Quo Vadis Software Engineering

Herbert Weber

Februar 2001

Während der CeBit 1999 haben die deutschen Fachverbände der Informations-, Kommunikations- und Medientechnik bekannt gegeben, dass mit ihren Produkten und Dienstleistungen eine Gesamtwertschöpfung erreicht wurde, die der der Automobilindustrie entspricht. Die Branche ist damit zu einer Spitzenbranche der deutschen Wirtschaft aufgestiegen, die mit ihren Wachstumsraten im zweistelligen Prozentbereich die meisten anderen Wirtschaftszweige überflügelt.

Die meisten der heute installierten betrieblichen Systeme sind im Ablauf der letzten Jahre zu integrierten Informations- und Kommunikationsinfrastrukturen zusammengewachsen, mit deren Hilfe nunmehr die Steuerung der gesamten Unternehmensprozesse und des betrieblichen Wissensmanagements ermöglicht wird. Sie sind damit heute von kritischer Bedeutung für Unternehmen und Organisationen jeder Art. Ohne ihre Existenz wären weder einzelne Aufgaben zu erledigen, noch ganze Geschäftsabläufe abzuwickeln und schon gar nicht ganze Unternehmen zu führen. Sie stellen damit eine unabdingbare Basis für das gesamte Handeln in Wirtschaft und Gesellschaft dar. Ihre Betreiber sind nicht mehr Servicezentralen für ihre Organisationen, sondern deren »Nervenzentralen«.

Die Basis dieses Erfolgs ist Software: Informations-, Kommunikations- und Medientechnik sind vor allen Dingen softwaretechnische Problemstellungen und -lösungen.

Software nimmt dabei unterschiedliche Funktionen wahr:

- Software wird Teil technischer Produkte: Ein oder mehrere Mikroprozessoren und die für ihren Einsatz notwendige Software steuern z.B. Verkehrssysteme, Maschinen, Anlagen, Kommunikationseinrichtungen etc.
- Software ist Basis einer Dienstleistung: In Banken, Versicherungen oder Touristikunternehmen etc. wird Software zum zentralen Arbeitsmittel für die dortigen Geschäftsaktivitäten, ohne deren Existenz die Geschäftsaktivitäten nicht durchführbar wären.
- Software ist ein eigenständiges Produkt: Software für so unterschiedliche Anwendungen wie Computerspiele, Steuerberechnungen, Medizinische Diagnosesysteme etc. ist ein Wirtschaftsgut, das in einem offenen und globalen Markt entwickelt und angeboten wird.

Software musste in ihren vielen Anwendungen dabei immer auch der Erwartung genügen, das flexibel änderbare und anpassbare technische Produkt zu sein, mit dessen Hilfe jeweils auch die notwendige Spezialisierung und Individualisierung von Anwendungen möglich gemacht werden kann. Diesem Anspruch sind Software-Entwickler und -Anbieter zunächst dadurch gerecht geworden, dass sie in immer umfangreicheren Pflege- und Wartungsaufwänden bis zu 70% und mehr ihrer gesamten Kapazität gebunden haben. Sie sind der Anforderung aber auch dadurch gerecht geworden, dass sie die entwickelten Systeme durch Parametrisierung zur Anpassung an vorhersehbare Änderungsanforderungen befähigt haben und dabei die Komplexität der Systeme zu Lasten der Handhabbarkeit und Beherrschbarkeit drastisch erhöht haben. Vor allen Dingen die Entwickler und Anbieter von Software, die Teil technischer Produkte ist, sind dieser Anforderung aber auch häufig dadurch gerecht geworden, dass sie mit jeder neuen Variante oder jedem neuen »Modell« des technischen Produkts die dazu nötige Software komplett neu erstellt haben.

Mit diesen Vorgehensweisen sind häufig technisch unakzeptable Lösungen und wirtschaftlich nicht vertretbare Aufwände entstanden.

Und wie kann die Entwicklung nun weitergeführt werden?

