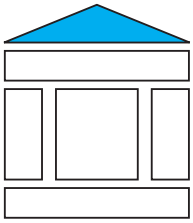
2 Warum ist Software so schwer zu entwickeln?



Frage

Seit Software entwickelt wird, beklagt man sich darüber, wie schwierig diese Tätigkeit ist. Immer wieder ist von der »Software-Krise« die Rede. Um diese Schwierigkeiten zu verstehen, muss man sich zunächst die Charakteristika von Software näher ansehen.

Außerdem muss man analysieren, wie sich Software in den letzten 30 Jahren entwickelt hat, um zu begreifen, dass, trotz wesentlicher Fortschritte in der Softwaretechnik, noch viel zu tun ist.

Überlegen Sie, durch welche Charakteristika sich Software von anderen technischen Produkten unterscheidet.

Antwort

Software unterscheidet sich durch folgende Charakteristika von anderen technischen Produkten:

- Software ist ein immaterielles Produkt.
  Software kann man nicht »anfassen«, nicht »sehen«. Das, was die Dokumentation beschreibt, stimmt oft nicht hundertprozentig mit dem lauffähigen Code der Software überein. Diese Immaterialität hat massive Konsequenzen für die Softwaretechnik und insbesondere für das Softwaremanagement. Der Entwicklungsfortschritt ist nicht objektiv zu ermitteln.
- Software unterliegt *keinem* Verschleiß.
  Technische Produkte nutzen sich im Laufe der Zeit ab. Software kann beliebig oft ablaufen, ohne dass Abnutzungserscheinungen auftreten. Daher ist eine klassische Wartung von Software, bei der es darum ginge, verschlissene Teile auszutauschen, nicht notwendig.
- Software altert.
  Auf den ersten Blick erscheint die Aussage unglaubwürdig, zumal gerade festgestellt wurde, dass Software keinem Verschleiß unterliegt. Beachtet man jedoch, dass sich die Umgebung, in der eine Software eingesetzt wird, ständig ändert, dann wird diese Aussage verständlich. Software, die nicht ständig an die sich ändernde Umgebung angepasst wird, altert und ist irgendwann veraltet, d. h. sie kann nicht mehr für den ursprünglich vorgesehenen Zweck eingesetzt werden. Beispielsweise läuft sie auf einer neuen Hardware-Plattform nicht mehr oder sie kann nicht über das Web benutzt werden.
- Für Software gibt es *keine* Ersatzteile.

Bei technischen Produkten werden Produktteile, die verschlissen oder fehlerhaft sind, ausgetauscht und durch in der Regel vorproduzierte Ersatzteile ersetzt. Für Software geht dies nicht. Es würde bedeuten, Software-Komponenten mehrfach zu entwickeln. Tritt ein Fehler in einer Komponente auf, dann müsste man darauf hoffen, dass in der ausgetauschten Komponente nicht zufällig derselbe Fehler auftaucht. Außerdem müssten die Schnittstellen und ihre Semantik exakt definiert sein. Experimente haben gezeigt, dass Entwicklungsteams, die Komponenten völlig unabhängig voneinander entwickeln, zwar nicht die exakt gleichen Fehler machen, aber ähnliche Probleme erzeugen. Das liegt daran, dass der Mensch aufgrund seiner mentalen Fähigkeiten dazu neigt, in bestimmte Fehlerfallen zu »tappen«.

- Software ist im Allgemeinen leichter und schneller änderbar als ein technisches Produkt.
  Um technische Produkte zu ändern, müssen oft erst neue Werkzeuge hergestellt werden. Ist Software gut strukturiert und modularisiert, dann können Änderungen schnell und einfach durchgeführt werden.
- Software ist schwer zu »vermessen«.
  Technische Produkte kann man mit Hilfe von Messgeräten in der Regel sehr exakt vermessen. Anhand der Maße können die Produkte dann sowohl untereinander als auch mit einem Standard oder einer Vorgabe verglichen werden. Das ist bei Software nur bedingt möglich. Erstens ist die Qualität für Software schwer definierbar und quantifizierbar. Zweitens ist die Korrelation zwischen dem, was man messen kann, und dem, was als Qualität gefordert ist, nicht oder nur in Ansätzen bekannt.

Veränderungen

Software hat sich in den letzten 20 Jahren in verschiedener Hinsicht gravierend verändert, sodass die Fortschritte in der Softwaretechnik *nicht* ausreichten, um mit den Veränderungen der Software Schritt zu halten.

