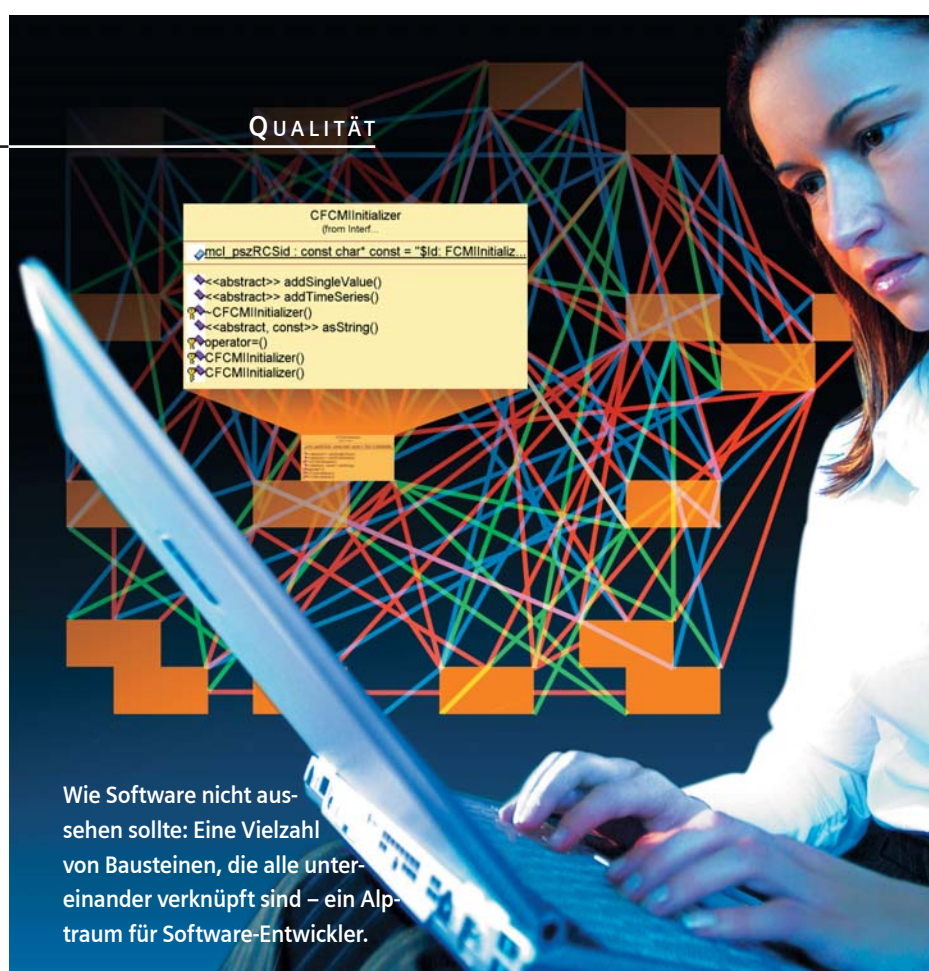


Kurzer Prozess

Bei Siemens sorgen strenge Qualitätskontrollen der Entwicklungsprozesse und automatische Tests dafür, dass Software zuverlässig funktioniert und die Entwicklungskosten im Rahmen bleiben.



Wie Software nicht aussehen sollte: Eine Vielzahl von Bausteinen, die alle untereinander verknüpft sind – ein Alptraum für Software-Entwickler.

Dieter Nuhr leidet. Unter Frauen, unter seiner Mutter, vor allem aber unter Computern. Neulich, erzählt der deutsche Comedystar, habe er einen besonders interessanten Absturz erlebt: „Es ist ein unerwarteter Fehler aufgetreten“, stand da auf dem Bildschirm. „Wirklich beruhigend“, seufzt Nuhr, „dass alle anderen Fehler erwartet werden.“

Nuhrs Publikum brüllt vor Lachen. Wie gut, denken viele, dass es nicht nur mir so geht. Die Probleme bei Bürosoftware kennt jeder, und manchmal sind sie nur mit Galgenhumor zu ertragen. Weniger lustig wäre es, wenn die Software von Autos, Zügen, Industrieanlagen, Medizingeräten oder Kraftwerken ähnliche Fehler produzieren würde und die Steuerrechner keine Befehle mehr annähmen. Solche Befürchtungen drängen sich auf, denn Software übernimmt immer mehr Funktionen der Hardware. So liegt bei Handys der Software-Anteil an der Wertschöpfung bei über 70 Prozent. Vom Siemens-Handy S25 bis zum S65 hat sich die Zahl der Programmzeilen verfünffacht: Sie übertrifft schon die Steuerung von Weltraumraketen der 1960er Jahre.

