Zeitschrift: Ferrum : Nachrichten aus der Eisenbibliothek, Stiftung der Georg Fischer AG

Herausgeber: Eisenbibliothek

Band: 59 (1988)

Artikel: Bewertung von Computer-Software

Autor: Ludewig, Jochen

DOI: https://doi.org/10.5169/seals-378217

Prof. Dr. Jochen Ludewig
Institut für Informatik
ETH-Zentrum
CH-8092 Zürich

Bewertung von Computer-Software

Zielsetzung und Überblick

Diejenigen, die von einem technischen Produkt auf irgendeine Weise betroffen sind, haben den Wunsch (oder sollten doch wenigstens die Möglichkeit haben), das Produkt zu bewerten und je nach Bewertung auf seine Eigenschaften oder auf seinen Einsatz Einfluss zu nehmen. Mein Beitrag diskutiert den speziellen Fall, in dem das Produkt ein Software-System ist.

Wie gezeigt wird, ist die Beurteilung von Software grundsätzlich sehr schwierig und daher prinzipiell ungenau und unzuverlässig. Für die unmittelbar Betroffenen, also den Verkäufer und den Kunden, genügt aber, wie man am florierenden Handel mit Software sieht, in den meisten Fällen eine grobe Beurteilung.

Völlig anders ist die Situation derer, die durch Software indirekt betroffen sind, ohne eine Chance zur Beurteilung und zur Einflussnahme zu haben; ihnen droht, wie im Beitrag begründet ist, eine Technokratie im Sinne des Wortes. Am Schluss sind einige Forderungen formuliert, die dieser Gefahr entgegenwirken sollen.

Der Beitrag befasst sich nur mit dem Aspekt, ob und wie Bewertung und Einflussnahme möglich sind; die Bewertung selbst, beispielsweise die der sozialen Folgen, ist hier nicht beabsichtigt.

Ich hätte diesen Text gern mit einigen Abbildungen aufgelockert. Aber Software ist abstrakt, man kann sich von ihr kein Bild machen. Das ist gerade ein Aspekt meines Themas.

Software und Software-Systeme

Zunächst ist zu klären, was Software bedeutet. In den Begriffsnormen der IEEE (1983) steht:

Software

(1) Computer programs, procedures, rules, and possibly associated documentation and data pertaining to the operation of a computer system.

(2) Programs, procedures, rules, and any associated documentation pertaining to the operation of a computer system. (ISO)

Entsprechend verstehe ich unter einem Software-System die Gesamtheit aller Programme, Verfahren und Regeln inklusive jeglicher zugehöriger Dokumentation, die einer bestimmten Problemlösung zugeordnet ist. Damit sind alle Informationen eingeschlossen, die zum Rechner-Programm in Beziehung stehen, auch wenn sie informal oder chiffriert sind. Einzige Bedingung ist, dass sie in permanenter Form vorliegen und zugänglich sind: Gedanken oder Gespräche zählen nur zur Software, wenn sie aufgezeichnet wurden.

Ein Software-System ist ein technisches Produkt, und viele Aussagen lassen sich daher von anderen Produkten auf Software übertragen (Ludewig, 1987). Allerdings gibt es spezielle Merkmale, vor allem sind zu nennen

- die im Vergleich zu anderen Artefakten extreme Komplexität
- der immaterielle Charakter und die Möglichkeit, fast ohne Aufwand Kopien herzustellen
- die sprungartige Änderung der Eigenschaften bei Modifikationen (d. h. das Fehlen der Stetigkeit)
- das Fehlen einer natürlichen Struktur

Damit entsteht insgesamt eine sehr spezielle Art von Produkten.

