Georg Herzwurm, Werner Mellis, Klaus Schmolling

Software Factory – Ein Statusbericht

Die Fabrik scheint nicht mehr das Vorbild der Softwareproduktion zu sein. Dennoch zeigt sich bei genauerer Betrachtung, daß sowohl das Ziel, Softwareprozesse wie Fabrikprozesse zu beherrschen, als auch die Fabrikanalogie, Softwareprozesse wie Fabrikprozesse zu betrachten, im wesentlichen weiterhin bestehen. Weil Erwartung und Erfolg so weit auseinanderklafften, hat die zunehmend auf technische Aspekte eingeengte und auf Automatisierung zielende Interpretation die Software Factory in Mißkredit gebracht. Daß die Praxis aus den technischen Lösungsansätzen in der Forschung und Softwarewerkzeugentwicklung nicht den Nutzen ziehen konnte, den man erwartet hat, muß nicht gegen die Lösungsansätze oder die Werkzeuge sprechen. Es ist aber deutlich geworden und dies wird auch durch eine Studie unseres Lehrstuhls bestätigt, daß die Umsetzungsprobleme in der Praxis außer auf technische Probleme wesentlich auf einem Mangel an Verständnis und Beherrschung der organisatorischen und methodischen Aspekte zurückzuführen sind. Es ist also die Bedeutung der im ursprünglichen Konzept der Software Factory wesentlichen organisatorischen und methodischen Aspekte wiederentdeckt worden. Allerdings wird die Verbesserung der organisatorischen und methodischen Rahmenbedingungen der Softwareentwicklung nicht mehr unter dem inzwischen als kontaminiert geltenden Begriff Software Factory betrieben, sondern unter den Begriffen: Prozeßorientierung, Total Quality Management etc. Aber es gibt inzwischen neben der Kritik einer zu eingeschränkten Problemsicht auch substantielle Kritik an der Fabrikanalogie selbst. Das verbesserte Verständnis der Softwareprozesse hat die Grenzen der Analogie zu den Fabrikprozessen deutlich gemacht.

Inhaltsübersicht

1 Entwicklung der Software Factory

1.1 Entwicklung in der Literatur

Bereits Ende der 60er Jahre wurden in der Literatur die ersten Vorschläge unterbreitet, wie die Softwareentwicklung nach dem Vorbild der industriellen Produktion erfolgen sollte: In der „Software Factory" werden mit Hilfe standardisierter, computergestützter Werkzeuge auf der Basis eines formalisierten, mittels technischer und ökonomischer Kennzahlen kontrollierten Prozesses Softwareprodukte – gegebenenfalls in „Massenproduktion" – erstellt [1]. In den darauffolgenden Jahren konzentrierte sich die Diskussion vor allem in Europa immer stärker auf technische Fragen. So wurde der Weg vom Computer Aided Software Engineering (CASE) zur Software-Fabrik im Rahmen des EG-Projekts European Software Factory (ESF) in erster Linie als Aufbau einer Software-Produktionsumgebung mit integrierten, „intelligenten" Werkzeugen und standardisierten Protokollen verstanden [2]. Mit der Software Factory wurden unmittelbar Schlagworte wie Standardisierung, Modularisierung und geplante Wiederverwendung assoziiert. Erst in den frühen 80er Jahren wurden auch die frühen Phasen der Software-Entwicklung explizit mit in die Überlegungen eingeschlossen und der Fokus wieder zunehmend auf die methodischen und