Viele Programmzeilen, viele Fehler? „Software wird immer komplexer“, gibt Dr. Frances Paulisch zu. Das sage aber nicht unbedingt

etwas über die Fehlerhäufigkeit aus. Paulisch leitet die 1996 gegründete Software Initiative von Siemens. Dabei geht es nicht in erster Linie darum, Fehler im Programmcode ausmerzen, sondern darum, die 30.000 Software-Entwickler von Siemens durch die Etablierung optimaler Prozesse zu unterstützen.

Fragenkatalog für Software-Qualität. Die Abteilung Software & Engineering / Prozesse von Siemens Corporate Technology (CT) – ein international anerkanntes Assessment-Center für Software-Prozesse – nutzt dazu einen Katalog mit 250 Fragen. Im Interview mit den Schlüsselpersonen eines Software-Projekts loten zwei Mitarbeiter die Stärken und Schwächen in den Arbeitsabläufen aus. Welche Kompetenzen benötigt der Projektleiter, wie sehen die Qualitätssicherungsmaßnahmen aus und so weiter. Aus den Antworten wird ein Reifeprofil mit einer Reifegradstufe von 1 bis 5 abgeleitet. Als Messlatte dient das internationale Capability Maturity Model Integration, CMMI. Im CMMI steht der Reifegrad 1 für völlig unorganisierte Prozesse, 5 hingegen für eine Software-Entwicklung, die auf Basis festgelegter Metriken kontinuierlich optimiert wird. Die Software Initiative strebt bis 2005

für die F&E-Einheiten von Siemens einen Reifegrad von 3 an, was den internationalen Bestrebungen entspricht. In etlichen Geschäftsbereichen wurde dieses Ziel bereits erreicht oder gar übertroffen.

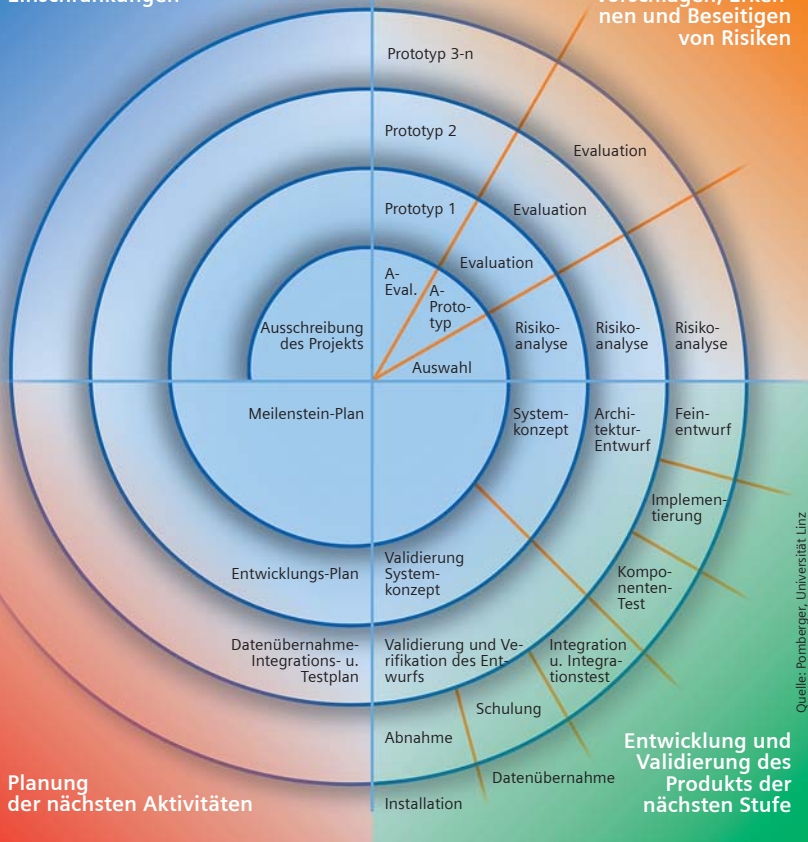
Nach der Auswertung erhalten die Geprüften ein Feedback und wenn nötig Schulungen. Dort werden unter anderem Review-Techniken gelernt, bei denen ein Software-Ingenieur seine Arbeit vor Kollegen präsentieren muss. Auch wenn anfänglich gemurrt werde, seien die Beratenen hinterher immer zufrieden. „Das sind 20 Prozent Technik und 80 Prozent Psychologie“, verrät Ludger Meyer, Leiter der Abteilung. Seine 36 Mitarbeiter führen weltweit Assessments für alle Software-Einheiten von Siemens durch.