Der Laie nimmt meist an, dass die Funktion eines Programms durch Testen (d. h. Ausprobieren) sicher überprüft werden kann. Das folgende Beispiel zeigt aber, wie aussichtslos dieses Unterfangen ist:

Gegeben sei ein winziges Programm, das zwei Zahlen bekommt und deren Summe ausgibt. Die Zahlen seien auf den Bereich 0 bis $2^{32}-1$, also etwas über 4 Milliarden, beschränkt, eine bei modernen Computern typische Grenze. Wir wollen nun probieren, ob alle Additionen korrekt ausgeführt werden. Das bedeuteet $2^{32} \times 2^{32}$ Additionen, von denen knapp die Hälfte den Zahlenbereich überschreitet, also 2^{63} legale Additionen. Wenn wir den Test automatisieren, so dass eine Addition in einer Mikrosekunde durchgeführt und geprüft wird, so dauert der gesamte Test etwa

$2^{63} \times 10^{-6}\,s = 9.22 \times 10^{12}\,s = 2.56 \times 10^{9}\,h = 292271\,a$

also fast dreihunderttausend Jahre!

Wir können also selbst elementare Programme keinesfalls «austesten», wir können nur versuchen (wie es auch die Elektroniker bei komplizierten Schaltungen tun müssen), sie gleich richtig zu konstruieren. Leider gibt es aber dafür kein zuverlässiges Verfahren, und in der Öffentlichkeit sind viele zum Teil spektakuläre Fälle bekannt, in denen trotz erheblicher Anstrengungen Fehler gemacht wurden, unentdeckt blieben und grossen Schaden verursacht haben. Am bekanntesten ist der Verlust einer amerikanischen Venussonde, die ihr Ziel weit verfehlte, weil eines der Programme an einer Stelle einen Punkt enthielt, wo ein Komma stehen sollte (vgl. Neumann, 1988).

Das Beispiel oben zeigt, mit welcher unvorstellbaren Komplexität wir es bei Software zu tun haben: In Abwandlung einer häretischen Frage («Kann Gott einen Stein machen? Kann er ihn aufheben? Kann er ihn so gross machen, das er ihn nicht mehr aufheben kann?») lässt sich also schon zu Beginn feststellen: Ganz zweifellos kann sich der Mensch Dinge ausdenken, die er anschliessend nicht mehr versteht, und die Kluft zwischen der konstruktiven und der analytischen Fähigkeit ist vermutlich nirgends so evident wie bei Software.

Andere wichtige Software-Qualitäten entziehen sich bis heute der Quantifizierung und lassen daher vorerst keine objektive Bewertung zu. Beispiele sind die Robustheit oder die Flexibilität gegenüber Veränderungen in der Umgebung (Adaptabilität, Portabilität, Modifizierbarkeit). Arbeiten, die auf eine Quantifizierung solcher Eigenschaften zielen, laufen heute auf der ganzen Welt (Stichwort «Qualitätsmasse»), doch sind die Ergebnisse bisher wenig ermutigend.

Aspekte der Bewertung

Für das hier gewählte Thema ist echte Ein-Personen-Software, wie sie vor allem spielerisch geschaffen wird, ohne Bedeutung. Ich unterstelle also, dass es zumindest zwei Beteiligte gibt, den Hersteller und den Kunden. Dabei kommt es nicht darauf an, ob diese Parteien auch juristisch unterschieden sind, also ob sie in verschiedenen Organisationen arbeiten.

Beim Thema «Bewertung von Computer-Software» stellen sich zunächst die Fragen:

- Wer bewertet?
- Mit welchem Zweck, aus welcher Sicht wird bewertet?

Für diesen Beitrag sind die Fragen wie folgt zu beantworten:

- Wer, das sollen diejenigen sein, die mit der Software oder ihren Auswirkungen zu tun haben, also je nach Art der Software nur Hersteller und Kunde oder – beispielsweise im Extremfall einer militärischen Anwendung – alle Menschen.
- Der Zweck der Bewertung soll sein festzustellen, ob der Einsatz eines Programms nützlich oder schädlich ist, und zwar aus der Sicht und Interessenlage des Bewertenden.

Entsprechend führt der Beitrag von der praktischen Bewertung der Gebrauchstauglichkeit zur Einschätzung der Software durch die Öffentlichkeit.

Bewertung durch Fachleute

Die Bewertung eines Programms findet zunächst im Bereich zwischen dem Hersteller und dem Kunden statt. Eventuell können weitere Fachleute herangezogen werden.