- Die in vielen Bereichen der Wirtschaft über zehn, zwanzig oder sogar dreißig Jahre Schritt für Schritt entstandenen Informations- und Kommunikations-infrastrukturen bestehen aus hunderten von Software-Systemen und Datenbeständen, deren Erhalt und Weiterentwicklung immer größere Schnittstellenprobleme und Erhaltungsaufwände mit sich bringen. Die Überwindung dieser Probleme erfordert die komplette Renovierung, d.h. die Überführung möglichst großer Teile der Altsysteme in möglichst weitgehend standardisierte (Branchen-) Architekturen mit standardisierten Schnittstellen, die Reduktion bzw. Elimination von Datenredundanzen und die Flexibilisierung durch Vorkehrungen für die weitere Änderungsfähigkeit der Informations- und Kommunikationsinfrastrukturen.
- Für Software als Bestandteil technischer Produkte müssen, in gleicher Weise wie für die technischen Produkte selbst, »Produktlinien« etabliert werden, mit denen ein möglichst großer Anteil der einmal entwickelten Software in den nächstfolgenden Produktgenerationen wiederverwendet werden kann.
- Die in Informations- und Kommunikationsinfrastrukturen zusammengeführten Software-Systeme verschiedener Hersteller werden zu neuen Releases, Varianten und Versionen weiterentwickelt und müssen in ihren neuen Ausprägungen in die Infrastruktur eingepasst werden können, ohne dass diese

Infrastruktur in ihrer Gesamtheit komplett renoviert werden muss.

Dies sind einige der Weiterentwicklungsszenarien, mit denen die Betreiber von Informations- und Kommunikationsinfrastrukturen konfrontiert sind. Deren systematische und wirtschaftliche Weiterentwicklung in den beschriebenen Szenarien kann sich bisher nicht auf erprobte ingenieurtechnische Verfahren abstützen.

Der größte Teil der heute in der Praxis installierten Software-Systeme sind also als Heterogene Software Föderationen zu bezeichnen: Sie sind aus voneinander unabhängig entwickelten Systemen durch deren schrittweise Integration entstanden und die in den Föderationen integrierten Systeme führen bis zu einem gewissen Grad ein »Eigenleben«. Sie sind möglicherweise von unterschiedlichen Herstellern, sie unterliegen unterschiedlichen Aktualisierungszyklen, sie sind heterogen im Hinblick auf Datenhaltung und Datenverwaltung, sie sind heterogen im Hinblick auf Basissysteme und im Hinblick auf Implementierungssprachen, etc. Heterogene Software Föderationen unterliegen sehr weitreichenden Aktualisierungs- und Anpassungsanforderungen und das Engineering für Aktualisierung und Anpassung muss kontinuierlich stattfinden.

Das »Continuous Engineering« soll dafür ingenieurtechnische Verfahren bereitstellen. Mit der Entwicklung von Informations- und Kommunikationsinfrastrukturen vollzieht sich ein drastischer Wandel der Natur solcher Systeme: Sie sind nicht mehr Produkte mit einer begrenzten Lebensdauer, sondern von unbegrenzter Lebensdauer. Notwendige Weiterentwicklungen, Anpassungen und Erweiterungen erfolgen durch partielle Erneuerung, durch partielle Erweiterung und durch partiellen Ersatz, ohne dass die Grundkonzeption der Infrastruktur in Frage gestellt werden kann.

Für die Weiterentwicklung von Informations- und Kommunikationsinfrastrukturen können nicht mehr nur klassische Verfahren angewendet werden. Die für das Continuous Engineering von Infrastrukturen notwendig werdenden Techniken lassen sich wie folgt charakterisieren:

- Systeme müssen sich leicht weiterentwickeln lassen, um neuen Benutzer- oder Betreiberanforderungen ohne großen Aufwand gerecht zu werden.
- Existierende Systeme müssen zu neuen Systemen zusammengefügt werden können, um zunächst isolierte Lösungen zu neuen integrierten Lösungen weiterzuentwickeln.
- Der Austausch von Komponenten muss einfach möglich sein, so dass die Neuentwicklung eines Systems schrittweise erfolgen und die Migration von einem alten System zu seinem neuen Äquivalent kontrolliert stattfinden kann.
- Die Portierung eines Systems von einer Standard-Plattform zu einer anderen Standard-Plattform soll mit möglichst geringem Aufwand verbunden sein.
- Die Wiederverwendung existierender Systeme und Komponenten zum Aufbau anderer Systeme soll ohne größeren Aufwand möglich sein.
- Durch die Veränderung einer Software-Infrastruktur in einigen Komponenten oder Teilsystemen sollen die verbleibenden Teile nicht beeinträchtigt werden.
- Die Architektur großer Systeme soll durch die Stabilität der Standard-Plattform möglichst invariant gehalten werden. So kann die Entwicklung und Weiterentwicklung immer durch einen Orientierungsrahmen kontrolliert durchgeführt werden, der durch die invarianten Plattformen definiert wird.

An all diesen Themen wird weltweit hart gearbeitet. Werden damit die Durchbrüche erzielt werden, um Aufwände für Entwicklung und Erhalt von Software drastisch zu reduzieren und die Qualität der Produkte drastisch zu erhöhen? Meine Antwort ist nein. Und dazu biete ich Ihnen die folgenden Begründungen an:

1. Wir sind Gefangene unserer eigenen Erfolgsgeschichte: Software ist letztendlich ein Programm – aber eben nur letztendlich. Sie ist viel mehr – sie ist Problemlösung für ein Anwendungsproblem, ja sogar eine Vielzahl von kodierten Problemlösungsverfahren und damit von kodiertem Problemlösungswissen. Aber sie ist auch immer nur ein Teil des Problemlösungswissens – der andere Teil, der nötig ist, um Software zu betreiben, findet sich, wenn überhaupt, in schlechten Dokumenten oder in den Köpfen der Beteiligten und damit ist es extrem flüchtig. Warum messen wir dem nicht als Programm dargestellten Problemlösungswissen eine so untergeordnete Rolle zu? Hier einige meiner Antworten:
   - Wir Software-Ingenieure sind vor allen Dingen Programmierer. Wir sind als Programmierer beruflich sozialisiert worden und empfinden Begeisterung für die Darstellung von Wissen durch Algorithmen und Programme. Wir haben algorithmisches Denken bis zu dem Grad verinnerlicht, dass es uns schwerfällt zu glauben, dass nicht jede Art von Problemlösungswissen als Algorithmus dargestellt werden kann. Wir lassen uns auf die Verwendung abstrakter Darstellungen als Modelle oder Spezifikationen ein um sie alsbald als zu aufwendig und damit lästig zu empfinden und sie über Bord zu werfen, d.h. wir kümmern uns einfach nicht um deren Aktualisierung.
   - Wir nutzen andere Darstellungen von Problemlösungswissen in Form von Spezifikationen oder Modellen häufig auch zu recht nur mit Widerwillen, weil sie nicht wirklich eine neue Abstraktion einführen, sondern eine Darstellung auf dem Abstraktionsniveau des Programms sind. Damit sind sie auch ähn-

lich umfangreich, komplex und kompliziert und ihre Erstellung kann mit einiger Berechtigung als Doppelarbeit zur Programmierung selbst betrachtet werden. Wenn entsprechende Techniken dann als Standards die Summe aller in den Standard einfließenden Konzepte umfassen, sind sie in aller Regel überladen und schlechter handhabbar als Programmiersprachen.

- Es mag vielleicht arrogant klingen, aber ich nehme für mich persönlich in Anspruch, den größten Teil der neueren Entwicklungen von Darstellungen von Problemlösungswissen als das Recycling alter Ideen zu betrachten, die die gewünschten Durchbrüche nicht hervorgebracht haben. Ich tröste mich damit, dass das zyklische »deja vue« letztendlich zur Etablierung eines »Canon« für das Software Engineering führen wird, das die nachhaltigsten Konzepte umfassen wird. Dies wir dann eben ein «Canon«, aber kein Durchbruch sein.

Meine Schlussfolgerung aus dem Obigen ist: Wir Software-Ingenieure schaffen Artefacte einer Größe und Komplexität, die wir mit dem uns zur Verfügung stehenden Instrumentarium nicht beherrschen können.
Wie überwinden wir das Dilemma?