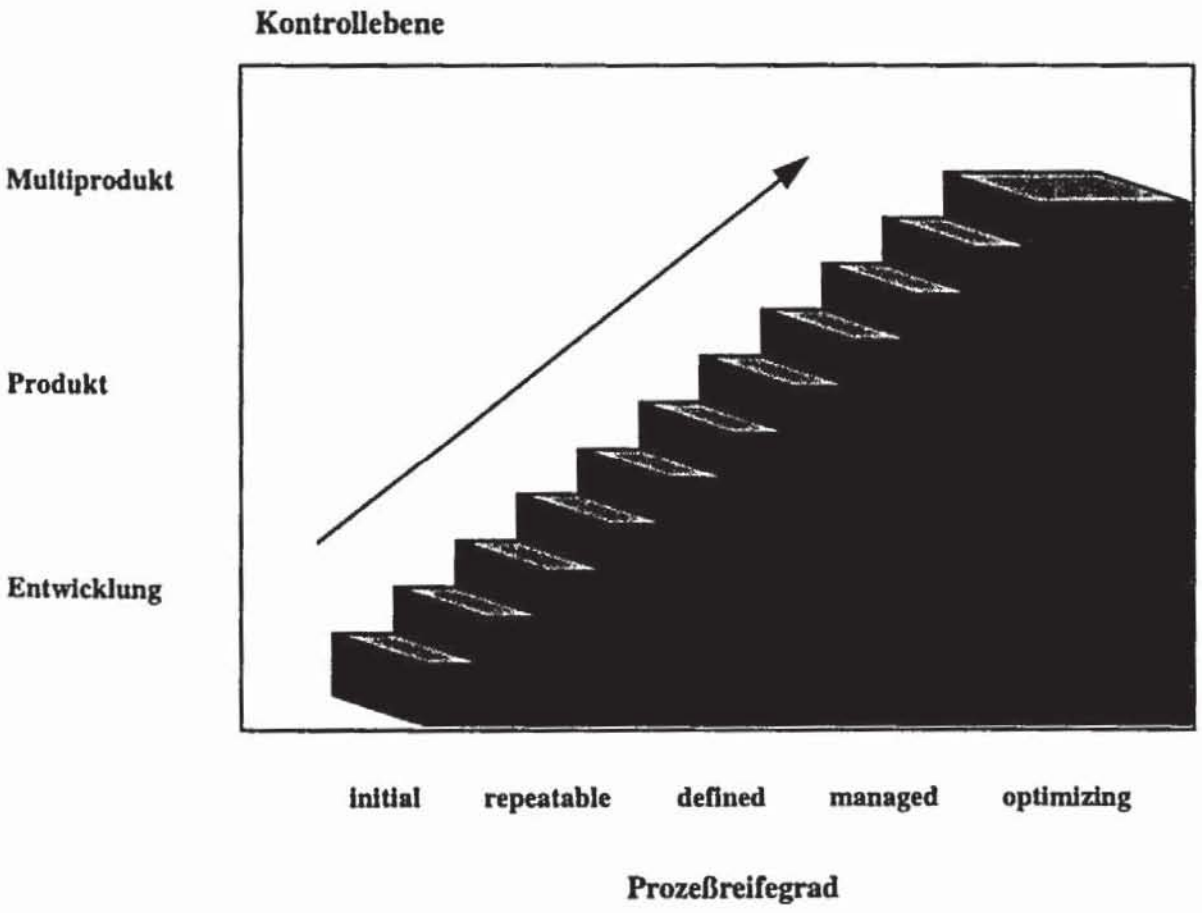

Abb. 1 Zusammenhang zwischen Kontrollebene und Prozeßreifegrad

organisatorischen Aspekte der Software-Entwicklung gesetzt. Die jüngste Literatur zum Thema Software Factory stellt in Analogie zur industriellen Produktion die Planung und vor allem die Kontrolle des Prozesses in den Mittelpunkt der Betrachtung [3]. Die Prozeßkontrolle beschränkte sich in den frühen Jahren der Software Factory auf die Entwicklungsphase (Projektkontrolle). Für die gezielte Unterstützung der Wartung und der Wiederverwendung wird allerdings darüber hinaus eine Produktkontrolle über den gesamten Produktlebenszyklus benötigt. Zur Erfüllung der Anforderungen einer Software Fabrik ist nach modernen Auffassungen sogar eine Multiproduktkontrolle erforderlich. Dieser Weg vom Projekt- zum Prozeßmanagement läßt sich nur stufenweise erreichen. Für die Software-Entwicklung hat das Software Engineering Institute (SEI) in seinem Capability Maturity Model (CMM) die Reife eines Softwareentwicklungs-Prozesses in fünf Stufen vom Chaos bis zum optimierten Prozeß beschrieben [4]. Den Zusammenhang zwischen den Kontrollebenen und den Prozeßreifegraden veranschaulicht Abb. 1.

Betrachtet man die quantitative Entwicklung wissenschaftlicher Veröffentlichungen zum Thema Software Factory, so stellt man fest, daß der Zenit Ende der 80er bzw. Anfang der 90er Jahre überschritten wurde [5]. Unterstellt man einen positiven Zusammenhang zwischen der Anzahl wissenschaftlicher Veröffentlichungen zu einem Thema und dessen Aktualität in Forschung und Praxis, so kommt man zu dem Ergebnis, daß das Schlagwort Software Factory in den letzten Jahren an Popularität verloren hat (siehe Abb. 2). Viele moderne Ansätze des Software Engineering wie ISO 9000 und CMM basieren jedoch auf einem Fabrik-Modell, was die Vermutung nahelegt, daß zwar der Begriff Software Factory, nicht aber die damit verbundenen Inhalte und Konzepte aus der Mode gekommen sind.

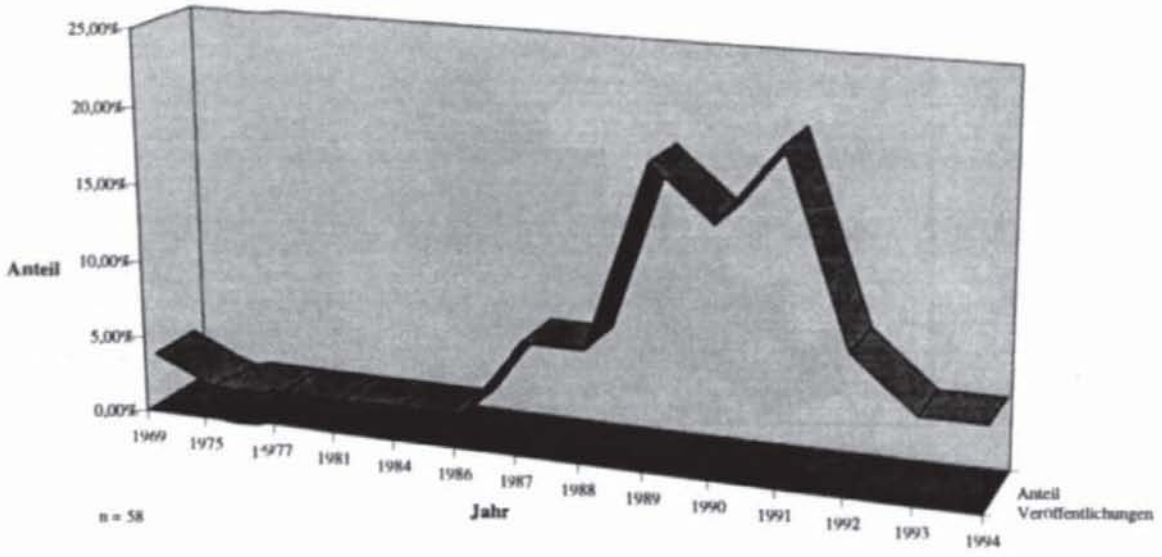

Abb. 2 Entwicklung der wissenschaftlichen Veröffentlichungen zum Thema Software Factory