Der Hersteller hat eigentlich alle Voraussetzungen, um sein Produkt realistisch zu beurteilen und wenn nötig zu verbessern. Es gibt dabei aber einige Schwierigkeiten:

- Subjektiv neigt der Programmierer (und vielleicht jeder Ingenieur) dazu, im Produkt nicht das zu sehen, was er wirklich realisiert hat, sondern das, was er realisieren wollte. Der Unterschied ist aufgrund der Komplexität eines Software-Systems schwer festzustellen.
- Wo mehrere Personen an einem System mitwirken (und das ist bei Software praktisch stets der Fall), gibt es niemanden mehr, der den vollen Überblick hat.
- Eine Beurteilung kann sich nur an Kriterien orientieren. Diese fehlen bei Software in der Praxis meist teilweise und oft ganz.
- Der verständliche Wunsch, in einem oft unseriösen Wettbewerb mitzuhalten, erhöht nicht die Bereitschaft, negative Wertungen mitzuteilen.

Der Kunde, der ein Programm auf seine Eignung untersucht, hat gewisse Erwartungen bezüglich Funktionalität und Schnittstellen (Hardware und Betriebssystem, Datei-Formate, Bedienung). Diese Dinge lassen sich (wenn auch nur mit beträchtlichem Aufwand) prüfen. Allerdings ist er dabei oft fachlich überfordert. Die nicht-quantifizierbaren Eigenschaften kann er nur durch den Gebrauch beurteilen (siehe unten, Bewertung durch die Benutzer). Seine Einflussmöglichkeiten sind durch die Marktmechanismen gegeben..

Eine unabhängige Prüfstelle kann die gleichen Untersuchungen wie der Kunde vornehmen; da sie allerdings nicht für den eigenen Bedarf prüft, muss sie Anforderungsprofile entwickeln, in denen sich nicht unbedingt jeder Kunde wiederfindet. Die Prüfstelle hat u. U. Zugriff auf die Entwicklungs- und Wartungsdokumente, also auch auf den Programmcode, und sie verfügt auch über das Fachwissen, um diese Information zu analysieren. Damit könnte sie über die oben genannten nicht quantifizierbaren Eigenschaften Aussagen machen. In der Praxis wird von dieser Möglichkeit allerdings kaum Gebrauch gemacht, weil die Wertungen nicht objektivierbar sind. Die Einflussmöglichkeiten der Prüfstellen sind erheblich, da Zertifikate grossen Einfluss auf die Marktposition der Software haben.

Die wenigsten Probleme schafft die Bewertung durch die Benutzer (die meist nicht mit dem Kunden identisch sind). Typisch ist hier die Schreibkraft, die ein (von höherer Stelle ausgewähltes) Textverarbeitungssystem verwendet. Damit entsteht durch den Gebrauch eine – wenn auch subjektive und durch die spezielle Anwendungssituation geprägte – Bewertung. Allerdings ist diese wegen des oft hohen Einarbeitungsaufwands meist nur dann möglich, wenn das Software-Paket bereits angeschafft ist. Die direkten Einflussmöglichkeiten der Benutzer sind gering, aber ihr passiver Widerstand hat schon manchen Software-Entscheid des Managements umgestossen.

Zusammenfassend lässt sich feststellen, dass die unmittelbar beteiligten Fachleute keineswegs in der Lage sind, Software vollständig oder objektiv zu bewerten, stets wird nur ein Ausschnitt erfasst und subjektiv beurteilt. Für den praktischen Gebrauch genügt diese Bewertung meist, trotz ihrer Risiken. Am günstigsten ist die Lage, wo Software immer wieder verwendet wird (Standard-Software). Dadurch werden nicht nur die Herstellungskosten aufgeteilt (oder die Gewinne vervielfacht), sondern die intensive Benutzung wirkt gleichzeitig als gigantischer Test, der entsprechend viele Fehler anzeigt. Nach einiger Zeit (und vielen Korrekturen) ist die Zahl der Restfehler tolerierbar geworden – oder das Produkt ist vom Markt verschwunden.