Frage

Wenn Sie sich schon längere Zeit mit der Softwaretechnik befassen, welche Änderungen stellen Sie fest?

Antwort

Die wesentlichen Veränderungen sind:

- Zunehmende Bedeutung
- Wachsende Komplexität
- Zunehmende Qualitätsanforderungen
- Nachfragestau und Engpassfaktor
- Mehr Standardsoftware
- Zunehmend »Altlasten«
- Zunehmend »Außer-Haus«-Entwicklung

Diese Veränderungen sind nicht isoliert voneinander zu sehen, sondern sie stehen in Wechselwirkung und bedingen sich teilweise gegenseitig. Im Folgenden werden diese Veränderungen etwas näher betrachtet.

Zunehmende Bedeutung

Die große Bedeutung der Software verdeutlicht folgendes Zitat: Zitat

»Software ist der fundamentale *Werkstoff des Informationszeitalters*. Innovative Produkte und Dienstleistungen sind ohne Software nicht mehr denkbar. Die Wettbewerbsfähigkeit der deutschen Wirtschaft hängt entscheidend von der Fähigkeit ab, Software-intensive Produkte und Dienstleistungen höchster Qualität zu erstellen. [...] Software wird in der Zukunft *integrierter* – in vielen Fällen sogar *dominierender – Teil großer komplexer Systeme* sein« [BJN+06, S. 210].

Die Abb. 2.0-1 zeigt als Beispiel den zunehmenden Anteil der Software im Automobilbereich. In einem PKW gehobener Ausstattung betrug im Jahr 2003 die Größe eingebetteter Software 70 MB, bis zum Jahr 2010 rechnet man mit 1 GB [BJN+06, S. 216].

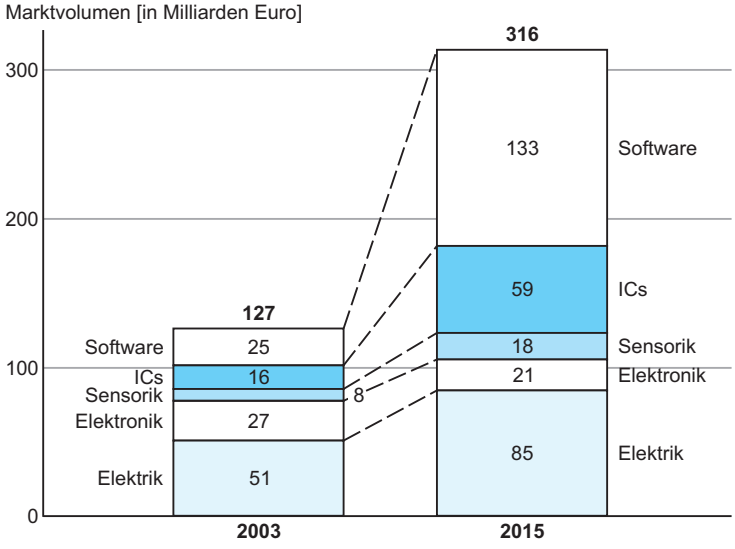


Abb. 2.0-1: Marktvolumen von Software im Automobilbereich für das Jahr 2003 mit der Prognose für 2015 (in Anlehnung an [Hons05]).

Wachsende Komplexität

Die Bedeutung der Software hat auch deshalb so stark zugenommen, weil Software in immer neuen Anwendungsgebieten eingesetzt wird und immer schwierigere Aufgaben unterstützt. Komplexere Aufgaben erfordern auch komplexere und umfangreichere Software. Die

Abb. 2.0-2 zeigt, wie sich der Umfang der Software für die Raumfahrt, für Vermittlungssysteme, für eingebettete Systeme und für Betriebssysteme entwickelt hat.

