vorgenommen werden muss. Dabei sind auch nicht-technische Indikatoren zu berücksichtigen. Hinzu kommt, dass für viele Anwendungen keine Möglichkeit besteht, den Quellcode zu analysieren oder zu verändern. Ziel dieses Beitrags ist es daher, eine erweiterte Sichtweise auf das Phänomen der Softwarealterung zu entwickeln, die auch die besonderen Interessen und

Herausforderungen des IT-Managements berücksichtigt. Diese erweiterte Sichtweise wird als das Ergebnis der Analysephase eines gestaltungsorientierten Vorgehens [He04] verstanden, in dessen weiterem Verlauf ein Artefakt entwickelt werden soll, das ein systematisches Management des Softwarealters unterstützt.

Im Anschluss an diese Einleitung ist der Artikel in vier Abschnitte untergliedert. Im folgenden Abschnitt werden die Ergebnisse einer systematischen Literaturanalyse zu Indikatoren für Softwarealterung beschrieben. Im dritten Abschnitt wird das Vorgehen bei der Durchführung einer qualitativ-empirischen Studie zur Softwarealterung aus Sicht des IT-Managements in der Finanzindustrie beschrieben. Die empirische Untersuchung hat das Ziel, Indikatoren zur Bewertung des Softwarealters sowie genutzte Maßnahmen zum Umgang mit diesem Phänomen zu identifizieren. Die Ergebnisse der empirischen Untersuchung werden im vierten Abschnitt beschrieben. Der Artikel schließt mit einer Zusammenfassung und einem Ausblick im fünften Abschnitt.

2 Bisheriger Stand der Forschung zur Softwarealterung

Seit Mitte der 90er Jahre werden Methoden zur Identifikation von Softwarealterung und Maßnahmen zum Umgang mit diesem Phänomen erforscht. Um die bisherigen Ergebnisse systematisch zu konsolidieren, wurde eine Literaturrecherche nach Brink [Br13] durchgeführt, die in drei Schritte untergliedert ist: (1) offene Literaturrecherche in elektronischen Volltextdatenbanken, Fachliteratur und Lehrbüchern, anschließend: Definition und Abgrenzung des Begriffs Softwarealterung (2) Erstellung einer Suchwortliste, diese umfasste zunächst die Suchterme „Software Aging“, „Software Age“, „IT-Management“, „Criteria“ und „Method“, später wurde die Liste um die folgenden Begriffe erweitert: „Code Deterioration“, „Software Risks“, „Software Quality“, „Software Rejuvenation“, „Software Maintenance“, „Software Refactoring“, „Software Health“, „Software Fitness“, „Software Lifetime Expectancy“ (3) Verwendung der Suchterme (kombiniert und einzeln) für eine ausführliche Recherche in sieben relevanten Datenbanken der Wirtschaftsinformatik [KW06], acht Journalen aus dem *„*Senior Scholars*' Basket of Journals“* der AIS sowie in den Publikationen von vier Konferenzen der Wirtschaftsinformatik. Die Literaturrecherche wurde dokumentiert und die Literatur auf Relevanz geprüft. Diese war gegeben, wenn die Abstracts Hinweise auf Methoden und Indikatoren für die Bewertung von Softwarealterung enthielten. Es verblieben 85 Quellen für die nähere Analyse. Im Folgenden werden die Ergebnisse der Literaturanalyse kurz zusammengefasst.

David L. Parnas legte mit seinem Artikel von 1994 den Grundstein für viele anschließende Forschungsarbeiten zum Thema Softwarealterung. Er beschreibt das Phänomen in seinem Artikel wie folgt: *„Programs, like people, get old. We can 't* prevent aging, but we can understand its causes, take steps to limits its effects, temporarily reverse some of the damage it has caused, and prepare for the day when the software is no longer viable.*“* [Pa94] Der Alterungsprozess von Software kann nur bedingt aufgehalten werden und endet stets mit der Abschaltung der veralteten Software oder der Ablösung durch eine neue. Faktoren, die den Alterungsprozess beeinflussen können, sieht Parnas insbesondere in der fehlenden Aktualisierung der Software sowie in

häufigen Änderungen durch unterschiedliche Mitarbeiter [Pa94]. Letztere können dazu führen, dass der Code und die Architektur strukturell zerstört werden. Eine schlechte Wartbarkeit führt zu weiteren Problemen wie abnehmender Leistungsfähigkeit, steigender Fehlerzahl und reduzierter Verfügbarkeit. Parnas geht nur kurz auf die Folgen ein, zu denen er steigende Kosten für Wartung und Weiterentwicklung sowie eine abnehmende Wettbewerbsfähigkeit zählt. Wie das Softwarealter durch das IT-Management konkret bewertet werden kann, wird in seinem Artikel nicht beschrieben. In der Literatur besteht Einigkeit darüber, dass das Phänomen der Softwarealterung nicht unmittelbar mit dem tatsächlichen Alter der Software in Jahren in Beziehung steht. Selbst wenn das tatsächliche Softwarealter (in Jahren) bekannt ist, so liefert dies keine qualitative Aussage über deren Zustand. Daher thematisiert die Forschung überwiegend die Bestimmung des optimalen Verjüngungszeitpunktes der Software, sodass sie ohne spürbare Beeinträchtigungen lange einsatzfähig bleibt. Die weiteren Artikel und deren Definitionen zur Softwarealterung, die im Rahmen der Literaturrecherche identifiziert wurden, stimmen mit den von Parnas genannten Kerneigenschaften überein. In erster Linie wird die Anhäufung von Fehlern mit Softwarealterung in Verbindung gebracht, weil alte Software aufgrund von häufigen und zum Teil inkonsistenten Änderungen schlechter wartbar wird [Ga98]. Insgesamt konnten in der Literatur 54 Indikatoren für Softwarealterung identifiziert werden, von denen die sechs meist genannten in Tabelle 1 aufgeführt sind.

