Harry M. Sneed · Richard Seidl

Softwareevolution

Erhaltung und Fortschreibung bestehender Softwaresysteme

dpunkt.verlag

Harry M. Sneed · Harry.Sneed@t-online.de
Richard Seidl · office@richard-seidl.com

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Fachliche Beratung und Herausgabe von dpunkt.büchern im Bereich Wirtschaftsinformatik:
Prof. Dr. Heidi Heilmann · heidi.heilmann@augustinum.net

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN 978-3-86490-041-9

1. Auflage 2013

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Vorwort

Der Begriff »*Softwareevolution*« impliziert alle Handlungen, die auf ein Softwaresystem ab dem Zeitpunkt wirken, ab dem es produktiv eingesetzt wird. Das System wird u.a. korrigiert, geändert, erweitert, optimiert, saniert, dokumentiert und integriert. Es bleibt jedoch im Kern immer das gleiche System, auch wenn sich das Verhalten ändert. Wenn es sich jedoch so weit wandelt, dass es sich ganz anders verhält, handelt es sich nicht mehr um Evolution, sondern um eine Weiterentwicklung und dafür gelten andere Gesetze. Die Migration des Systems in eine andere technische Umgebung bzw. die Transformation des Codes in eine andere Sprache dürfte kein anderes Verhalten verursachen, ist aber trotzdem nicht als Evolution zu betrachten, denn was dabei herauskommt, ist ein anderes Produkt.

Neuentwicklung und Migration sind einmalige große Anstrengungen, um ein System in seiner Gesamtheit neu zu schaffen oder zu verlagern. Evolution bedeutet hingegen eine Reihe begrenzter Aktionen, um das bestehende System in seiner ursprünglichen Umgebung ständig zu verbessern. Softwareevolution ist mit Softwarewartung eng verwandt. Beide Begriffe beziehen sich auf die gleichen Grundaktivitäten. Der Unterschied besteht darin, dass Evolution der umfassendere Begriff ist. Viele haben unter »Wartung« (Maintenance) nur die Korrektur und Anpassung der Software verstanden. Mit dem Begriff »Evolution« ist eindeutig auch die Weiterentwicklung des Systems gemeint, also die fortdauernde Erweiterung durch neue Funktionen. Dies zu unterstreichen war der Hauptgrund für die Verwendung dieses Begriffs.

Seitdem die Softwareentwicklung agil geworden ist, besteht die Notwendigkeit, Evolution auch von der Entwicklung abzugrenzen. Wenn die Entwicklung sich inkrementell in kleinen Schritten vollzieht, worin liegt dann der Unterschied zur Evolution, bei der das Produkt auch in kleinen Schritten ständig verbessert wird? Auf diese Frage kann es nur eine Antwort geben, und zwar, dass bei der Evolution das Produkt produktiv im Einsatz ist. Bei einer agilen Entwicklung dürfte dies noch nicht der Fall sein, und wenn es doch der Fall ist, dann ist es keine Entwicklung mehr, sondern Evolution.

Dies dürfte für den Leser wie ein Streit um des Kaisers Bart erscheinen, aber diese Begriffsunterscheidung ist für das Verständnis des vorliegenden Buches wichtig. Bei der Softwareevolution herrschen ganz andere Gesetze als bei der Softwareentwicklung. Im Vordergrund steht die Aufrechterhaltung der laufenden Dienstleistung. Das System, das im Einsatz ist, darf in keiner Weise in seiner Leistung beeinträchtigt werden. Wenn sein Verhalten sich ändert, dann nur geringfügig und zum Besseren. Das Gleiche trifft auf seine Konstruktionsweise zu. Kurzum, die Erhaltung der bestehenden Funktionalität hat Vorrang vor der Entstehung neuer Funktionalität. Das Ziel eines agilen Entwicklungsprojekts ist es, neue Funktionalität zu erschaffen, auch wenn es erforderlich ist, die bisher geschaffene Funktionalität zu bewahren. Die letztere ist nicht so wichtig, weil die Software noch nicht im Einsatz ist. Im Falle der Evolution ist es umgekehrt. Das Hauptziel ist die Erhaltung der bestehenden Software, auch wenn es erforderlich ist, neue Funktionalität zu schaffen.

Im Lehrfach Software Engineering an den Hochschulen und Bildungsanstalten steht die Entwicklung neuer Systeme im Mittelpunkt. Wenn überhaupt, wird der Umgang mit bestehenden Systemen nur am Rande behandelt. Diese Vernachlässigung steht im krassen Widerspruch zu dem Bedarf an Wartungspersonal in der Industrie. Laut verschiedenen Schätzungen sind über 70 % des Entwicklungspersonals mit der Erhaltung bestehender Systeme beschäftigt – Systeme, die im Durchschnitt 7 Jahre im Einsatz sind. Es versteht sich von selbst, dass die Nachbesserung und Fortschreibung solcher Systeme andere Kenntnisse verlangen als die Entwicklung neuer Systeme. Nicht nur die Vorgehensweise, sondern auch die Programmier- und Testtechniken sind verschieden.