Bewertung eines Programms durch die Betroffenen

Da Software nicht im Vacuum hergestellt, gehandelt und verwendet wird, sind von ihr nicht nur die direkt Beteiligten betroffen, sondern u. U. sehr viele Menschen, entweder (bei Programmen, die in technischen Systemen eingesetzt sind) durch direkte physische Wirkung oder (bei Anwendungen in menschlichen Organisationen) durch Beeinträchtigung der persönlichen Rechte und Möglichkeiten. Beispiele sind

- Verkehrssysteme (Ampelsteuerungen, Eisenbahn, Flugverkehr)
- grosse Anlagen mit hohem Betriebsrisiko (Kraftwerke, chemische Prozesse, Staudämme)
- militärische Einrichtungen (rechnerunterstützte Lagebewertung, automatische Auslösung und Steuerung der Waffen)
- Computeranwendungen zur Speicherung und Auswertung persönlicher Daten (alle Programme in Verwaltungen, bei der Polizei, in Firmen, Banken usw.)

In allen diesen Fällen besteht neben dem (tatsächlichen oder vermeintlichen) Nutzen ein Risiko, dass Menschen durch Software zu erheblichem Schaden kommen.
So besteht beispielsweise der Nutzen einer Software für die Verfahrenstechnik im günstigsten Fall darin, dass die Leistung der Anlage optimal genutzt wird, wenig Schadstoffe erzeugt werden und kaum Überwachungsarbeiten erforderlich sind. Das Risiko entsteht durch die Schäden an Menschen, an der Natur und an Sachen, die durch ein Versagen der Software verursacht werden können.

Formen der Auseinandersetzung mit der Technokratie

Die vorgesehenen Möglichkeiten, auf Systeme irgendeiner Art einzuwirken, sind in demokratischen Staaten unterschiedlich, aber typisch wie folgt gestaffelt:

Zuordnung	**Einfluss**
voll im privaten Bereich	frei entscheidbar
Grenze des privaten Bereichs	Besprechung mit Beteiligten
Arbeitsbedingungen	Mitbestimmung oder Mitsprache
Lebensumgebung	Stimmrecht auf Gemeindeebene usw.
Gesetzgebung, Beziehungen zu anderen Staaten	Stimmrecht auf Landesebene

Darüber hinaus gibt es Wege, den Widerstand gegen eine Entwicklung zu demonstrieren:

- Protest in der Öffentlichkeit (Demonstration)
- Symbolische Sabotage (heute vielfach als «Aktionen»)
- Sabotage mit praktischer Wirkung

Die Grenzen zwischen diesen Formen sind fliessend.

Die zweite dieser Widerstandsformen liegt bezüglich ihrer Legalität im Grenzbreich; daher gibt es dazu auch eine schwankende Rechtsprechung. Zumindest werden solche Aktionen aber von vielen Menschen als moralisch legitim betrachtet (Beispiel: Verhindern der Giftmüllverklappung in der Nordsee).

Untersucht man, welche Rolle die Einfluss- und Widerstandsformen bei Software spielen, so zeigt sich, dass die Betroffenen, soweit sie nicht Hersteller, Kunde oder Berater sind, praktisch keinerlei Möglichkeiten haben. Die systemkonforme Mitwirkung ist nicht anwendbar:

- Die Betroffenen werden nicht informiert.
  Es ist extrem schwierig (und scheitert sogar innerbetrieblich in vielen Fällen), die notwendigen Informationen zusammenzutragen und so

aufzubereiten, dass der Laie damit etwas anfangen kann. Dieser Aufwand wird in der Praxis nicht betrieben.

- Die Voraussetzung zur Auseinandersetzung reicht bei den Betroffenen nicht aus. Selbst wenn man unterstellt, dass die Intelligenz homogen verteilt ist, erfordert die Auseinandersetzung mit Software-Entscheiden Ausbildung und Erfahrung, über die nur eine winzige Minderheit verfügt.
- Die Betroffenen haben keinen Einfluss.
  Selbst in der Schweiz, wo durch die direkte Demokratie sehr viele Einzelentscheide durch Abstimmungen getroffen werden können, beschränkt sich die Mitwirkung auf eine Auswahl zwischen JA und NEIN (was selbst zum Gemüse-Einkauf nicht ausreicht). Wahlen ersetzen die Sachentscheide durch Personalentscheide und schliessen damit einen direkten Einfluss auf technische Entwicklungen aus.