Kategorie	**Beschreibung**	**Anzahl Quellen**	**Exemplarische Quellen**
Ressourcen-verbrauch	Bei alter Software nimmt der Ressourcenverbrauch (insbesondere der Speicherverbrauch) zu.	30	[Ca01] [EDB08] [WHG09]
Verfügbar-keit	Die Ausfallwahrscheinlichkeit nimmt auf Grund von Fehlern zu und die Verfügbarkeit der Software nimmt ab.	28	[An11] [RD07] [ZS08]
Fehler	Im Laufe der Zeit entstehen durch häufige Änderungen mehr Fehler, wie z. B. arithmetische Rundungsfehler oder Systemfehler.	22	[SLM10] [TV02] [VT05]
Performance	Die Performance nimmt im Laufe der Zeit ab, weil z. B. der Speicher nicht korrekt freigegeben wird.	15	[CT02] [GBB02] [VG03]
Antwortzeit	Aufgrund der zunehmenden Programmkomplexität können Antworten länger dauern und fehlerhaft sein.	10	[Bo11] [Ji08] [LXJ10]
Software-qualität	Die Softwarequalität (Funktionalität, Zuverlässigkeit, Effizienz, Wartbarkeit, Übertragbarkeit) nimmt im Laufe der Zeit durch häufige und inkonsistente Änderungen ab.	10	[BF09] [Jo07] [SD00]

Tabelle 1: Die 6 meistgenannten Softwarealterungskriterien aus der Literatur

Anhand dieser Indikatoren wird deutlich, dass Softwarealterung bisher primär aus der Perspektive der Softwareentwicklung erforscht wurde. Diese eng gefasste Sichtweise ist für das IT-Management nicht ausreichend und sollte erweitert werden. Lediglich Bombardieri und Fontana versuchen, das komplexe Phänomen der Softwarealterung umfassender zu betrachten. Sie sprechen in ihrem Artikel über Softwarealterung auch Faktoren wie Kosten für Änderungen und sich verändernden Nutzeranforderungen an [BF09]. Die Symptome der Softwarealterung können vielfältig sein: fehlende Unterstützung für Dateiformate, Kommunikationsstandards und neue Hardware oder graphische Benutzeroberflächen, die nicht die Anforderungen der Nutzer erfüllen [BF09]. Dieses Verständnis von Softwarealterung verdeutlicht die Abhängigkeit zwischen Software und Umwelt. Software ist nach ihrer Entwicklung in eine bestehende Anwendungslandschaft integriert, wird in Betrieb genommen und von unterschiedlichen Personen verwendet. Insgesamt fehlen in den identifizierten Quellen Indikatoren wie steigende Entwicklungskosten oder der (negative) Einfluss einer inkonsistenten Dokumentation auf die Softwareentwicklung. Diese sind jedoch für das IT-Management relevant, da es für die Steuerung und Aufrechterhaltung der Informationsinfrastruktur unter ökonomischen Gesichtspunkten verantwortlich ist [Kr10] [HL05]. Es fehlen Ansätze, die die Ursachen und Auswirkungen von Softwarealterung frühzeitig identifizieren können und sie mess- und bewertbar machen. Diese Ansätze müssen zudem in heterogenen Anwendungslandschaften für eigenentwickelte und zugekaufte Systeme anwendbar sein.

3 Methode und Vorgehen

Die Ergebnisse der Literaturrecherche zeigen, dass Softwarealterung aus der Sicht des IT-Managements von der Forschung bisher nicht explizit behandelt wurde. Um diesem Defizit zu begegnen, sollen bisherige Erkenntnisse aus der Forschung und Praxis miteinander in Beziehung gesetzt werden. Langfristiges Ziel ist es, einen neuen systematischen Bewertungsansatz für Softwarealterung zu entwickeln, der das IT-Management in seinen Aufgaben unterstützt. Um diesem Ziel näher zu kommen wurden in einer empirischen Studie Indikatoren von Softwarealterung und potentielle Anforderungen für den Bewertungsprototypen erhoben. Als Branche wurde die Finanzindustrie ausgewählt, da diese dafür bekannt ist, sehr große Anwendungslandschaften zu betreiben, deren Systeme teilweise bereits sehr lange im Einsatz sind. Das Forschungsparadigma Design Science nach Hevner et al. [He04] bietet einen geeigneten Rahmen, um dieser Herausforderung systematisch zu begegnen. Es untergliedert den Forschungsprozess in die vier Phasen Analyse, Entwurf eines Artefaktes, dessen Evaluation sowie die Diffusion der Forschungsergebnisse. In dem vorliegenden Artikel liegt der Fokus auf der Analyse-Phase. Am Ende des gesamten iterativ gestalteten Prozesses soll ein Ergebnis stehen, welches einen Beitrag zur Forschung und Praxis liefert [He04] [Ba08]. Dem Design Science-Paradigma folgend bietet die Knowledge Base geeignete Methoden für die Datenerhebung und -auswertung an [HC10]. Nachfolgend werden die für die Datenerhebung und -auswertung ausgewählten Methoden sowie das Vorgehen bei ihrer Anwendung beschrieben.

