IN DIESEM KAPITEL

Kapitel 1
Software ist Code

In diesem Kapitel werden, ausgehend von der Bedeutung, die Software im Alltag spielt, Probleme erläutert, die bei der Softwareentwicklung immer wieder auftreten. Danach wird die zentrale Stellung des Programmcodes im Entwicklungsprozess begründet.

Erwartungen an Software

Software ist aus dem täglichen Leben nicht mehr wegzudenken. Sie findet sich nicht nur in Computern und Smartphones, sondern ersetzt zunehmend sogar einen Teil der mechanischen Bauteile, beispielsweise in Türschlössern und Nähmaschinen. Vor allem die aufwendig herzustellenden Steuerungskonstruktionen weichen elektronisch angesteuerten einfachen Schaltelementen.

Den meisten Benutzern ist es dabei völlig gleichgültig, was Software ist. Sie erwarten, dass das Gerät oder das Programm funktioniert.

Allerdings ist niemand total überrascht, wenn das einmal nicht der Fall ist. Überlegen Sie kurz, wie oft Sie selbst schon schulterzuckend einen Softwarefehler hinnehmen mussten. Meist ist das zum Glück nur ärgerlich.

Es scheint unmöglich zu sein, weitgehend fehlerfreie Software zu produzieren.

Der Vergleich mit anderen komplexen technischen Apparaten wirft allerdings die Frage auf, wieso es bisher nicht gelungen ist, Verbesserungen in vergleichbarem Maße zu erreichen wie etwa bei der Sicherheit von Flugzeugen. Darauf gibt es keine einfache Antwort.

Die beschriebene Problematik ist fast so alt wie der Computer. Bereits 1972 erwuchs aus der unbefriedigenden Situation der Begriff *Softwarekrise*, und 1993 wurde auf einer Tagung gefragt: »Kann eine Krise 20 Jahre dauern?«

Trotz unleugbarer Fortschritte muss man eingestehen, dass die Krisensymptome heute noch genauso sichtbar sind wie damals. Geändert hat sich die Wahrnehmung. Das, was zunächst als Krise erschien, wird mittlerweile als Dauerzustand hingenommen.

Das heißt allerdings nicht, dass Sie als Entwickler diesen Zustand akzeptieren müssen. Ich hoffe, Sie lesen dieses Buch gerade deshalb, um daran etwas zu ändern. Denn Veränderung tut not, und zwar immer dringender. Je mehr Software in unser Leben eingreift, desto schwerwiegender werden ganz zwangsläufig die Auswirkungen von Fehlern sein.

Softwarekrise

In den Anfangsjahren der Computernutzung entfiel der Löwenanteil der Kosten auf die Hardware. Programmiert wurden vor allem gut aufbereitete und verstandene mathematische Algorithmen. Bezeichnenderweise war die erste Programmiersprache im heutigen Sinn das 1957 erschienene FORTRAN, kurz für FORmula TRANslator. Über Methoden zur Softwareentwicklung dachte niemand intensiver nach.

Der technische Fortschritt führte jedoch schnell zu sinkenden Hardwarekosten und in der Folge zu einer stärkeren Verbreitung von Computern. Die Aufgaben wurden anspruchsvoller und komplizierter. Dadurch stiegen die Kosten der Programmierung und begannen Mitte der 1960er-Jahre diejenigen der Hardware zu übersteigen.

Gleichzeitig kam es zu ersten spektakulären Folgen von Programmierfehlern wie 1962 dem Verlust der Venussonde Mariner-1. Es wurde unübersehbar, dass mangelhafte Software gewaltige Kosten verursachen kann. So wie bisher konnte es nicht weitergehen.

Die Entwicklung war in eine Krise, die Softwarekrise, geraten. Auf der Suche nach einem Ausweg entstand das *Softwareengineering*. Dieser Begriff tauchte zum ersten Mal 1968 im Namen einer NATO-Tagung auf.

Trotz entscheidender Verbesserungen kann die Softwarekrise bis heute nicht als tatsächlich überwunden gesehen werden.

Probleme haben Ursachen

Wenn man auf Probleme stößt, ist es erfahrungsgemäß nützlich, deren Ursachen aufzuklären. In den meisten Fällen gibt es dafür nicht nur einen Grund. Für die Softwarekrise lassen sich zwei wichtige Ursachengruppen finden.

Allgemeine Ursachen

Für die oft mangelhafte Qualität von Software gibt es ebenso wie für das ganze oder teilweise Scheitern von Entwicklungsprojekten verschiedene Gründe, unter anderem:

- ✔ Ungenügende Vorbereitung, insbesondere unvollständige Aufgabenbeschreibung und Abgrenzung
- ✔ Fehler in der Projektorganisation und der Projektleitung
- ✔ Unverständnis oder Unkenntnis im Management bezüglich der Besonderheiten der Softwareentwicklung
- ✔ Unzureichende Mittel (Zeit und Geld)
- ✔ Vernachlässigung der Qualitätssicherung (Test)
- ✔ Schwierigkeiten bei der Beherrschung der Komplexität der entstehenden Software

Diese Liste ist bei Weitem nicht vollständig. Sie soll Ihnen nur zeigen, dass es zwar viele aus Sicht der Entwicklung externe Ursachen gibt, aber zumindest die letzten beiden Punkte fallen in die Verantwortung der Entwickler.

Tests sind unverzichtbarer Teil jeder professionellen Softwareentwicklung. Sie können damit zwar nicht die vollständige Korrektheit Ihres Programms beweisen – ein Test ist immer nur eine Stichprobe –, aber Sie dokumentieren, was nachweisbar funktioniert, und verringern die Wahrscheinlichkeit, dass Fehler übersehen werden.