Ein hohes CMMI-Level ist allerdings keine Garantie, dass eine Software nie versagt. Doch es gibt einen messbaren Zusammenhang zwischen CMMI-Level einerseits und Software-Qualität und Kosten andererseits. „Wenn die Prozesse stimmen, kann man Kosten und Zeitaufwand besser einschätzen“, sagt Frances Paulisch. Um Level 4 oder höher zu erreichen, was bei sicherheitsrelevanten Anwendungen sinnvoll ist, müssen verstärkt Metriken verwendet werden. Das sind Messverfahren, die



Festlegung von Zielen, Nebenbedingungen und Einschränkungen

Erarbeiten und Bewerten von Lösungsvorschlägen, Erkennen und Beseitigen von Risiken



Planung der nächsten Aktivitäten

Entwicklung und Validierung des Produkts der nächsten Stufe

Pombergers Spiralmodell der Software-Entwicklung: Prototypen werden in Rückkopplungsschleifen immer wieder verbessert und erweitert.

Für Stefan Hohrein ist Software aber nur eine Seite der Medaille. Hohrein ist Leiter der System-und-Software-Initiative beim Automobilzulieferer Siemens VDO in Regensburg. Schon das „System“ in Hohreins Funktionsbezeichnung deutet an, dass Siemens VDO parallel zur Software-Entwicklung die Disziplinen System-Engineering, Hardware- und Mechanik-Entwicklung im Blick hat. Für Siemens hat das Vorbildcharakter, denn im Konzern wird die meiste Software für „eingebettete“ Systeme geschrieben, bei denen die Software eng mit der Hardware verknüpft ist, wie in der Waschmaschine oder einem Handy. In Autos wird Hardware zunehmend durch Software ersetzt, weil sie flexibler ist: Eine Studie der Mercer Group sagt einen Wertanteil der Software von 13 Prozent im Jahr 2010 voraus, 2000 waren es nur vier Prozent.

Die System- und Softwareentwicklung verläuft bei Siemens VDO nach festen Prozessen, wobei Art und Verlauf der Tests je nach Kundenanforderung definiert werden. Zunächst werden die einzelnen Programmteile unabhängig voneinander geprüft. Danach kommt der Integrationstest, der das reibungslose Zusammenspiel der einzelnen Funktionen sicherstellt. Letzte Hürde ist der Systemtest, der im kompletten Gerät unter möglichst realen Umgebungsbedingungen stattfindet. Um die Komplexität unter Kontrolle zu halten, arbeiten führende Auto- und Elektronikhersteller an einer Art Betriebssystem mit einheitlichen Schnittstellen, bei dem man bewährte Programmmodule andocken und wieder verwenden kann (S. 53). Die Zahl der Steuergeräte soll so von heute 70 in Oberklasse-Limousinen auf etwa 20 gesenkt werden.

Programme testen Programme. Eine zunehmend wichtige Rolle spielen automatisierte Tests, die zu einem frühen Zeitpunkt der Entwicklung stattfinden. Einfache Syntaxfehler im Code finden schon die Compiler, die ein Programm in Maschinenbefehle übersetzen. Ob jedoch alle Programmteile richtig zusammenwirken, sollen neue Testverfahren prüfen. Das Fraunhofer-Institut für Rechnerarchitektur und Softwaretechnik in Berlin und jenes für Experimentelles Software Engineering in Kaiserslautern haben hierzu Techniken unter dem

die Qualität des Programmcodes definieren, zum Beispiel die Fehlerdichte. Ein weiteres Level-4-Kriterium ist die Fähigkeit zur Software-Modularität und Wiederverwendbarkeit, was die Kosten und die Komplexität senkt.