1.2 Entwicklung in der Praxis

In den U.S.A. gilt die amerikanische System Development Corporation (SDC) als Pionier im Bereich Software Factory. Die SDC Software Factory bestand aus drei Elementen:

- einem integrierten Toolset (Programm-Library, Projektdatenbank, Online-Interface zwischen Tools und Datenbank, automatisierte Unterstützung von Test- und Dokumentation),
- standardisierten Verfahrens- und Arbeitsabläufen für Design und Realisierung,
- einer Matrixorganisation mit organisatorischer Trennung der Entwicklung mit Kundenkontakt („high level system design") und der eigentlichen Programmierung in der Software Factory („program development").

Nach anfänglichen, sehr beeindruckenden Erfolgen stellte sich im Lauf der Zeit heraus, daß die eher proprietären Tools nicht sehr portabel und flexibel waren, so daß immer mehr Projektleiter versuchten, „an der Software Factory vorbei" zu entwickeln.

Hitachi implementierte die erste und bis heute größte Software Factory in Japan [6]. Eine zentralistische Organisation mit unternehmensweiter Prozeßstandardisierung und -kontrolle sollte die Software-Entwicklung von einem unstrukturierten Service zu einem bezüglich Qualität und Kosten planbaren Produkt wandeln. Auch hier stellten sich anfangs Verbesserungen ein, aber letztendlich fehlten zum endgültigen Erfolg ausgereifte Reuse-Konzepte ebenso wie Techniken zur Bestimmung der Best Practice als Voraussetzung für das Setzen erfolgreicher (Unternehmens-)Standards.

Letztlich scheiterten viele Software Factory-Konzepte in der Praxis daran, daß die ursprünglich im Vordergrund stehenden organisatorischen Aspekte mehr und mehr zugunsten technischer Fragen in den Hintergrund traten. So wurde die Wiederverwendung häufig zwar durch Tools, nicht aber durch entsprechende organisatorische Maßnahmen unterstützt. Erst die jüngeren Ansätze in Forschung und Wissenschaft zeugen von einer Wiederentdeckung der organisatorischen Probleme, worauf in Kapitel 3 ausführlicher eingegangen wird.

2 Software Factory-Elemente in deutschen Unternehmen

Die Gewährleistung der konstituierenden Elemente des Software Factory-Konzepts bildet die notwendige Voraussetzung zur Realisierung der hiermit intendierten Produktivitätseffekte in der Softwareentwicklung. Inwieweit diese Voraussetzungen in den Software-Entwicklungsabteilungen von deutschen Unternehmen gegeben sind bzw. ein Problem darstellen, und inwiefern die am Markt verfügbaren Werkzeuge die Anforderungen des Software Factory-Konzepts erfüllen, ist Gegenstand des folgenden Abschnitts. Grundlage hierbei sind eigene empirische Erhebungen des Lehrstuhls in den Software-Entwicklungsabteilungen der größten deutschen Unternehmen sowie die Evaluierung ausgewählter CASE-Tools [7].

2.1 Prozeßbezogene Elemente

Ausgangspunkt der Bestrebungen zur Verbesserung der Softwareentwicklung unter Bezugnahme auf das Software Factory-Konzept war insbesondere die mangelnde organisatorische Institutionalisierung des Software-Entwicklungsprozesses, woraus eine Fülle von singulären Problemen resultierte.

Ein Aspekt der empirischen Studie betraf die Bewertung prozeßbezogener Merkmale des Entwicklungsprozesses. Konkret sollten die Unternehmen Auskunft zu der gegenwärtigen und zukünftigen Bedeutung des Controllings, des Projektmanagements, der Dokumentation und der Prozeßoptimierung bei der Softwareentwicklung geben. Die Bedeutung wurde dabei anhand des Aufwands zur Gewährleistung dieser Merkmale gemessen. Wie aus der ersten Säulenreihe der Abb. 3 ersichtlich wird, ist der gegenwärtige Aufwand zur Gewährleistung der aufgeführten prozeßbezogenen Qualitätsmerkmale als sehr gering einzuschätzen. Insbesondere dem Aufgabenbereich des Controllings von Entwicklungsprozessen ist bisher wenig Aufmerksamkeit zugemessen worden. Ein Grund dafür besteht in den Schwierigkeiten bei der Erfassung der relevanten Größen. Auch heute noch wird die Entwicklung von Software weitläufig als kreativer Prozeß gesehen, dessen Ergebnisse nicht durch eindimensionale Maße, wie z. B. LOC, gemessen werden können.