- Wir müssen Software-Ingenieure sowohl zu algorithmischem Denken, aber in noch viel stärkerem Maße auch zu konstruktivem Denken befähigen.
- Wir müssen Instrumente für das Wissensmanagement für den »konstruktiven Software-Bau« auf einem weit oberhalb der Programmierung gelegenen Abstraktionsniveau entwickeln und anbieten. An eine «bottom-up«-Entwicklung dieser Konzepte – quasi von der Programmierung abstrahierend – glaube ich nicht mehr. Radikal neue Ideen für das Wissensmanagement braucht das «Software Engineering Land«.

2. Wir haben uns in eine Sackgasse treiben lassen:
   Viele von uns glauben noch immer, das Allheilmittel für das Software Engineering sei die bessere Gestaltung des Software-Prozesses – also der geplanten Abläufe des Software Engineering. Ich gebe zu, ich habe mich schon vor dem großen Auftritt von Lee Osterweil schwer damit getan, die idealisierten Lebenszyklusmodelle ernst zu nehmen, wie immer sie auch ausgesehen haben und von wem immer sie auch massiv gestützt werden. Um so mehr hat mich die Vorstellung, den Software-Prozess als Programm zu begreifen, gestört. Die vielen weichen Faktoren der Software-Entwicklung halte ich seit langem für die wesentlich wichtigeren, als die in Software-Prozessen darstellbaren harten Faktoren. Ich muss gestehen, dass mich nach meiner ersten Skepsis vor 15 Jahren der Mut verlassen hat, dies auch öffentlich lautstark zur Kenntnis zu bringen. Das bedaure ich heute – unabhängig davon, ob ich damals gehört worden wäre oder nicht. Noch schlimmer, während eines großen EUREKA-Projektes, an dem ich damals federführend beteiligt war, ist ein erheblicher Aufwand in die Entwicklung des harten Software-Prozess-Managements geflossen. Nicht überraschend war, dass die im Projekt erarbeiteten Resultate dann eine viel größere Bedeutung für die Planung und Steuerung von Geschäftsprozessen als für Software-Prozesse gewonnen haben.
   Ich denke inzwischen, das »Software-Prozesse« ein in die Sackgasse führendes Konzept sind.
   Um nicht missverstanden zu werden, auch ich bin der Meinung, dass Software Engineering organisiert werden muss. Aber nach meinem Dafürhalten muss vor allen Dingen die Kommunikation zwischen den Beteiligten organisiert werden. Große Software-Projekte erfordern die Einbindung der Beteiligten in immer wieder neue Kommunikations-Teams, die Organisation der Kommunikation in den Teams und die Kommunikation zwischen den Teams. Software Engineering zu organisieren, entspricht für mich heute der Aufgabe, »Communicating Communities« zu organiseren.
   In Communicating Communities wird Wissen in den Köpfen der Beteiligten erzeugt, Information auf Basis dieses Wissens kommuniziert, Information zu Wissen in Beziehung gesetzt etc. Um dann dieses Wissen im Software Engineering als Wert betrachten zu können und gezielt zugänglich zu machen, ist eine Darstellung dieses Wissens durch Informationen und das dazu notwendige Informationsmanagement sicherzustellen. Die Abläufe, d.h. die Ordnung mit den die Abläufe bestimmenden Aktivitäten, spielen nach meiner Einschätzung eine nur untergeordnete Rolle.
   Leider wissen wir Software-Ingenieure nur annähernd wie Kommunikationsprozesse ablaufen und wie sie zu organisieren sind. Hier wird -–sicherlich mehr Mithilfe der Wissenschaftsdisziplinen, die sich mit der menschlichen Kommunikation beschäftigen – ein neuer Anfang nötig sein.

Sie werden sich jetzt vielleicht fragen, ist mir bei all der Kritik und bei all den Zweifeln, die ich vorgetragen habe, der Optimismus für die Zukunft verloren gegangen. Dies kann ich klar verneinen. Mir ist aber bewusst geworden, dass auch alte Denkmuster der Software-Ingenieure nur in langen Zeiträumen durch neue ersetzt werden können.