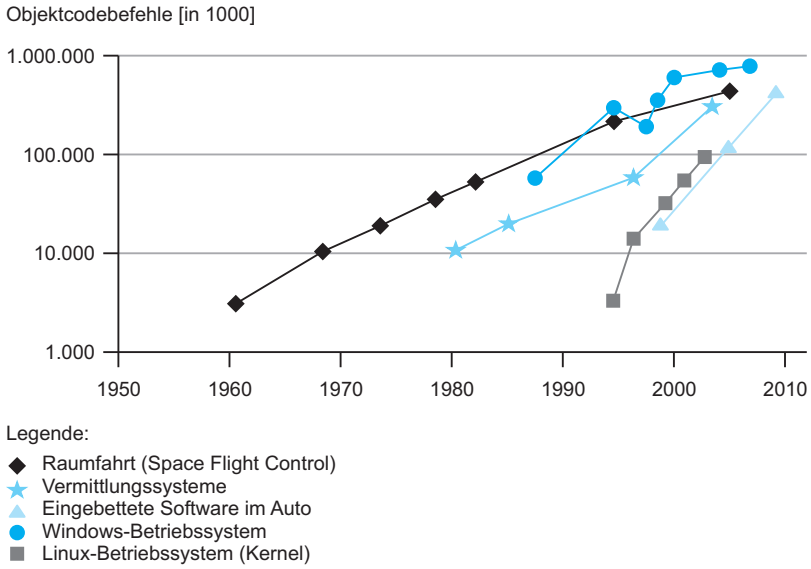


Abb. 2.0-2: Zunehmender Umfang von Softwaresystemen [Eber08, S.2].

In den 70er und 80er Jahren verdoppelte sich der Umfang nach 5–7 Jahren. Heute verdoppelt sich der Umfang bereits alle 2–4 Jahre, d. h. die Verdopplungsrate hat sich verdoppelt.

Trend

Ein Trend der letzten 10 bis 20 Jahre ist die zunehmende **Multifunktionalität** von Systemen verbunden mit einer wachsenden **Heterogenität** der assoziierten technischen Komponenten [Broy06, S. 73]. Die Zunahme multifunktionaler Geräte zeigt folgendes Beispiel: Im Jahr 2005 wurden zum ersten Mal mehr in Handys integrierte Kameras verkauft als eigenständige Kameras.

Zunehmende Qualitätsanforderungen

Zitat

»Für die Wertschöpfung im Produktionsgüter- und Dienstleistungsbereich ist Software Engineering die "Produktionstechnik des 21. Jahrhunderts" [...] Die künftigen neuen Produkte werden ohne hochgradige Qualitätszusicherungen, die nur durch *qualitätsorientierte Entwicklungs- und Produktionsprozesse* erreicht werden, nicht erfolgreich sein« [BJN+06, S. 212].

Nachfragestau und Engpassfaktor

Zitat

»Es gibt einen geschätzten Bedarf von 385.000 Softwareentwicklern in Deutschland [Zahlen von 2002, Anm. des Autors]. Auch bei vorsichtiger Bewertung dieser Zahl ergibt sich daraus, dass ein weiterer Ausbau der Softwaretechnik an deutschen Universitäten dringend erforderlich ist [...] Durch den schnellen technologischen Wandel, gerade im Bereich Software Engineering, besteht ein wachsender Bedarf an kontinuierlicher *Weiterbildung* und auch eine steigende Nachfrage nach der Zertifizierung von Software-Engineering-Kompetenzen« [BJN+06, S. 215 f.].

Nach Schätzung des Branchenverbandes BITKOM liegt der Bedarf an Informatikabsolventen von Hochschulen bei etwa 20.000 pro Jahr. Nach Angaben des Statistischen Bundesamts schlossen 2006 15.400 Absolventen ihr Informatikstudium erfolgreich ab. Daraus ergibt sich eine jährliche Lücke von ca. 5000 Informatikabsolventen pro Jahr, das ist eine Lücke von ca. 25 Prozent pro Jahr.

Der inzwischen schon hohe Grad der Akademisierung nimmt weiter zu. Ein Drittel aller beschäftigten IT-Fachleute hat einen Hochschulabschluss. Bei allen Beschäftigten liegt dagegen der Akademikeranteil knapp unter 10 Prozent. Außerdem arbeiten nur wenige in der Informatik in Teilzeit (7 Prozent verglichen mit 17 Prozent bei allen Beschäftigten im Jahr 2006) [Hohn07].

Nach Erhebungen von BITKOM sind 2009 in Deutschland 820.000 Personen in der IT beschäftigt. Zwei Drittel davon sind Hochschulabsolventen. 2009 fehlen 45.000 IT-Fachleute in Deutschland (Quelle: Computer Zeitung, 12.1.2009, S. 21).