Daß aber ein dringender Bedarf zum Controlling des Entwicklungsprozesses besteht, zeigt die zweite Säulenreihe der Abb. 3 anhand des prognostizierten Aufwands zur Durchführung von Controllingaktivitäten. Somit kann vermutet werden, daß eine kostenorientierte Transparenz, die als Voraussetzung zur Über-

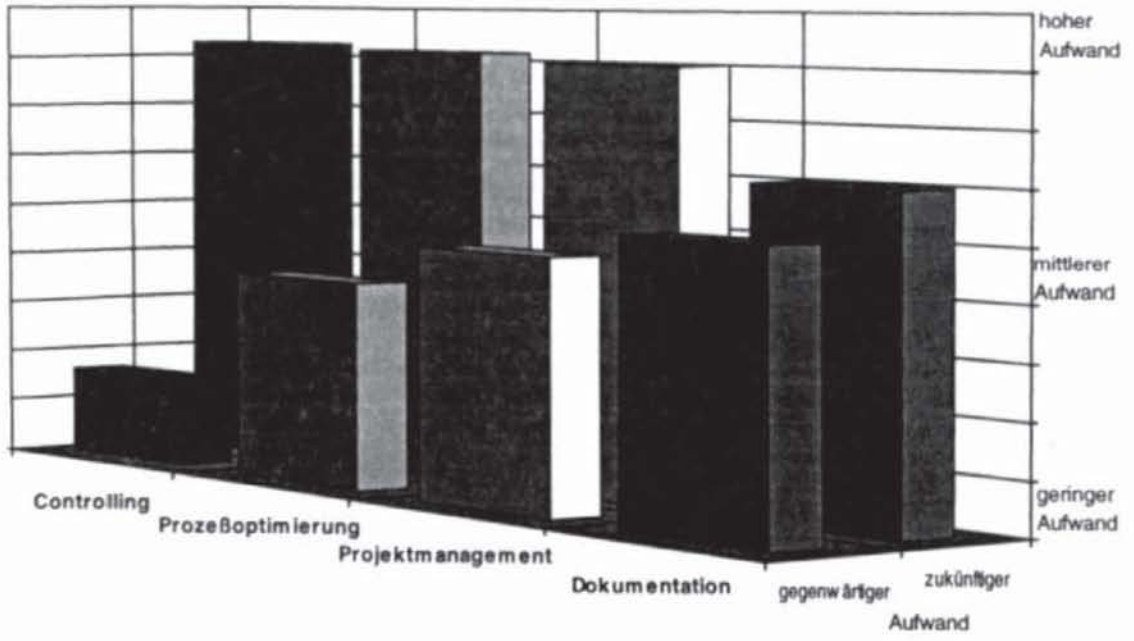

Abb. 3 Gegenwärtiger und zukünftiger Aufwand zur Gewährleistung von Qualitätsmerkmalen des Entwicklungsprozesses

prüfung der Wirtschaftlichkeit der Softwareentwicklung gilt, nicht gegeben ist. Grundsätzlich wird auch der Optimierung des Prozesses, wie auch der Durchführung eines effektiven Projektmanagements zukünftig ein höherer Stellenwert zukommen. Lediglich im Bereich der Dokumentation des Software-Entwicklungsprozesses ist die Diskrepanz zwischen IST- und SOLL-Vorstellungen der Befragten nicht so groß.

2.2 Technologiebezogene Elemente

Analog zu der Verwendung von automatisierten Hilfsmitteln in Produktionsunternehmen (z. B. CIM) wird dem Einsatz von computergestützten Werkzeugen zur methodischen Unterstützung der Softwareentwicklung innerhalb des Software Factory-Konzepts eine dominierende Rolle zugeschrieben.

2.2.1 CASE-Methoden

Aus den Antworten auf eine Frage zur Anwendung von Methoden in der oben zitierten Studie konnte festgestellt werden, daß 16% der befragten Unternehmen überhaupt keine Methoden zur Unterstützung des Software-Entwicklungsprozesses nutzen. Unter den eingesetzten Methoden ist die Entity-Relationship-Modellierung zur Datenmodellierung weit verbreitet (71,0%). Daneben werden häufig Structured Analysis zur Modellierung der funktionalen Systemanforderungen (51,6%) sowie Structured Design zur Designspezifikation (38,7%) eingesetzt. Neben diesen konventionellen Methoden wird in jüngster Zeit zunehmend über den Einsatz von objektorientierten Methoden diskutiert. Eines der wesentlichen Argumente ist hierbei die Unterstützung der Wiederverwendung.