Vorgehensweise bei der Datenerhebung: Um die Sicht des IT-Managements von Banken und Versicherungen auf das Phänomen der Softwarealterung zu erheben, wurde das leitfadengestützte Experteninterview als Methode für eine direkte und kontextbezogene Datenerhebung ausgewählt [Tr05] [MN05]. Insgesamt wurden in 27 Interviews 28 Experten[1] befragt, wovon 16 dem Schwerpunkt Banken und 12 dem Schwerpunkt Versicherungen zuzuordnen sind. Die befragten Experten sind folgenden Rollen zuzuordnen: Führungsposition (11 Personen, Vorstand, Mitglied der Geschäftsführung, Chief Information Officer, IT-Leiter), IT-Bereichsleitung (11 Personen, Teilverantwortung für einen Bereich, z. B. Unternehmensarchitektur, interne Anwendungsentwicklung), IT-Abteilungsleitung (3 Personen, Verantwortung für ein Softwareprodukt oder Entwicklungsteam), IT-Beratung (3 Personen, berät verschiedene Unternehmen und hat einen Überblick über deren IT-Management). Für die Experteninterviews wurde ein Interviewleitfaden mit offenen Fragen erstellt und iterativ weiterentwickelt [GL10] [LT09]. Für die Konstruktion des Interviewleitfadens wurden die Empfehlungen aus der Literatur berücksichtigt [SF12] [MN05] [LT09]. Die Interviewpartner erhielten den Interviewleitfaden vor den Interviewterminen zur Vorbereitung. Die Interviews wurden vor Ort in den jeweiligen Unternehmen geführt und für die Auswertung digital aufgezeichnet [LT09].

Vorgehensweise bei der Datenauswertung: Für die Auswertung der erhobenen Daten wurde die qualitative Inhaltsanalyse ausgewählt, um typische Gemeinsamkeiten sowie die Unterschiede und Besonderheiten zu analysieren [MN05]. Konkret orientierte sich die Auswertung der Experteninterviews an dem Ablaufmodel von Gläser und Laudel [GL10]. Die aufgenommenen Interviews wurden vollständig transkribiert und anonymisiert [LT09]. Die Transkription führte zu 285 Seiten Interviewmaterial, das vollständig mithilfe der Software ATLAS.ti kodiert wurde. Erste Kodes wurden deduktiv aus den Interviewleitfäden und der recherchierten Literatur entwickelt [LS07]. Neue Kodes wurden während der Kodierung induktiv erstellt und im Kodierschema ergänzt. Im Gegensatz zu der Empfehlung von Gläser und Laudel wurde ein Probedurchlauf zur Kodierung vollzogen und das Kodierschema iterativ überarbeitet [GL10]. Dabei wurden Fehler korrigiert. Bedeutungsgleiche Kodes ein weiteres Mal geprüft und teilweise zusammengefasst oder untergliedert [GL10]. Im letzten Schritt wurden die gewonnen Informationen in Hinblick auf die Forschungsfrage verdichtet [GL10].

4 Softwarealterung in der Finanzindustrie: Indikatoren und Maßnahmen

In diesem Abschnitt werden die Ergebnisse der empirischen Untersuchung beschrieben. Zunächst wird die Sicht der Praktiker auf das Phänomen der Softwarealterung zusammengefasst. Anschließend werden die in der Praxis relevanten Indikatoren für Softwarealterung beschrieben. In Anlehnung an die Literatur zum IT-Management [HL05] [Kr10] werden diese in eine technische, eine organisatorische, eine finanzielle und eine strategische Dimension eingeordnet. Abschließend werden Maßnahmen beschrieben, die die Experten für den Umgang mit Softwarealterung vorschlagen.

[1] Alle Experten waren männlich, daher wird im Folgenden nur die männliche Form verwendet.