Der Bedarf an Wartungspersonal wäre gar nicht so akut, wenn die Systeme so bleiben würden, wie sie sind, aber Softwaresysteme sind ein Spiegelbild der betrieblichen Wirklichkeit und diese ändert sich immer schneller, laut einer Untersuchung von Les Hatton [Hatt07] um 10 bis 16 % jährlich. Das heißt, bei einem System mit 200.000 Anweisungen werden jährlich mindestens 20.000 Anweisungen geändert oder hinzugefügt. Die mittlere Änderungsproduktivität liegt unter 400 Anweisungen pro Personenmonat. Dennoch kann ein Systemerhalter maximal 5.000 Anweisungen pro Jahr verändern bzw. 50.000 Anweisungen bzw. 75.000 Codezeilen betreuen. Sämtliche Untersuchungen zum Thema Softwarewartung kommen zum Schluss, dass eine Person jährlich nicht mehr als 50.000 Anweisungen bei einer jährlichen Änderungsrate von 10 % betreuen kann. Leider wird in vielen Anwenderbetrieben das Wartungspersonal gezwungen, bis zu 100.000 Anweisungen zu betreuen. Verantwortliche IT-Manager müssen erkennen, wo die Grenzen der Technologie liegen.

Tatsache ist, dass immer mehr neue IT-Systeme in die Welt gesetzt werden, ohne dass die alten Systeme aus dem Verkehr gezogen werden. Einzelne Systeme werden abgelöst, aber die Mehrzahl alter Systeme bleibt in Betrieb, weil irgendwelche Benutzer sie noch benötigen. Also müssen sie gepflegt und angepasst wer-

den. Das bindet Personal. Wenn allein in Deutschland jährlich 6.000 IT-Anwendungen mit einer mittleren Größe von 100.000 Anweisungen freigegeben werden, schafft das einen Bedarf an 12.000 zusätzlichen Wartungstechnikern. Diese Wartungstechniker müssten wissen, wie man Fehler findet oder korrigiert, wie man Code ändert ohne unerwünschte Begleiterscheinungen, wie man neue Funktionalität einbaut und wie man Systeme saniert, ohne ihr Verhalten zu beeinträchtigen.

Daraus ergibt sich ein beträchtlicher Ausbildungsbedarf für Softwareerhaltungsspezialisten. Ob unter dem Stichwort »Softwarewartung« oder »Softwareevolution«, junge Menschen müssen mehr im Umgang mit alter Software ausgebildet werden. Neben den Kursen in Modellierung und Entwicklung von Software sollten auch Kurse in der Erhaltung von Software angeboten werden. Gemessen an dem industriellen Bedarf müssen die Erhaltungskurse sogar den Vorrang haben. Dieses Buch ist einerseits Begleitmaterial zu einem Kurs in Softwareevolution und anderseits als ein Nachschlagwerk für praktizierende Softwareerhalter gedacht. Es soll darüber hinaus auch dazu beitragen, dass dieses wichtige Thema die Aufmerksamkeit erhält, die ihm zusteht. Die Entwicklung neuer Systeme kann man jederzeit einstellen. Die Erhaltung der bestehenden Systeme muss weitergehen. Dafür werden gezielt ausgebildete Spezialisten gebraucht.

In diesem Buch wird auf die wesentlichen Aktivitäten der Softwareevolution eingegangen – Fehlerbehebung, Änderung, Sanierung und Erweiterung sowie Regressionstesten und Nachdokumentation. Diese Aktivitäten sind in den meisten Lehrplänen der Informatik nicht ausreichend beachtet. Möglicherweise liegt es daran, dass es so schwierig ist, sie abzugrenzen und zu lehren. Das müsste sich ändern. Statt Spezifikationen für Entwicklungsaufgaben sollten Studenten fertige Programme in einem schlechten Zustand bekommen und aufgefordert werden, diese zu bearbeiten. Sie sollten den Code korrigieren, sanieren, dokumentieren, ändern und erweitern. Wie sie dabei vorzugehen haben, wird hier geschildert. Außerdem wird auf die Gesetze und die Wirtschaftlichkeit der Softwareevolution eingegangen.

Das Buch ist aber nicht nur für Studierende gedacht, sondern auch als Unterstützung für die praktische Arbeit. Die Rolle des Wartungsprogrammierers wird hier aufgewertet. Es sind schließlich diese Programmierer, die den IT-Betrieb in Gang halten. Sie haben jeden Grund, darauf stolz zu sein. Diese Lektüre sollte dazu beitragen, mehr Professionalität in ihre alltäglichen Tätigkeiten einzubringen. Die Voraussetzung dafür ist, dass sie die Bedeutung ihrer Tätigkeit verstehen und vertreten können. Auch das wird von diesem Buch befördert.

Zum Schluss ist zu sagen, dass der Stoff, der hier zusammengefasst ist, auf mehr als 30 Jahre Erfahrung in der Bearbeitung bestehender Software basiert. Der Umgang mit Legacy-Systemen war schon immer eine große Herausforderung und wird es in der Zukunft auch bleiben. Schon vor 20 Jahren habe ich in einem Artikel für die Computerwoche proklamiert: »Die größte Herausforderung der Zukunft ist die Bewältigung der Vergangenheit.« Diese Aussage gilt nach wie vor.

Dieses Buch ist dem Vater der Softwareevolution, Manny Lehman, gewidmet, der vor Kurzem in Jerusalem verstorben ist. Er hat zusammen mit Les Belady erkannt, dass komplexe Softwaresysteme das Produkt eines langen Evolutionsprozesses sind. Ich hatte das Privileg, beide als meine Gäste auf der Internationalen Software Maintenance Conference 2005 in Budapest begrüßen zu dürfen.

Das war kurz nachdem ich meinen Koautor Richard Seidl kennengelernt habe. Wir arbeiteten zusammen in einem großen Testprojekt in Dresden. Seit damals haben wir bei verschiedenen Buchprojekten zusammengearbeitet: Bücher über das Testen, das Vermessen und jetzt über Softwareevolution. Ich bin sehr froh, Herrn Seidl als Koautor für dieses Buchprojekt gewonnen zu haben. Ohne seinen wertvollen Beitrag wäre dieses Buch nie in dieser Form erschienen.

Harry Sneed
Wien, im Juli 2013