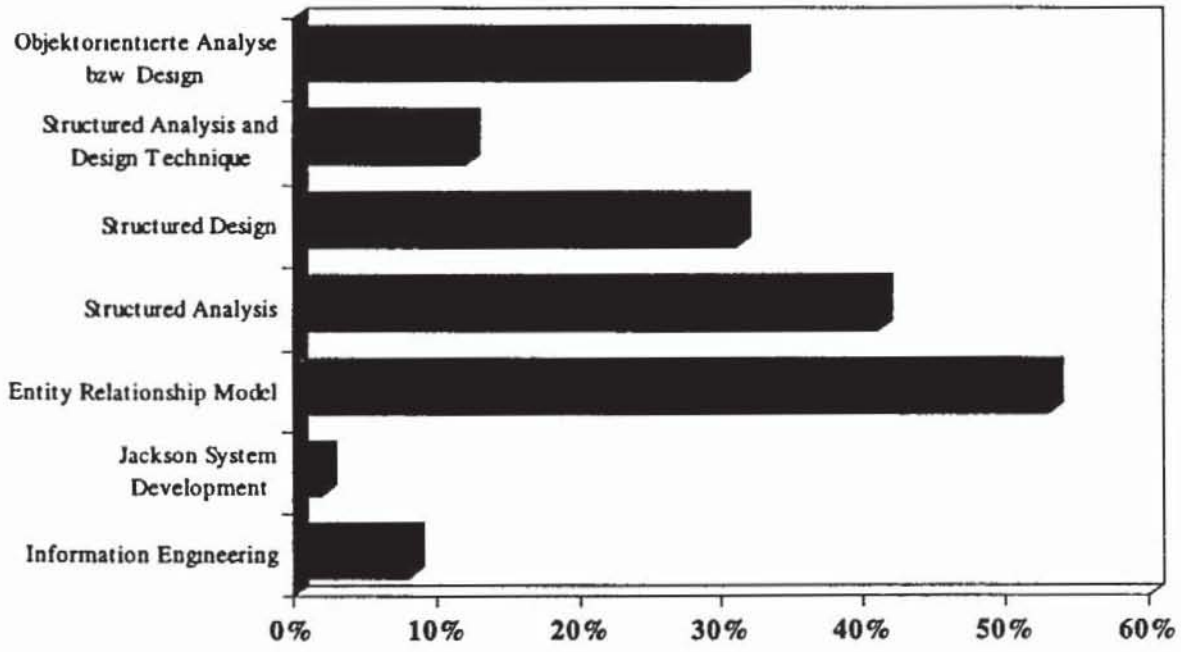

Abb. 4 Unterstützte Methoden (Basis: Befragung bei 60 Anbietern von CASE-Tools; Mehrfachnennungen waren möglich)

die eine der zentralen Komponenten im Software Factory-Konzept darstellt. Ca. 23% der befragten Unternehmen haben bisher objektorientierte Methoden zur Entwicklung von Software eingesetzt. Jedoch handelt es sich hierbei zumeist um kleinere Vorhaben (durchschnittliche Projektgröße: 16,4 Personenmonate), die somit als Pilotprojekte charakterisiert werden können.

Insgesamt gut 50% der Unternehmen, die eine oder mehrere Methoden nutzen, unterstützen diese durch CASE-Tools. Ein angemessenes Angebot an CASE-Tools existiert mittlerweile. So zeigt Abb. 4, daß analog zu dem Verbreitungsgrad der Methoden in den Unternehmen auch eine entsprechende Anzahl der diese Methoden unterstützenden CASE-Tools bereitsteht. Auffällig ist hierbei, daß sich die objektorientierten Methoden einer großen Beliebtheit bei den CASE-Tool-Herstellern erfreuen, obgleich einschränkend bemerkt werden muß, daß nicht alle so angebotenen Tools eine durchgängig objektorientierte Softwareentwicklung unterstützen.

2.2.2 CASE-Werkzeuge

Ein wichtiger Aspekt des Werkzeugeinsatzes innerhalb des Software Factory-Konzepts besteht in dem integrativen Zusammenwirken einzelner Tools, womit eine durchgängige Unterstützung der Phasen des gesamten Systemlebenszyklus intendiert wird. Nach den Ergebnissen der Lehrstuhlstudie trifft dies jedoch nur bei 1% der in die Auswertung einbezogenen Unternehmen zu. 55% der Unternehmen gaben an, CASE in den frühen Phasen des Systemlebenszyklus, d. h. in der Analyse- und Entwurfsphase, einzusetzen, während 39% der Unternehmen CASE-Tools zur Unterstützung der späten Phasen, d. h. zum Zwekke der Realisierung sowie Erprobung und Konsolidierung, nutzen. Demgegenüber bieten die am Markt verfügbaren CASE-Tools weitreichende Funktionalität für fast alle Phasen an. Dies bedeutet jedoch nicht, daß die entsprechenden Komponenten zur Analyse, zum Entwurf und zur Realisierung integrativ zusammenwirken. Dies stellt vielmehr auch heute noch ein ungelöstes Problem für die Hersteller dar, zumal bei der strukturierten Systementwicklung aufgrund der Methodenbrüche eine Durchgängigkeit des Entwicklungsprozesses auch idealtypisch nicht gegeben ist. Diese Anforderung wird durch die objektorientierten CASE-Tools schon eher erfüllt; sie ermöglichen eine iterative Vorgehensweise und bieten eine gute Anbindung an Datenbanken und C++-Compiler. Allerdings haben auch sie ihre Schwächen (z. B. bei der Anbindung an Oberflächen), so daß den gesamten Systemlebenszyklus umfassende Lösungen zur Zeit nicht verfügbar sind.

Neben der Unterstützung aller Phasen des Systemlebenszyklus fordert das Software Factory-Konzept weiterhin die Möglichkeit zur Realisierung der phasenübergreifenden Aktivitäten. Damit sind u. a. die Funktionen des Projektmanagements, der Dokumentation, des Reengineerings sowie der Wiederverwendung angesprochen. Nach den Ergebnissen der Studie besteht der Schwerpunkt der Unterstützung ganz eindeutig in der Dokumentation des Entwicklungsprozesses (ca. 82%). Für diesen Bereich existieren mittlerweile in Wissenschaft und Praxis anerkannte Vorgaben, deren Umsetzung mit Hilfe von CASE-Tools rela-

