Das Mauerblümchen
Software-Wartung

Von Christoph Bommer

Software-Wartung ist nicht sexy – trotzdem ist sie eine wichtige und real existierende Aufgabe.

Entsprechend dem Mauerblümchendasein der Software-Wartung gibt es nur wenige Studien, Methoden und Werkzeuge zum Thema. Ziel dieses Artikels ist es deshalb, eine Übersicht zu geben.

Wo liegen also die Herausforderungen in der Software-Wartung? In der Regel hat man es mit grossen, komplexen Systemen zu tun, die schlecht oder gar nicht dokumentiert sind. Zudem sind die «geistigen Väter» der ersten Stunde des Systems meist nicht mehr greifbar, und die ursprüngliche Architektur ist im Dschungel der Erweiterungen und Fehlerbehebungen nicht mal mehr in Ansätzen erkennbar.

Fehlende gemeinsame Sprache

Zusätzlich fällt auf, dass in der Software-Wartung eine gemeinsame Sprache, wie sie in der Software-Entwicklung bereits existiert, fehlt. Wenn uns ein Software-Ingenieur sagt, er sei zur Zeit mit dem Design beschäftigt, so haben wir eine klare Vorstellung davon, welche Aufgaben er bisher erledigt hat, was er im Moment gerade tut und was noch vor ihm liegt. Genauso verhält es sich, wenn uns jemand sagt, er sei Projektleiter, Konfigurationsmanager oder Architekturverantwortlicher. In all diesen Fällen haben wir mehr oder weniger dasselbe Bild von diesen Rollen. Wie sieht es dagegen bei der Software-Wartung aus? Da wird die Verständigung plötzlich schwierig. Es fällt einem nur schon schwer, klar zwischen Tätigkeiten der Entwicklung und der Software-Wartung zu unterscheiden. Wo liegt genau die Grenze? In welche Gebiete und Themen lässt sich die Software-Wartung überhaupt unterteilen?

Definition

Häufig und irrtümlicherweise wird Software-Wartung lediglich mit dem Beheben von Fehlern verbunden.

Wie die Norm aber zeigt, hat man darunter deutlich mehr zu verstehen. Gemäss IEEE 1219-98 bezeichnet man als Wartung die Veränderung eines Software-Produkts nach dessen Auslieferung – zwecks Fehlerbehebung, Leistungssteigerung oder Anpassungen an die veränderte Umgebung. Daraus abgeleitet können folgende vier Arten der Wartung unterschieden werden (siehe Grafik):

- **Korrigierende Wartung:** Die korrigierende Wartung deckt die klassische Vorstellung der Wartung ab und beschäftigt sich mit der Behebung von Fehlern, die nach der Auslieferung der Software aufgetreten sind. Da es sich hier um eine reaktive Massnahme handelt, ist sie oft nicht planbar (siehe die jüngsten Ereignisse im Eisenbahnbereich).
- **Präventive Wartung:** In der präventiven Wartung geht es darum, latente Fehler nach der Auslieferung der Software zu beheben, bevor sie im Feld als effektive Fehler aufgetreten sind. Dies ist eine proaktive Massnahme und daher planbar.
- **Adaptive Wartung:** Während der Lebensdauer einer Software kann sich die ursprüngliche Systemumgebung verändern. Damit die Applikation weiterhin für den Endanwender nützlich bleibt, muss man sie diesen Veränderungen anpassen.
- **Perfektionierende Wartung:** Das Ziel der perfektionierenden Wartung ist die Verbesserung der Software. Dies kann gemäss IEEE 1219-98 die Performance oder andere Attribute betreffen. Die Performance zielt auf die Schnelligkeit des Systems, aber auch auf die Effizienz der Nutzung von Systemressourcen ab. Hingegen öffnet der Begriff «andere Attribute» ein weites Feld von Möglichkeiten, wie z.B. die Wartbarkeit, die (Re-) Dokumentation, aber auch Strukturverbesserungen im Rahmen von Refactoring bis hin zu Reengineering.

Zusammenfassend bezeichnet die Software-Wartung als Tätigkeit eine Änderung an einem Software-System nach dessen Auslieferung, als Prozess die Schritte, die in einem Wartungsfall sequenziell durchzuführen sind, und als Phase den Abschnitt des Lebenszyklus von der Auslieferung bis zur Stillegung.

Wartungskosten

Wenn es um die Wartungskosten geht, hört man als verantwortlicher Manager schnell Vorwürfe wie «die Wartung ist das Bermuda-Dreieck der Entwicklung», denn das Geld verschwindet einfach auf mysteriöse Art und Weise. Wenn man anschliessend die Situation im stillen Kämmerlein nüchtern betrachtet, fällt es vielen verantwortlichen Wartungsmanagern doch schwer, klar und deutlich zu zeigen, wofür das Geld genau ausgegeben wurde und wird. Wahrscheinlich wissen die meisten gerade noch, wie viele Fehler sie damit behoben haben. Wissen sie aber auch, wieviel Prozent ihres Budgets in korrigierende Wartung fliesst und wieviel beispielsweise in präventive Wartung? Daraus abgeleitet stellt sich etwa die Frage, ob es sich überhaupt lohnt, in präventive Wartung zu investieren. Eigentlich sollte es billiger sein, einen Fehler im eigenen Haus zu entdecken, zu beheben und den Zeitpunkt der damit verbundenen Ausbreitung im Feld selber zu bestimmen – abgesehen vom Imageverlust beim Kunden. Als erstes gilt es deshalb, die eigene Kostentransparenz rigoros herzustellen. Erst darauf können weitere Aktionen aufbauen.