4.1 Der Begriff Softwarealterung aus Sicht des IT-Managements

In den Interviews wurde den Befragten keine Definition zum Begriff Softwarealterung genannt. Viele der befragten Experten verstehen das Softwarealter als eine Metapher, die für nicht oder schwer wartbare und langfristig problematische Software steht. Für sie geschieht *„Alterung […] dur*ch technologische, organisatorische und fachliche *Änderungen“*. Zum einen kann Software altern, wenn diese zu oft geändert wurde und ohne geeignete Gegenmaßnahmen mit der Zeit unwartbar wird. Zum anderen führen Änderungen außerhalb der Software zum Altern. Zu diesen Änderungen gehören die eingesetzten Technologien in der Anwendungslandschaft, die fachlichen Geschäftsprozesse oder die Organisation. Unter den Experten bestand Einigkeit darin, dass eine Software nicht schlecht sein muss, nur weil diese ein hohes Alter (in Jahren) aufweist. Eine alte Software ist oftmals stabiler als eine neue Software, da viele Fehler im Verlauf der Nutzung und Weiterentwicklung korrigiert wurden. Die Experten wiesen zudem auf einen engen Zusammenhang von Softwarealter und Softwarequalität hin.

4.2 Technische Dimension

Häufige Änderungen des Quellcodes und der Architektur, z. B. durch neue fachliche Anforderungen, können zu einer **hohen Komplexität und einer schlechteren Wartbarkeit** führen. Dies tritt insbesondere dann ein, wenn die Architektur nicht flexibel für Änderungen entworfen wurde. Es ist aus Sicht der Experten eine Herausforderung, die Anforderungen nachhaltig und optimal in die vorhandene Architektur zu integrieren. Dabei weist eine neue Software noch eine klare funktionale Trennung der Module auf. Zitat: *„Architektur und Schnittstellentechniken spielen* dann auch eine Rolle. Das ist sicherlich bei Produkten, die sich weiterentwickeln, ein Problem. Dazu zählt, wie aufgeräumt und strukturiert der Code ist und ob es Redundan*zen gibt.“* Zeit- und Kostendruck führen in der Praxis jedoch schnell dazu, dass diese Trennung aufgelöst wird. Auf diese Weise verliert die Architektur nach einigen Jahren ihre Konsistenz und wird dadurch schlechter wartbar.

Die **Aktualität von Architekturkonzepten, Standards und Best Practices** zeigt, dass zuvor gewählte Ansätze als überholt gelten. Ob die Software den aktuellen Standards entspricht, erfahren die Experten durch Befragung der Entwickler. Ebenso wie sich Best Practices weiterentwickeln, entwickeln sich Programmiersprachen im Laufe der Zeit weiter. Zitat: *„Die Software ist ja in einer Programmiersprache entwickelt. In der Regel* entwickeln sich Programmiersprachen weiter. Von daher kann die verwendete Programmiersprache nicht mehr zeitgemäß sein.*“* Früher wurden vorwiegend Programmiersprachen wie z. B. COBOL, PL/I oder Smalltalk eingesetzt. Diese gelten heute als nicht mehr zeitgemäß. Insbesondere altert die Software durch die Programmiersprache, wenn die Integration innovativer softwaretechnischer Konzepte oder neuer Plattformen nicht möglich ist.

Neben der **eingesetzten Programmiersprache** haben auch die **verwendeten Technologien** einen Einfluss auf das Softwarealter. Werden z. B. Bibliotheken, Frameworks oder Hardware nicht regelmäßig geprüft und aktualisiert, können Fehler auf Grund von Inkompatibilitäten auftreten oder auch Sicherheitslücken entstehen, weil

nicht die aktuellsten Updates installiert wurden. **Auslaufende Wartungsverträge** von Basis-Plattformen und deren Austausch können die Ursache dafür sein, dass Anpassungen an der Software notwendig sind. Ohne diese Anpassungen käme es zu Fehlern und die Software wäre auf Dauer nicht mehr einsatzfähig. Die resultierenden Fehler und der eintretende Alterungsprozess können durch **fehlgeschlagene Tests** angezeigt werden. Werden diese ersten Hinweise ignoriert, ist auch eine zunehmende Fehlerhäufigkeit im operativen Betrieb der Software zu beobachten.

4.3 Organisatorische Dimension

Aus der organisatorischen Perspektive betrachtet, hat eine häufig **wechselnde Zusammensetzung des Entwicklungsteams**, etwa durch Fluktuation oder dem bevorstehenden Ruhestand einiger Teammitglieder, einen Einfluss auf die Softwarealterung. Bei der Weiterentwicklung der Software kann es dann der Fall sein, dass die **Dokumentation nicht vollständig oder konsistent** ist und Mitarbeiter wichtige Aspekte im Laufe der Zeit vergessen können. Zitat: *„Da altert die Software, weil man* Personalwechsel hat. Das ist schon ein wichtiger Faktor. Denn man kann so viel dokumentieren wie man will, denn ein wesentlicher Teil steckt immer in den Köpfen der Mitarbeiter *[...]*. Auch durch das Wechseln von Mitarbeitern gehen wichtige *Informationen verloren.“* Daher sind langjährige Mitarbeiter für den Erhalt und die Weiterentwicklung der Software wichtig. Sie kennen sich besonders gut mit der Software aus und können entsprechend der Architektur die geforderten Änderungen umsetzen. Auf der anderen Seite können diese Teammitglieder ihre Macht ausnutzen, wenn sie sich über den Wert ihres Wissens im Projekt bewusst sind und es nicht an ihre Kollegen weitergeben.