tiv einfach erfolgen kann. Diese Aussage wird auch mit der gegenwärtig auf dem Markt für CASE angebotenen Funktionalität der Werkzeuge unterstrichen. Eine weitere phasenübergreifende Unterstützung wird für den Bereich des Projektmanagements (ca. 53%) in Anspruch genommen. Allerdings sind entsprechende Funktionen bei nur ca. einem Drittel der untersuchten CASE-Tools implementiert worden. Ca. 41% der befragten CASE-Nutzer gaben weiter an, die Wiederverwendung von Software mit Hilfe von Werkzeugen zumindest rudimentär zu unterstützen. Aus einer genaueren Analyse der Werkzeuge wurde darüber hinaus ersichtlich, daß selbst bei den objektorientierten Werkzeugen keine vernünftigen Funktionen zum Ablegen oder Auffinden von Klassen existieren. Problematisch ist auch der Aspekt des Reengineerings. Zwar würden 35% der befragten Unternehmen eine Computerunterstützung für diese Aktivität in Anspruch nehmen, kritisieren aber diesbezüglich die mangelnde Verfügbarkeit an Werkzeugen. Demgegenüber wird nach Aussage der Anbieter von knapp 60% der CASE-Tools diese Funktionalität angeboten; es muß deshalb vermutet werden, daß viele Hersteller dabei von einem Reengineering der mit ihren CASE-Tools erstellten Anwendungen ausgehen.

Insgesamt kann festgestellt werden, daß eine große Lücke zwischen den Bedürfnissen der Unternehmen und dem Angebot der CASE-Hersteller klafft. Dies gilt insbesondere bei der phasenübergreifenden Unterstützung durch CASE. Aufgrund mangelnder theoretischer Konzepte im Bereich der Wiederverwendung sowie des Reengineerings fehlen über die Werkzeugebene hinaus die notwendigen Grundlagen auf dem Schritt zu einer Software Factory.

3 Gegenwart und Zukunft der Software Factory

Auch wenn der Begriff Software Factory seine Popularität verloren hat, die Idee ist lebendig und ist weiterhin wirksam. Die meisten wissenschaftlichen Arbeiten zur Weiterentwicklung des Software Engineering und die praktischen Bemühungen zur Verbesserung der Softwareherstellung zielen auch heute auf eine Softwareproduktion, die mit Hilfe standardisierter, computergestützter Werkzeuge auf der Basis eines formalisierten, mittels technischer und ökonomischer Kennzahlen kontrollierten Prozesses Software herstellt. Das Ziel Software Factory bleibt also unausgesprochen bestehen. Und auch die Fabrikanalogie wird weiterhin genutzt. Vielfach werden aber neuerdings die Analogien nicht mehr explizit gemacht.

In diesem Kapitel sollen die aktuellen Strömungen in Forschung und Praxis kurz dargestellt werden, die sich einerseits als dem Ziel Software Factory verpflichtet verstehen lassen, und die sich andererseits wesentlich der Fabrikanalogie bedienen. Dabei wird deutlich werden, daß das Ziel einer Softwareentwicklung, die in ihren Prozessen vergleichbar beherrscht wird wie die Herstellung von Serienprodukten, weiter vorangetrieben wird. Allerdings hat sich die Vorstellung über den Weg dorthin gewandelt. Natürlich gibt es auch weiterhin Beiträge zur Automatisierung in der Softwareherstellung, die sich an der Fabrikanalogie orientieren. Zur Zeit steht die Automatisierung aber nicht im Vordergrund, sondern das Management und die Qualität der Prozesse.

3.1 Verbesserung der methodischen Grundlagen

Wachsender Beachtung erfreut sich zur Zeit das Messen in der Softwareherstellung, eine Methode, die in der Zielformulierung der Software Factory angesprochen war, aber lange Zeit nicht die angemessene Aufmerksamkeit und Bearbeitung erfuhr. In den letzten Jahren hat es hier einerseits eine Verschiebung des Interesses von Produktmaßen zu Prozeßmaßen und gleichzeitig eine Verschiebung zur Managementorientierung gegeben. Andererseits sind beachtliche Fortschritte insbesondere im Hinblick auf die Entwicklung von Prozeßmaßen und Vorgehensmodellen für die Einführung von Meßprogrammen erreicht worden [8].

Maße lassen sich zu ganz unterschiedlichen Zwecken einsetzen. Eine im Kontext des Themas Software Factory wichtige Verwendung, die die Autoren dieses Artikels schwerpunktmäßig beschäftigt, ist der Einsatz von Maßen zur rationalen Steuerung und Kontrolle der Softwareentwicklung. Während das Fabrikprodukt nach Menge, Wert und Qualität relativ problemlos gemessen wird und damit genaue und zuverlässige Daten für Kontrolle und Steuerung der Produktion vorliegen, bereitet dies beim Softwareprodukt noch erhebliche Schwierigkeiten. Auch viele Fabrikprozesse lassen sich gut vermessen. In großen Unternehmen spielen quantitative Methoden bei Planung, Steuerung und Kontrolle der Prozesse eine wesentliche Rolle. Softwareprozesse lassen sich dagegen nicht einfach als Wiederholprozesse auffassen, und die Quantifizierung bereitet auch aus diesem Grund Probleme. Aber das aktuell zu beobachtende Bemühen um das Messen in der Softwareherstellung [9] wird zweifellos eine erhebliche Verbesserung der Rationalität des Managements der Softwareherstellung nach sich ziehen.