Dieser Hinweis steht seiner Wichtigkeit wegen bereits hier und damit ganz am Anfang. Nehmen Sie ihn bitte ernst und vergessen Sie nicht: Auch Tests sind Code, und für diesen Testcode sind die gleichen Qualitätsanforderungen zu erfüllen wie für den zu testenden Code.

Hardwareentwicklung

Einen sehr großen Einfluss auf die Softwarekrise hat die Entwicklung der Hardware ausgeübt. Durch die rapide Leistungssteigerung der Prozessoren und die parallel dazu gewachsenen verfügbaren Speichervolumina entstand ein stetiger Druck, noch größere Aufgaben in Angriff zu nehmen.

Die Softwarehersteller hatten nie Zeit, um in Ruhe Luft zu holen und ihre Produkte zu konsolidieren. Stets stand schon eine neue, noch leistungsfähigere Hardwaregeneration vor der Tür, die Anpassungen und Weiterentwicklungen verlangte. Die Programme wurden immer umfangreicher und komplexer, ohne dass mehr Zeit für die Entwicklung verfügbar war.

Alle im Entwicklungsprozess erreichten Verbesserungen wurden sofort dafür eingesetzt, in diesem Rennen Schritt zu halten. Die Softwarequalität geriet dabei oft ins Hintertreffen. Schon in den 1980er-Jahren entstand dazu das wahlweise Niklaus Wirth oder Martin Reiser zugeschriebene Bonmot: Software wird schneller langsamer, als die Hardware schneller wird.

Mittlerweile steigt die reine Prozessorgeschwindigkeit kaum noch, dafür werden die Strukturen mit Mehrkern- und Multiprozessoren komplizierter. Diese neuen Möglichkeiten können nur durch nebenläufige Programme voll ausgenutzt werden, was die Software noch einmal um Größenordnungen komplexer macht. Die Lage hat sich also kaum verändert.

Nichts ohne Code

Bisher habe ich ständig von Software gesprochen, ohne diesen Begriff genauer zu erläutern. Das hat seinen Grund. Es ist schwer, eine Definition zu finden, die sowohl ausreichend konkret als auch umfassend genug ist. Hier werde ich mich deshalb auf einen wesentlichen Aspekt beschränken.

Software ist in gewisser Weise ein indirektes Produkt. Sie entsteht aus formalen Beschreibungen, die im Allgemeinen *Programmcode* oder kurz *Code* genannt werden. Ihre Wirksamkeit oder Nutzbarkeit ist an bestimmte technische Geräte – die Hardware – und weitere Software wie Betriebssysteme und Compiler gebunden.

Wenn Sie von dieser Umgebung, die gemeinhin vorgegeben ist, absehen, lassen sich alle wichtigen Eigenschaften einer Software auf ihren Code zurückführen. Um eine Software zu schreiben oder zu erweitern, müssen Sie Code schreiben. Das Gleiche gilt, wenn Sie einen Fehler beheben wollen.

Das Produkt jedes Softwareentwicklungsprozesses ist Code. Es gibt gemeinhin keine genauere Beschreibung dessen, was eine Software leistet, als den Programmcode.

Die Schlussfolgerung lautet daher: Software besteht im Wesentlichen aus Code. Und daraus folgt unmittelbar: Code ist wichtig. Bessere Software lässt sich nur durch besseren Code produzieren.

Es wird Sie daher nicht überraschen, dass in vielen Unternehmen schon heute ein erheblicher Teil der Vermögenswerte aus Code besteht. Dessen wirtschaftliche Bedeutung zeigt sich nicht zuletzt in den mit viel Einsatz geführten Auseinandersetzungen um Rechtsverstöße und Lizenzbedingungen.

Da immer mehr Funktionen durch Software realisiert werden, ist absehbar, dass die Wichtigkeit von Code für unser Leben weiter zunehmen wird.

Code existiert oft wesentlich länger als die Hardware, für die er ursprünglich entwickelt wurde. Beispielsweise verwenden manche Forschungseinrichtungen noch heute mathematische Fortran-Bibliotheken, die bereits mehrere Jahrzehnte alt sind. In vielen großen Programmsystemen finden sich Codeteile, die aus weit zurückliegenden Versionen überlebt haben.

Voraussetzung eines solch langen Lebens ist lediglich, dass neuere Versionen der Compiler abwärtskompatibel sind, das heißt, dass sie den alten Quelltext weiterhin akzeptieren und fehlerfrei verarbeiten können. Wenn ein genügend großer Codebestand vorhanden ist, wird das von den Compilerproduzenten gewöhnlich gewährleistet, um die Kunden schneller auf neue Versionen locken zu können.

Ganz nebenbei: Die Angst, dass die für eine solche langfristige Sicherung der Codebasis notwenige kritische Masse an Programmcode nicht zustande kommt, ist eines der größten Hindernisse bei der Verbreitung neuer Programmiersprachen.

Das Wichtigste in Kürze

- ✔ Software ist zu einem unverzichtbaren Bestandteil des Alltagslebens geworden.
- ✔ Unzureichende Softwarequalität und fehlgeschlagene Entwicklungsprojekte sind seit Langem ein Dauerthema.
- ✔ Trotz aller Bemühungen hat die Softwareentwicklung mit der rasanten Verbesserung der Hardware nur mühsam mithalten können.
- ✔ Die schnell fortschreitende Digitalisierung verlangt erhebliche Verbesserungen der Softwarequalität.
- ✔ Software ist vor allem der Code, der die von Ihnen gewünschte Funktion definiert und aus dem lauffähige Systeme erzeugt werden.
- ✔ Code ist wichtig!