Module sichern Qualität. Modularität hat sich auch Siemens Medical Solutions auf die Fahnen geschrieben. Statt wie früher für jedes Produkt eine neue Software zu entwickeln, setzen die Erlanger heute bei Computer- und Kernspintomographen und weiteren bildgebenden Geräten auf die Softwareplattform *syngo*. Die Plattformstrategie verringert die Fehleranfälligkeit. Bei den Prozessen wenden die Medizintechniker das V-Modell an, das vom „Wasserfallmodell“ abgeleitet ist: Spezifikation, Design, Implementierung und Tests laufen nacheinander ab, jedoch mit Überlappungen, um Zeit zu sparen.

Moderner, aber noch nicht so lange im Einsatz, sind inkrementelle Verfahren, bei denen der ganze Prozess in mehrere Mini-Was-

serfälle mit vorher definierten Schnittstellen zerlegt wird. Gustav Pomberger, Professor für Software-Engineering an der Universität Linz, erforscht solche alternativen Entwicklungsprozesse und arbeitet dabei mit Siemens zusammen. Pomberger favorisiert das Prototyping. Bei dieser Strategie erstellen die Entwickler schon sehr früh einen testbaren Prototypen der Software, der längst nicht alle Funktionen haben muss, der aber dem Auftraggeber ein Gefühl dafür gibt, ob das Programm einmal das leisten kann, was es soll. Sind Änderungen nötig – und das sind sie häufig in dieser Phase – wird der Prototyp in Rückkopplungsschleifen angepasst und erweitert. Pomberger hat die Idee des Prototyping für bestimmte Entscheidungsprozesse zu einem spiralartigen Prozessmodell verfeinert (Grafik oben). Ähnlich einem Schneckenhaus arbeiten sich die Software-Ingenieure in mehreren Schleifen aus der Mitte (Ausschreibung des Projekts) über diverse Prototypen, Evaluationen, Konzepte und Tests nach außen.

*Der Code-Inspector – ein **Analysetool** – warnt frühzeitig vor Fehlern, spart Zeit und Kosten.*

Namen Quasar entwickelt, die sehr frühzeitige Tests von Software ermöglichen.

Erstes Testobjekt war eine Autotür mit integrierten Tasten zur Sitzverstellung. Zunächst half Quasar, die Anforderungen des Herstellers DaimlerChrysler in übersichtlichen Diagrammen festzuhalten. So darf die Sitzverstellung höchstens bei Schritttempo aktiv sein, was die Kopplung mit dem Tacho erfordert. Die mehreren hundert Funktionskombinationen werden simuliert und auf Konsistenz getestet. Erst danach wird die Software erstellt. Auch später hilft Quasar: Es simuliert Sensoren und aktiviert automatisch alle Funktionen des Mikrocontrollers. Besonders sicherheitsrelevante Funktionen lassen sich ohne solche Testwerkzeuge gar nicht entwickeln – etwa eine elektrische Lenkung. „Die Hersteller müssen beweisen, dass dieses Steer-by-wire ebenso zuverlässig ist wie eine mechanische Lenkung“, fordert Prof. Holger Schlingloff, Quasar-Projektmanager.

Inspektor für Codes. An Hilfsmitteln für Programmierer arbeitet auch die Abteilung Software Engineering bei Siemens Corporate Research in Princeton. Sie hat ein statistisches Analysetool, den Code-Inspector, für die wichtigsten Programmiersprachen C (20% Anteil bei Siemens), C++ (30%), Java (12%) und andere entwickelt, der den Quellcode nach „Bugs“ durchforstet. Der Schnüffler hat sich schon bei etlichen Geschäftsbereichen bewährt und durch frühes Warnen vor Fehlern Zeit und Kosten gespart. Außerdem liefert der Code-Inspector spezielle Kennzahlen, die von den Kunden nachgefragt werden. So hat das Eisenbahn-Bundesamt einen Katalog mit Qualitätskriterien für Software, die in C++ geschrieben ist. Mit dem Code-Inspector schaffte es Siemens Transportation Systems, den Qualitätsnachweis zu den halben Kosten zu erbringen.

Eine logische Weiterentwicklung wäre, wenn der Code-Inspector Fehler gleich selbstständig ausmerzen würde. „Das ginge heute schon für einige Qualitätskriterien“, sagt Jean Hartmann, einer der Väter des Code-Inspectors in Princeton, „aber als ehemaliger Software-Entwickler würde ich mir auch nicht gerne von einer Maschine ins Handwerk pfeuschen lassen.“

■ Bernd Müller