Ein weiterer wesentlicher Faktor ist die **Motivation der Mitarbeiter**. Es kann vorkommen, dass das IT-Management seine Mitarbeiter unter Druck setzt, neue Anforderungen zu integrieren oder Fehler in der alten Software schneller zu beheben. Doch die **Bearbeitungsdauer** nimmt auf Grund der sinkenden Softwarequalität zu. In der Folge können sich die Mitarbeiter weigern, die neuen Anforderungen in die Software zu integrieren, weil dies zu aufwändig ist. Es kann aber auch Mitarbeiter geben, die aus Gründen des Arbeitsplatzerhalts neue Anforderungen in die Software integrieren wollen. Projekte können sich somit zu einer politischen Angelegenheit entwickeln, in denen nicht immer die optimale Lösung für die Software gefunden wird.

Softwarealterung wird von vielen Experten mit dem **Alter der Teammitglieder** und den von ihnen **gelernten Programmiersprachen und Technologien** verbunden. Zitat: *„Außerdem sterben die Sprachen aus und wenn du keinen Entwickler mehr has*t, der diese Sprache beherrscht. Das ist so ziemlich das Schlimmste, was du dir vorstellen kannst. Du hast dann eine stillgelegte Anwendung mit ziemlich vielen Kellern, Balkonen, Anbauten und hast dann keine *Leute, die sich damit auskennen.“* Mit den älteren Mitarbeitern, die das Unternehmen verlassen, *„sterben“* die verwendeten alten Programmiersprachen aus. An den Hochschulen lernen die Studierenden vielfach nur moderne Technologien kennen und müssen nach ihrem Abschluss Schulungen im Unternehmen besuchen, um alte Programmiersprachen zu lernen. Dieses scheitert

oftmals an der fehlenden Motivation, etwas nicht Innovatives zu erlernen. Daher ist es eine Herausforderung für die Unternehmen, den Bedarf an Fachkenntnissen hinsichtlich der Programmiersprachen in Einklang mit der Personalplanung zu bringen. Die meisten Experten schätzen diese Entwicklung als problematisch ein. Sie sind der Ansicht, dass es nicht genügend Personen gibt, die alte Programmiersprachen beherrschen bzw. neu erlernen. Durch die hohe Nachfrage können diese Mitarbeiter im Vergleich etwas teurer sein.

4.4 Finanzielle Dimension

In den Interviews wurden das zur Verfügung stehende **IT-Budget und dessen Verteilung** sehr häufig thematisiert. Es wird sehr genau abgewogen, wie viel des Budgets in die Umsetzung regulatorischer Anforderungen sowie in Innovation und neue Funktionalitäten investiert werden kann. Oft bleibt nur ein geringer Anteil für den Erhalt und die Erneuerung von Software übrig. Deshalb sollten die gesamten Kosten bei der Entwicklung einer neuen Software, also die Total Cost of Ownership, betrachtet werden. Ist die Software flexibel für Änderungen entwickelt worden, so können langfristig die Betriebskosten gering bleiben. Häufig wird in der Finanzbranche ein bestimmter Anteil des **IT-Budgets für Refactorings** geplant. Jedoch muss der Mehrwert von den Fachbereichen anerkannt und dafür ausreichend Budget zur Verfügung gestellt werden. Kleinere Refactorings lassen sich oftmals im Rahmen von Wartungsverträgen durchführen. Dennoch sind die Verhandlungen von Wartungsverträgen eine Gradwanderung für das IT-Management, in denen sie Kosten und Nutzen zueinander ins Verhältnis setzen müssen.

Einige Experten aus der Bankenbranche berichteten, dass teilweise bis zu 70% des IT-Budgets für die Umsetzung von regulatorischen Anforderungen investiert wird. In alter Software lassen sich diese neuen **regulatorischen Anforderungen nicht so schnell umsetzen**. Das verbleibende Budget wird nur noch nach Kritikalität und Kosten-Nutzen-Verhältnis verteilt, womit nur die dringendsten Aufgaben abgedeckt werden können. Daher bleibt selten Budget für verjüngende Maßnahmen übrig und der Alterungsprozess kann oft nicht aufgehalten werden. Das Hauptproblem, das fast alle Befragten während der Interviews nannten, ist, dass die **Fachbereiche kein Geld für Erneuerungsprojekte oder Wartungsmaßnahmen** zur Verfügung stellen, da diese keinen Mehrwert in den Maßnahmen erkennen. Dieses ist in der Banken- und Versicherungsbranche gleichermaßen deutlich erkennbar. Wird kein Budget zur Verfügung gestellt, so wird die IT „kaputt gespart“. Um diese schwierige Situation zu vermeiden, werden zum Teil indirekt Aufwände für Wartungsmaßnahmen in neue Funktionalitäten einkalkuliert.

4.5 Strategische Dimension

Softwarealterung hat einen **Einfluss auf den Wettbewerb und die Zufriedenheit der Kunden**. Die IT-Abteilung muss mit einer angemessenen **Umsetzungsgeschwindigkeit** neue Anforderungen und Produkte kostengünstig in ihre bisherige Software integrieren können, um wettbewerbsfähig zu bleiben. Zitat: *„Den Banken ist es wi*chtig, auf Änderungen zu reagieren und diese möglichst effizient umzusetzen. Effizienz wird

letztendlich in Kosten gemessen. Wir sehen einen Wettbewerb um die Kundschaft und das gilt auch für uns. Es ist äußerst wichtig, die Änderungen kostenarm und effizient *umzusetzen."* Dies ist eine besondere Herausforderung zu Zeiten großer Dynamik und großen Drucks durch die Gesetzgebung. Anforderungen müssen schnell und kostengünstig zugleich umgesetzt werden, da sonst die Software schnell veraltet.