Mehr Standardsoftware

Der Nachfragestau bei Software, aber auch die hohen Kosten der Individualsoftware haben dazu geführt, dass der Anteil der Standardsoftware ständig zunimmt.

Für Standardsoftware spricht:

- Kann dort sinnvoll eingesetzt werden, wo die Anforderungen weitgehend standardisiert sind, wie z. B. im Rechnungswesen oder in der Finanzbuchhaltung.
- Individuelle Software, die mit konventionellen Werkzeugen entwickelt wird, ist *nicht* mehr bezahlbar.
- Individualsoftware ist in der Regel fehleranfällig, da zum ersten Mal eingesetzt.

Für Individualsoftware spricht:

- Die Kosten, um die Standardsoftware an die Firmenspezifika anzupassen, übersteigen deutlich die Kosten der Standardsoftware. In den meisten Fällen wird dann aus Kostengründen das Anwendungsunternehmen an die Standardsoftware angepasst und nicht umgekehrt.
- Vorhandene Anwendungen können nur schwer in die Standardsoftware integriert werden.
- Manche Branchen sind zu klein für Standardlösungen. Im Extremfall gibt es nur ein oder ganz wenige Unternehmen in einer Branche, z. B. nur wenige große Touristikanbieter.

Trend

Trenduntersuchungen prognostizieren, dass der Anteil der Individualsoftware bis auf fünf Prozent sinken wird und dass sich solche Individualsoftware nur große Firmen leisten können. Aus diesem Trend ergibt sich, dass der Anteil vollständig neu geschriebener Software abnimmt. Zunehmend wichtiger werden Anpassungen.

Zunehmend »Altlasten«

In den letzten Jahrzehnten wurde viel Software entwickelt. Anwendungssoftware wird oft 20 Jahre und länger eingesetzt. Da sich die Einsatzumgebung dieser Anwendungssoftware ständig ändert, muss diese Software ebenfalls ständig angepasst werden. Diese permanenten Anpassungsprozesse verursachen oft 2/3 aller Software-Kosten. Bei diesen »Altlasten« stellt sich immer wieder die Frage, ob eine weitere Sanierung möglich und ökonomisch sinnvoll ist oder ob eine Ablösung durch ein neues Softwaresystem erforderlich ist.

Die Idee, sämtliche »alte Software« durch »neue Software« zu ersetzen, hat sich aus Kostengründen als *nicht* realisierbar herausgestellt. Oft wird daher alte Software »eingepackt« in eine neue Programmierschnittstelle *(Wrapping)*, sodass z. B. moderne Benutzungsoberflächen realisiert werden können. Der alte Kern der Software wird aber weiterhin benutzt.

Trend

Durch die zunehmende Verbreitung von Software werden die »Altlasten« weiter zunehmen, denn die Softwareprodukte von heute sind die Altlasten von morgen.

Zunehmend »Außer-Haus«-Entwicklung

Während früher Software in größeren Firmen durch eigene Software-Abteilungen entwickelt wurde, zeichnet sich seit einigen Jahren ein Trend ab, Software *nicht* selbst zu entwickeln, sondern bei Software-Häusern im In- und/oder Ausland in Auftrag zu geben (Outsourcing).

intern 55 % extern 45 %

Es wird davon ausgegangen, dass von den Softwareprodukten und den zugehörigen Dienstleistungen generell etwa 55 % intern und 45 % extern erbracht werden. Durch die zunehmende Produktintegration von Software (eingebettete Systeme) wird der Prozentsatz intern erstellter Software nicht drastisch zurückgehen.

Zitat

»[..] wenn es um den Fremdbezug von Anwendungen geht, dann liegt Software aus deutschen Landen weit vorne [...] Mehr als 90 Prozent setzen heimische Software ein « (Quelle: Computer Zeitung, 11.3.2009, S. 11).

Offshoring

Zitat

»Je nachdem wie sich der Mix aus lokalen und Offshore-Ressourcen ergibt, können wir in typischen Global Sourcing-Projekten eine Reduzierung der Kosten in Höhe von 30 Prozent erreichen [...] Das gelte vor allem für Projekte, die eine Laufzeit von mindestens sechs Monaten haben und dabei zehn Mitarbeiter binden« (Quelle: Computer Zeitung, 11.3.2009, S. 11).

http://www.springer.com/978-3-8274-1705-3

Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering

Balzert, H.

2009, XVIII, 624 S., Hardcover
ISBN: 978-3-8274-1705-3