Noch weniger als die traditionellen Mitwirkungsmöglichkeiten lassen sich die Widerstandsformen gegen Software einsetzen:

- Für eine plakative Forderung ist die Materie zu komplex, im Falle einer Demonstration wäre die Zielsetzung kaum zu vermitteln, zumal kaum einer der Journalisten verstände, worum es geht.
- Symbolische Sabotage scheitert am nichtmateriellen Charakter der Software. Man kann sich wohl vor ein Waffenlager auf die Strasse setzen oder im Schlauchboot vor ein Walfangschiff fahren, man kann aber nicht Software blockieren.
- Echte Sabotage ist nicht nur definitv illegal, sondern auch völlig aussichtslos, weil sich Software praktisch kostenlos und fast verzögerungsfrei vervielfältigen und über beliebige Distanzen transportieren lässt. Darum hat die Zerstörung von Software keine Wirkung (ausser in Organisationen, die die elementaren Regeln der Software-Sicherung vernachlässigen). Eine Gefahr bildet dagegen die unbemerkte Sabotage, also das «Verseuchen» von Programmen. Allerdings ist dies extrem schwierig und angesichts der laufend verbesserten Sicherungsmassnahmen praktisch nur Insidern möglich.

Regeln für den verantwortungsvollen Einsatz von Software

Fälle der individuellen Ohnmacht gibt es im gesamten Bereich der Technik, die Situation ist nicht softwarespezifisch. Beispiele sind gerade alle diejenigen technischen und organisatorischen Systeme, in denen auch Rechner eingesetzt werden können, also beispielsweise Strassen und Transportsysteme, Waffensysteme, Staudämme und Kraftwerke, Fabriken und Lagereinrichtungen. Durch die Verwendung von Software gibt es aber insofern einen qualitativen Sprung, als alle traditionellen Formen der Mitwirkung und des Protests unbrauchbar werden.

Die Verwendung von Software in Anwendungen mit vielen Betroffenen stellt also einen technokratischen Vorgang im strengen Sinne des Wortes dar: Der Mensch wird von der Technik wesentlich beeinflusst (also regiert), ohne sich dagegen wehren zu können. Welche Schlüsse sind daraus für die Anwendung von Software zu ziehen?

1. Die Entscheidung, in einem bisher von Menschen kontrollierten System Software einzusetzen, hat sehr weitreichende Konsequenzen. Wo diese nicht überblickt werden oder nicht akzeptabel sind, kommt die Verwendung von Software nicht in Frage (Goldberg, 1985).
2. Da die Möglichkeit der Mitwirkung bei Software-Entscheiden extrem eingeschränkt ist, erfordern solche Entscheide mehr als nur eine einfache Mehrheit, sie erfordern einen mehr oder minder vollständigen Konsens.
3. Da die Fachleute einen Informationsvorsprung haben, der noch deutlich höher als auf anderen technischen Gebieten ist, tragen sie in besonderem Masse Verantwortung. Auf diese Verantwortung muss immer wieder hingewiesen, sie muss unterrichtet und eingefordert werden (Brauer, Hesse, 1987; Valk, 1987).

Literatur

Brauer, W., Hesse W. (Hrsg.) (1987): Themenheft «Zur Verantwortung des Informatikers». Mit Beiträgen von W. Brauer und W. Hesse; D. L. Parnas; H. W. Hofmann; K. H. Bläsius und J. H. Siekmann. Informatik-Spektrum, 10 (Februar 1987), 1–39.

Goldberg, A. (1985): President's letter: Reliability of computer systems and risks to the public. Communications of the ACM, 28, 131–133.