3.2 Verbesserung der Organisation

Die mit CASE-Einführungen verbundenen Hoffnungen auf Produktivitätserhöhungen haben sich in der Praxis häufig nicht erfüllt. Viele CASE-Einführungen haben mit einer großen Ernüchterung geendet. Die Analyse dieser Erfahrungen [10], zeigt, daß in den meisten Fällen die organisatorischen Voraussetzungen der Automatisierung nicht erfüllt waren und nicht oder nur teilweise hergestellt wurden, wobei dies dann in der Regel auch erst nach Auftreten der ersten Probleme mit deutlicher Verzögerung geschah.

Eine weitere wichtige Erfahrung wurde von den DV-Großanwendern, insbesondere in den USA vom Department of Defence (DoD) beigetragen, deren Erfahrung bei der Auswahl von Softwarelieferanten zu der Forderung nach einem methodischen, wissenschaftlich abgesicherten Auswahlverfahren zu den SEI-Assessments geführt haben. Die Grundlage dieser Assessments ist das CMM, das der Beurteilung eines potentiellen Auftraggebers, seinem Projektmanagement und seiner Ablauforganisation entscheidende Bedeutung zumißt. Es ist zwar ganz auf die Herstellung von Software bezogen, aber es hat seinen Ursprung dennoch in den an der Produktion orientierten Vorstellungen Deming's, dessen Managementstufen als Vorlage dienten.

Schließlich ist noch eine dritte Strömung zu beachten, die sich aus der Übertragung der Erfahrung aus der Serienherstellung ergibt. Die Erfolge des TQM legen die Anwendung dieser Methoden in der Softwareproduktion nahe. Dies hat zu beachtlichen Erfolgen geführt [11]. Dabei zeigt sich aber, daß die Fabrikanalogie zwar nützlich, aber begrenzt ist. So ist die Statistische Prozeßkontrolle (SPC) von der Anwendbarkeit noch relativ weit entfernt, weil die Softwareprozesse nicht in ausreichendem Maße standardisiert sind. Hier sind also auch aus diesem Grund Bemühungen zu erwarten und zu erkennen, die Ablauforganisation bei der Softwareherstellung stärker zu formalisieren.

3.3 Der Beitrag von Standards

Einen wichtigen Beitrag zur Gestaltung der Software Factory liefert die Qualitätsmanagementnorm ISO 9000. Eines ihrer wesentlichen Ziele ist es, bei Kunden Vertrauen zu schaffen in die Fähigkeit eines Lieferanten, vereinbarte Anforderungen zu erfüllen. Sie gehört also wie das CMM in den Problemkreis der Lieferantenauswahl. Die ISO 9000 stößt zwar insbesondere in USA und Japan auch auf heftige Kritik, findet aber dennoch inzwischen weltweite Anerkennung. Sie ist stark an Prozessen der Serienherstellung orientiert, baut also bei ihrer Anwendung auf die Softwareherstellung auch auf der Fabrikanalogie auf. Allerdings haben die Väter der ISO 9000 erkannt, daß die Analogie problematisch ist und zur Übertragung der Norm auf die Softwareherstellung eine spezifische „Interpretationsnorm", die ISO 9000-3, formuliert.

Die große Beachtung, die die ISO 9000 zur Zeit erfährt, wird auch in der Softwareherstellung dazu führen, daß die meisten Unternehmen ihr Qualitätsmanagementsystem nach ISO 9000 zertifizieren lassen. Dadurch wird die ISO 9000 deutlich zur Rezeption von nicht-technischen Ideen der Software Factory beitragen, auch wenn dies von den Vätern der ISO 9000 nicht beabsichtigt war.

3.4 Probleme des Factory model

Haben die negativen Erfahrungen mit der technischen, auf Automatisierung zielenden Sicht der Software Factory den Begriff unberechtigt seiner Popularität beraubt, so haben die Anstrengungen um die Übertragung der stärker organisatorisch ausgerichteten Ansätze CMM, TQM, ISO 9000 die Fabrikanalogie substantiell in Frage gestellt.

Die Schwierigkeiten bei der Anwendung von SPC, einer für das TQM wesentlichen Methode, in der Softwareherstellung zeigen, daß unklar ist, wie Softwareprozesse als Wiederholprozesse aufgefaßt und so weitgehend stabilisiert werden können, daß sie unter statistischer Kontrolle sind. Vermutlich gelingt dies z. Zt. nicht einmal bei gut standardisierten Prozessen wie formalen Inspektionen und Tests [12].

Interessant ist die Kritik, die am CMM entstand. Ein Teil dieser Kritik sieht die dem CMM unterliegende Fabrikanalogie als ein wesentliches Problem und konzentriert sich auf die Analyse charakteristischer Unterschiede, wie z. B. die ganz

unterschiedlichen Risiken, die in einem Produktionsprozeß (Replikationsrisiken) bzw. bei einem Softwareprozeß (Konstruktionsrisiken) kontrolliert werden müssen. Sie fordert daher u. a. eine Ausdehnung der organisatorischen Anforderungen. Es sollen z. B. Anforderungen an das Risikomanagement berücksichtigt werden [13].

3.5 Das Ende der Software Factory?

Es zeigt sich, daß die Fabrikanalogie ihre Grenzen erreicht hat und daß die Andersartigkeit der Softwareherstellung die Überprüfung der häufig naiven Analogie und die Entwicklung eines spezifischen Modells der Softwareherstellung erfordert. Damit ist dann die Idee der Software Factory fruchtbar geworden bis zu ihrer eigenen Überwindung. Die Popularität des Begriffs Software Factory ist nicht nur geschwunden, weil seine immer engere, nur auf Automatisierung zielende Fokussierung sich als unfruchtbar erwiesen hat und er daher als zu kontaminiert gilt, als daß sich sein weiterer Gebrauch noch empfehlen würde. Die Popularität schwindet auch, weil die Grenze der Fruchtbarkeit der Fabrikanalogie für die Gewinnung neuer Erkenntnisse erreicht wurde.