Ebenso ist es im Wettbewerb wichtig, die vorhandenen Kunden zu erhalten und neue zu gewinnen. Aus diesem Grund messen einige Unternehmen die **Kundenzufriedenheit** über Befragungen, die z. B. die Leistungsfähigkeit und die Benutzbarkeit der Software bewerten. Eine alte Software kann die Bedürfnisse der Kunden nicht mehr vollständig erfüllen und dadurch sinkt die Zufriedenheit. Aus der Sicht der Produktentwicklung eines IT-Herstellers kann es sogar gewünscht sein, dass Software veraltet. Dies ist aber nur der Fall, wenn neue Innovationen bereits entwickelt wurden und das Interesse der Kunden zum Kaufen geweckt werden soll. Ein weiterer Indikator besteht darin, ob für eine Software noch **neue Kunden gewonnen werden können**.

4.6 Maßnahmen gegen Softwarealterung

Die Maßnahmen, die in der Praxis eingesetzt werden, um der Softwarealterung entgegenzuwirken, können einer technologischen und einer organisatorischen Ebene zugeordnet werden. Auf der technologischen Ebene wurde vorgeschlagen, bei jeder Weiterentwicklung **Refactoring-Maßnahmen** einzuplanen, damit die Software auch für die Zukunft gut wartbar bleibt und gut getestet werden kann. Deshalb ist es von Bedeutung, Budget und Zeit für Refactorings einzuplanen. Die Softwareentwicklung kann im Rahmen von **Code Reviews** auf Verbesserungen hinweisen, wenn der Code durch zu viele Änderungen seine Struktur verloren hat. Diese Verbesserungsvorschläge können später für die Realisierung priorisiert und geplant werden. Durch Refactoring-Maßnahmen werden die Effekte von Alterung minimiert und die Weiterentwicklungskosten für die Zukunft bleiben vergleichsweise gering.

Eine weitere Strategie, die insbesondere Unternehmen aus der Versicherungsbranche anwenden, ist die **Konsolidierung von Anwendungslandschaften und das Ablösen alter Software**. Durch vorhergegangene Fusionen sollen alte und zum Teil redundante Applikationen systematisch abgeschaltet und die Anwendungslandschaft beispielsweise mittels einer Service-orientierten Architektur vereinheitlicht werden. Dadurch sollen eine bessere Übersichtlichkeit erzeugt und die Kosten für Lizenzen und Infrastruktur reduziert werden. Nach der Planung einer Soll-Anwendungslandschaft können gezielt Systeme abgeschaltet werden. Die Experten plädieren für eine **vollständige Prüfung der Fach- und Anwendungsarchitektur auf Alterungseffekte**.

Eine andere mehrfach genannte Maßnahme zur Reduzierung der Komplexität und zur Ablösung alter Software, die zum Teil umstritten ist, besteht in der **Einführung von Standardsoftware**. Als Grund für den Einsatz von Standardsoftware gaben die IT-Manager sehr oft an, dass damit die Verantwortung weitestgehend an den Softwarehersteller übertragen werden kann. Der Softwarehersteller muss seine Standardsoftware in Release-Zyklen kontinuierlich weiterentwickeln, sodass die Software fachlich und technologisch für den Kunden aktuell bleibt. Speziell in der

Bankenbranche werden die Kernbankensysteme, die eine lange Lebenserwartung besitzen, derzeit in einigen Banken durch Standardsoftware ersetzt. Die Experten versprechen sich davon langfristig eine Komplexitätsreduzierung und Investitionssicherheit. Aus der organisatorischen Perspektive ist das **Fachwissen der Mitarbeiter** für das IT-Management von besonderer Bedeutung. Die eingesetzten Technologien in den Unternehmen haben eine erwartete Lebenszeit von teilweise mehreren Jahrzehnten. Daher ist es wichtig, dass dieses Fachwissen erhalten bleibt und sogar ausgeweitet wird, indem die Mitarbeiter regelmäßig geschult werden.

In den meisten Unternehmen werden fachliche und technische Anforderungen sowie Wartungsmaßnahmen im Rahmen der **jährlichen Planungsprozesse** oder Strategieüberprüfungen besprochen. Diese münden in Projekt- und Budget-Planungen für das folgende Jahr sowie in einer strategischen Planung mit langfristigen Zielen. Diese Prozesse sollen sicherstellen, dass die Wettbewerbsfähigkeit und die Innovationskraft des Unternehmens erhalten bleiben und es nicht zu kritischen Situationen kommt, in denen mit hohen Investitionen ein Nachholbedarf ausgeglichen werden muss. Zwar wird Softwarealterung in diesen Prozessen noch nicht explizit berücksichtigt, dies wäre aber nach Meinung vieler Experten sinnvoll. Zitat: „Man muss sicher regelmäßig seine Softwarelandschaft prüfen, welche Systeme auf keinen Fall altern dürfen und bei welchen Systemen akzeptieren wir das.“ Lediglich eines der befragten Unternehmen hat bisher eine systematische Überprüfung des Zustandes der Software der gesamten Anwendungslandschaft durchgeführt. Eine systematische Wiederholung dieser Überprüfung und die wiederkehrende Nutzung der dabei gewonnenen Informationen in den Planungsprozessen wurden in diesem Fall jedoch ebenfalls nicht praktiziert.