Ludewig, J. (1987): Software Engineering: Computer-Programme als technische Produkte. Technische Rundschau, 79 (1987), Heft 7, 50–57.

Neumann, P. (1988): Software Failures. Buch mit einer Sammlung spektakulärer Software-Fehler. Genauer Titel unbekannt, erscheint 1988.

Valk, R. (1987): Der Computer als Herausforderung an die menschliche Rationalität. Informatik-Spektrum, 10 (April 1987), 57–66.

4. Entscheide in Sachen Software sind aus den dargelegten Gründen tendenziell obskur und verlangen daher in besonderem Masse nach Transparenz; die Einrichtung der Stelle eines Datenschutzbeauftragten ist ein kleiner, aber richtiger Schritt in diese Richtung.

Zusammenfassung

Software lässt sich heute aus technischer Sicht unter gewissen Aspekten präzise beschreiben und prüfen. Allerdings deckt dies einen grossen und wichtigen Teil der Anforderungen und Erwartungen nicht ab (beispielsweise die Korrektheit, die sich nicht prüfen lässt, vor allem aber Eigenschaften wie Robustheit und Wartbarkeit). Offenbar sind diese Schwierigkeiten aber für die Direktbeteiligten, also Hersteller und Kunden, in der Regel erträglich.

Das gilt aber nicht für diejenigen, die nicht beteiligt, aber möglicherweise betroffen sind: Sie sind den Fachleuten blind ausgeliefert. Daher tun sie gut daran, diesen Fachleuten mit grossem Misstrauen zu begegnen und ihnen laufend auf die Finger zu schauen (was nur heissen kann: durch andere Fachleute schauen zu lassen). Wo das Risiko zu hoch ist, darf Software nicht eingesetzt werden.

Vor hundert Jahren: Der Bau des Eiffelturmes

C. Moser
Eisenbibliothek

Schon damals Opposition gegen neue Bauwerke

Eine der schönsten und repräsentabelsten Eisenkonstruktionen feiert dieses Jahr ihr 100jähriges Baujubiläum. Gemeint ist der Eiffelturm. Mit den Fundamentierungsarbeiten wurde am 28. Januar 1887 begonnen, und knapp 26 Monate später, am 31. März 1889, fand die offizielle Einweihung des Turmes statt. Die Meisterleistung der Ingenieurbaukunst des 19. Jahrhunderts war damit erreicht.

Aber nicht erst in unseren Tagen regt sich der Unmut gegen Bauwerke. Auch das Projekt der Eiffelturmes wurde angegriffen.

Vehemente Proteste

Als die Stadtbehörden von Paris ihre Zustimmung für den Bau des 300 Meter (oder 1000 Fuss) hohen Turmes publik machten, gab es eine grosse Opposition gegen dieses Vorhaben. Künstler und Schöngeister griffen das Projekt vehement an. Für sie schien der Turm eine Verschandelung des Stadtbildes, ein nacktes Gerüst, und das ganze Vorhaben kam einer Gotteslästerung gleich. Den Höhepunkt der Kritik am Eiffelturm stellt der schriftliche «Protest der Künstler» dar, der an den damaligen Direktor der Arbeiten der Stadt Paris, an A. Alphand adressiert war:

«Mein Herr und lieber Landsmann. Wir, Schriftsteller, Maler, Bildhauer, Architekten, leidenschaftliche Liebhaber der bis jetzt unversehrten Schönheit von Paris, wollen mit allen unseren Kräften, unserer ganzen gerechten Entrüstung im Namen des verkannten französischen Geschmacks, im Namen der bedrohten französischen Kunst und Geschichte gegen die Errichtung des unnützen und missgestalteten Eiffelturms mitten im Herzen unserer Hauptstadt protestieren, den die so oft von gesundem Menschenverstand und Gerechtigkeitssinn geprägte Bosheit des Volkes bereits ‹Turm zu Babel› getauft hat.»

So beginnt dieser Aufruf, und er ergiesst sich weiter in Gemeinplätzen, die dem damaligen Zeitgeschmack entsprachen, ohne dabei etwas Sachliches am Projekt auszusetzen.