Allerdings sind die Unterschiede von Produktions- und Softwareherstellungsprozessen noch nicht so ausreichend verstanden, daß man die Fabrikanalogie völlig zu den Akten legen könnte. Sie ist solange sinnvoll und fruchtbar, als Ideen, Methoden, Werkzeuge, Regeln und Prinzipien aus der Produktion in die Softwareentwicklung übertragbar sind. Viele Innovationen in der Softwareentwicklung entstanden in der Vergangenheit und werden auch in der Zukunft durch derartige Übertragungen entstehen. Aber es gibt auch erste substantielle Kritik am Fabrikmodell.

Gesucht ist daher ein angemesseneres Modell, bei dem die Prozesse weder Einmalprozesse noch reine Wiederholprozesse sind. Es ist unwahrscheinlich, daß man ein solches vorbildhaftes Modell für den Softwareprozeß von einem anderen Prozeß ableiten kann. Eher wird dann der Softwareprozeß auf der Basis wiederverwendbarer Elemente ein Modell für andere Prozesse, wie z. B. den Konstruktionsprozeß in den klassischen Ingenieurdisziplinen abgeben.

4 Literatur

[1] Vgl. *R. W. Bemer:* Position Papers for Panel Discussion – The Economics of Program Production. In: o. Hrsg.: Information Processing 68. Proceedings. Amsterdam 1969, S. 1626–1627 und *M. D. McIlroy:* Mass Produced Software Components. In: *P. Naur, B. Randell* (Hrsg.): Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Brussels, Scientific Affairs Division, NATO. Brüssel 1969, S. 151–155 zitiert nach *Michael A. Cusumano:* Software Factory. In: *John J. Marciniak* (Hrsg.): Encyclopedia of Software Engineering. New York 1994, S. 1209–1219

[2] Vgl. *Herbert Weber:* From CASE to Software Factories. In: Datamation. April 1, 1989, S. 34–36

[3] Vgl. im folgenden *Michiel van Genuchten:* Towards a Software Factory. Dordrecht, Boston, London 1992, S. 95 ff.

[4] Vgl. *Mary Beth, Bill Curtis, Mark C. Paulk:* Capability Maturity Model for Software, Version 1.1. Technical Report CMU/SEI-93-TR-024. Pittsburgh 1993. Siehe hierzu auch die Ausführungen in Kapitel 3 dieses Beitrages.

[5] Die Grafik baisert auf den Ergebnissen einer CDROM-Recherche in nationalen und internationalen Literaturdatenbanken in der Zentralbibliothek der Universität zu Köln. Berücksichtigt wurden nur wissenschaftliche Fach- und Sachbücher sowie Fachbeiträge in Zeitschriften und Sammelwerken. Die Beiträge mußten sich im Titel oder im Abstract explizit auf das Fabrikmodell der Softwareentwicklung beziehen, um in die Statistik aufgenommen zu werden.

[6] Vgl. im folgenden *Michael A. Cusumano:* Software Factory. In: *John J. Marciniak* (Hrsg.): Encyclopedia of Software Engineering. New York 1994, S. 1209–1219

[7] Die empirische Untersuchung konzentriert sich auf die 100 größten Unternehmen (ohne Banken und Versicherungen) in der BR Deutschland, die mittels eines Fragebogens zu ihrem Softwareentwicklungsprozeß befragt wurden. Die CASE-Tool Marktuntersuchung bezieht 60 CASE-Tools ein, hiervon wurden die 17 am weitesten verbreiteten Produkte auf der Basis einer Fallstudie durch den Lehrstuhl evaluiert. Vgl. *Mellis, Herzwurm, Stelzer* (Hrsg.): Studien zur Systementwicklung des Lehrstuhls für Wirtschaftsinformatik der Universität zu Köln. Bände 2 und 5. Köln 1994

[8] Vgl. *Nicholas Ashley, Martion Bush:* METKIT: Toolskit for Metrics Education. In: IEEE Software. November, 1993, S. 52–54 und *Alison Rowe, Robin Whitty:* Ami: promoting a quantitative approach to software management. In: Software Quality Journal. Nr. 4, 1993, S. 291-296

[9] Vgl. *Shari Lawrence Pfleeger, Hans Dieter Rombach:* Measurement-Based Process Improvement. In: IEEE Software. July, 1994, S. 8–11

[10] Vgl. *Werner Mellis:* Praxiserfahrungen mit CASE – Eine systematische Analyse von Erfahrungsberichten. In: *Werner Mellis, Georg Herzwurm, Dirk Stelzer* (Hrsg.): Studien zur Systementwicklung – Band 2 – CASE-Technologie in Deutschland. Köln 1994, S. 51–97

[11] Vgl. *Robert B. Grady:* Practical Software Metrics for Project Management and Process Improvement. Englewood Cliffs, New Jersey 1992

[12] *Hsiang-Tao Yeh:* Software Process Quality. New York 1993, S. 77

[13] Vgl. *T. B. Bollinger, C. McGowan:* A Critical Look at Software Capability Evaluations. In: IEEE Software. Nr. 6, 1991, S. 25–48