5 Zusammenfassung und Ausblick

In den letzten 20 Jahren wurde Softwarealterung primär aus der Perspektive der Softwareentwicklung erforscht, die sich auf Indikatoren wie steigende Fehlerrate, Performanceverlust und Ausfälle beschränkt. Praxistaugliche Ansätze für die Bewertung des Softwarealters aus der Sicht des IT-Managements, welche selbst für große Anwendungslandschaften anwendbar sind, stehen bisher nicht zur Verfügung. Die Literatur beschreibt Softwarealterung als einen Prozess, in dem die Software fehleranfälliger wird und daraufhin Abstürze und Ausfälle auftreten. Dagegen sehen die befragten Personen aus dem IT-Management die Ursache von Softwarealterung in der Veränderung der Software selbst, die unter anderem durch die Integration neuer Anforderungen und der damit einhergehenden steigenden Komplexität verursacht wird. Zugleich hat auch die Veränderung der Umwelt einen Einfluss auf die Softwarealterung, da Software nicht isoliert zu betrachten ist, sondern in einem komplexen System aus Abhängigkeiten zu sich ändernden Anforderungen, anderen Anwendungen, Mitarbeitern, Kunden und Lieferanten eingebettet ist.

Trotz der großen Bedeutung wird das Alter von Software in den befragten Unternehmen bisher nicht explizit regelmäßig bewertet. Eine professionelle Strategie zur Planung und Umsetzung von Gegenmaßnahmen zur Softwarealterung ist bisher nicht etabliert. Die

Strategien, wie mit Softwarealterung umgegangen wird, unterscheiden sich dabei zwischen Literatur und Empirie deutlich. Die Ansätze der Softwareverjüngung aus der Literatur sind sehr kurzfristig, wie z. B. ein Neustart der Software, und beziehen sich überwiegend nur auf einzelne Applikationen. Dagegen beschreibt das befragte IT-Management Strategien, die die Effekte von Softwarealterung langfristig beheben sollen, wie z. B. die Auswahl von Software für regelmäßige Refactorings, die Konsolidierung der Anwendungslandschaft, Weiterbildungsangebote für Mitarbeiter oder auch die eine Veränderung der Prozesse in der Budget- und Jahresplanung. Zukünftig müssen neue Ansätze entwickelt werden, welche die verschiedenen Dimensionen der Softwarealterung berücksichtigen und eine bessere Entscheidungsgrundlage für das IT-Management bieten.

Literaturverzeichnis

[An11] Andrade, E. C. et al.: Modeling and Analyzing Server System with Rejuvenation through SysML and Stochastic Reward Nets. In: Proc. ARES 2011, 2011; S. 161-168.

[Ba08] Baskerville, R.: What design science is not. European Journal of Information Systems (17:5), 2008; S. 441-443.

[BF09] Bombardieri, M.; Fontana, F.: Software aging assessment through a specialization of the SQuaRE quality model. In: Proc. WOSQ '09, ICSE Workshop, 2009; S. 33-38.

[Bo11] Bovenzi, A. et al.: Workload Characterization for Software Aging Analysis. In: Proc. ISSRE 2011, Hiroshima, 2011; S. 240-249.

[Br13] Brink, A.: Anfertigung wissenschaftlicher Arbeiten: Ein prozessorientierter Leitfaden zur Erstellung von Bachelor-, Master- und Diplomarbeiten. Springer Gabler, Wiesbaden, 2013.

[Ca01] Castelli, V. et al.: Proactive Management of Software Aging. IBM Journal of Research and Development (45:2), 2001; S. 311-332.

[Co11] Cotroneo, D. et al.: Software Aging und Rejuvenation: Where We Are and Where We Are Going. In: Proc. 3rd Int. Workshop on Software Aging and Rejuvenation, 2011; S. 1-6.

[CT02] Calzarossa, M. C.; Tucci, S. (Hrsg.): Performance Evaluation of Complex Systems: Techniques and Tools. Springer, Berlin, 2002.

[EDB08] El-Shishiny, H.; Deraz, S.; Bahy, O.: Mining Software Aging Patterns by Artificial Neural Networks. In (Marinai, S.; Schwenker, F. Hrsg.): Artificial Neural Networks in Pattern Recognition. Springer, Berlin, 2008; S. 252-262.

[Ga98] Garg, S. et al.: A methodology for detection und estimation of software aging, In: Proc. 9th Int. Symp. on Software Reliability, 1998; S. 283-292.

[GBB02] Gross, K. C.; Bhardwaj, V.; Bickford, R.: Proactive detection of software aging mechanisms in performance critical computers. In: Proc. 27th Annual NASA Software Engineering Workshop, 2002; S. 17-23.

[GL10] Gläser, J.; Laudel, G.: Experteninterviews und qualitative Inhaltsanalyse: Als Instrumente rekonstruierender Untersuchungen. VS Verlag für Sozialwissenschaften, Wiesbaden, 2010.

[Ji08] Jia, Y.-F. et al.: On the Relationship between Software Aging and Related Parameters. In: Proc. QSIC '08, Oxford, 2008; S. 241-246.

[HC10] Hevner, A.; Chatterjee, S. (Hrsg.): Design Research in Information Systems: Theory and Practice. Springer, Berlin, 2010.

[He04] Hevner, A. R. et al.: Design Science in Information Systems Research. MIS Quarterly (28:1), 2004; S. 75-105.

[HL05] Heinrich, L. J.; Lehner, F.: Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur. Oldenbourg, München, 2005.
[Hu95] Huang, Y. et al.: Software rejuvenation: analysis, module und applications. In: Proc. 25th Int. Symp. on Fault-Tolerant Computing, 1995; S. 381-390.
[Jo07] Jones, C.: Geriatric Issues of Aging Software. In: Crosstalk, The Journal of Defense Software Engineering (20:12), 2007.
[Kr10] Krcmar, H.: Informationsmanagement. Springer, Berlin, 2010.
[KW06] Knackstedt, R.; Winkelmann, A.: Online-Literaturdatenbanken im Bereich der Wirtschaftsinformatik. Wirtschaftsinformatik (48:1), 2006; S. 47-59.
[LS07] Lewins, A.; Silver, C.: Using software in qualitative research: A step-by-step guide. Sage, Los Angeles, 2007.
[LT09] Liebold, R.; Trinczek, R.: Experteninterview. In (Kühl, S. Hrsg.): Handbuch Methoden der Organisationsforschung. VS Verlag für Sozialwissenschaften, Wiesbaden, 2009; S. 32-56.
[LXJ10] Lei, X.; Xue, K.; Jia, Y.: A comprehensive metric to evaluating software aging. In: Proc. ICCASM 2010, Taiyuan, 2010; S. V11-518-V11-520.
[MN05] Meuser, M.; Nagel, U.: ExpertInneninterviews - vielfach erprobt, wenig bedacht. In (Bogner, A. Hrsg.): Das Experteninterview : Theorie, Methode, Anwendung. VS, Verlag für Sozialwissenschaften, Wiesbaden, 2005; S. 71-93.
[Pa94] Parnas, D.: Software Aging. In: Proc. Int. Conf. on Software Engineering, 1994; S. 279-287.
[RD07] Rinsaka, K.; Dohi, T.: A Faster Estimation Algorithm for Periodic Preventive Rejuvenation Schedule Maximizing System Availability. In (Malek, M. et al. Hrsg.): Service Availability, Springer, Berlin, 2007; S. 94-109.
[SD00] Swanson, E. B.; Dans, E.: System Life Expectancy and the Maintenance Effort: Exploring Their Equilibration. MIS Quarterly (24:2), 2000; S. 277-297.
[SF12] Stigler, H.; Felbinger, G.: Der Interviewleitfaden im qualitativen Interview. In (Stigler, H.; Reicher, H., Hrsg.): Praxisbuch empirische Sozialforschung in den Erziehungs- und Bildungswissenschaften. Studien-Verlag, Innsbruck, 2012; S. 129-134.
[SLM10] Salfner, F.; Lenk, M.; Malek, M.: A survey of online failure prediction methods. ACM Comput. Surv. (42:3), 2010; S. 1-42.
[Tr05] Trinczek, R.: Wie befrage ich Manager? In (Bogner, A. Hrsg.): Das Experteninterview: Theorie, Methode, Anwendung. VS, Verlag für Sozialwissenschaften, Wiesbaden, 2005; S. 225-238.
[TV02] Trivedi, K. S.; Vaidyanathan, K.: Software Reliability and Rejuvenation: Modeling and Analysis. In (Calzarossa, M. C.; Salvatore, T. Hrsg.): Performance Evaluation of Complex Systems: Techniques and Tools. Springer, Berlin, 2002; S. 318-345.
[VG03] Vaidyanathan, K.; Gross, K.: MSET Performance Optimization for Detection of Software Aging. In: Proc. 14th IEEE Int. Symp. on Software Reliability Engineering, 2003.
[VT05] Vaidyanathan, K.; Trivedi, K.: A Comprehensive Model for Software Rejuvenation. In: IEEE Transact. on Dependable and Secure Computing (2:2), 2005; S. 124-137.
[WHG09] Wu, Q. E.; Han, Z. Y.; Guo, T. S.: Application of an Uncertain Reasoning Approach to Software Aging Detection. In: Proc. NCM '09, Seoul, 2009, S. 1592-1597.
[ZS08] Zhao, L.; Song, Q.: Availability and Cost Analysis of a Fault-Tolerant Software System with Rejuvenation. In: Proc. ICACTE '08, 2008; S. 261-265.