Des weiteren werden Entwicklungsprojekte typischerweise aufmerksamer verfolgt als Wartungsprojekte. Dies birgt die Gefahr, dass der Entwicklungsprojektleiter, der zur Einhaltung der Projektkosten angehalten ist, sein Projekt möglichst schnell abschliessen und der Wartung übergeben möchte.

Dabei wird sein Vorhaben durch die ungenügende Regelung, wie der

Die vier Wartungskategorien

	Korrektur	Verbesserung
Reaktiv	Korrigierende Wartung	Adaptive Wartung
Proaktiv	Präventive Wartung	Perfektionierende Wartung

Quelle: Universität Zürich

Wartung ist die Veränderung eines Softwareprodukts nach dessen Auslieferung.

Infoweek.ch

Übergang von der Entwicklung zur Wartung auszusehen hat, unterstützt. Um diese ungerechtfertigten Folgeaufwände – man spricht in diesem Zusammenhang auch von der berühmten «grünen Banane» – in der Wartung zu vermeiden, braucht es klare Kriterien für diesen Übergang.

Zuletzt sind da noch die «Eh-da Kosten». Leistet man sich ein separates Wartungsteam, so hat dies im Gegensatz zu einem Entwicklungsprojekt meist eine fixe Grösse. Diese Leute sind, völlig unabhängig von der Anzahl Fehler, die im Feld auftreten oder eben auch nicht, «eh da». Mit dieser Teamgrösse kann man also bloss auf eine aktuelle Situation, die eben schlecht planbar ist, reagieren.

Wartung im Kontext der Mitarbeiter

Ungerechtfertigerweise gilt die Wartung als unattraktiv und als das «Sibirien der Informatik». Schliesslich hat man es oft mit veralteten Systemen (zum Beispiel Mainframes) und Programmiersprachen (zum Beispiel Cobol) zu tun. Ausserdem wird eher renoviert als kreiert. Vor diesem Hintergrund tragen viele Manager das ihre dazu bei und schieben «ausgediente» Software-Entwickler in die Wartung ab. Dies ist natürlich der Gewinnung von guten Mitarbeitern für diese wichtige Aufgabe wenig förderlich. Aber genau solche Mitarbeiter sind sehr wichtig!

Während ein hervorragender Mitarbeiter wenige Stunden braucht, um einen Fehler im System zu finden, benötigt ein schwacher bald einmal einige Tage. Zum anderen konzentriert sich das Wissen häufig auf die wenigen guten Mitarbeiter, die somit den Status von «Gurus» erreichen. Dies hat zwei Nachteile. Erstens schafft man sich eine grosse Abhängigkeit von diesen Gurus und kann einen Abgang nur schwer verkraften. Zweitens laufen viele Problemklärungen und technische Diskussionen über diese wenigen Schlüsselpersonen, die so unweigerlich zum Flaschenhals werden. Um solchen Effekten entgegenzuwirken, ist es wichtig, dass die Software-Wartung nicht abgewertet, sondern im Gegenteil mit den Besten besetzt wird.

Software-Systeme werden durch häufige Änderungen tendenziell komplexer und umfangreicher. Dadurch verringert sich deren Verständlichkeit und Wartbarkeit. Um dem entgegenzuwirken, sind entsprechende Codehygiene-Massnahmen einzuplanen und durchzuführen.

Codehygiene

Diese können vom einfachen Refactoring bis zum Reengineering gehen. Früher führte man ganz nach dem Motto «never touch a running system» möglichst wenige Veränderungen durch. Heute unterstützen die neuen Werkzeuge Refactoring-Vorhaben mit automatisierten und sicheren Codetransfers. Geeignete Metrik-Tools können zusätzlich die Sanierungsmassnahmen begleiten und über deren Erfolg oder Misserfolg Auskunft geben. All dies ist aber nur möglich, wenn das Management mitspielt.

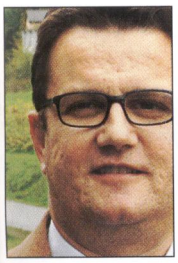
Fazit

Abschliessend lässt sich sagen, dass das Gebiet der Software-Wartung zunehmend an Bedeutung gewinnen wird und es sich somit lohnt, sich damit frühzeitig und umfassend auseinanderzusetzen.

Christoph Bommer ist Leiter der Entwicklung Leittechnik bei Siemens Schweiz AG Transportation Systems.

Infoweek.ch • Infoweek.ch • Infoweek.ch • Infoweek.ch • Infoweek.ch • Infoweek.ch • Infoweek.ch • Infoweek.ch

SYSTEME & MANAGEMENT: REINHARD RIEDL



Praxistaugliche Theorien statt theorieabstinente Praxisnähe

Die bange Frage lautet: Was wird die nächste grosse Attacke sein? Denn IT-Sicherheit ist keine statische und absolute Eigenschaft, sondern hängt vom dynamisch sich ändernden, relativen Kräfteverhältnis zwischen Angreifern und Verteidigern ab. Theoretisch sind die meisten IT-Systeme hoch unsicher. Die Ursachen sind unter anderem Bastel-Architekturen, Implementationsfehler und fehlendes Auditing. Ein umfängliches Beseitigen dieser Sicherheitsrisiken würde die IT-Security-Budgets der meisten Firmen um ein Vielfaches übersteigen. Praktisch ist es zudem sowieso relativ schwierig, in die ungeordnete Ansammlung von Sicherheitslöchern mit Erfolg einzudringen und tatsächlich Schaden anzurichten. Für erfolgreiche Angriffe sind neue Ideen notwendig, die nicht alle Tage entstehen. Darum scheint es sinnvoll, bei der Weiterentwicklung der Unternehmens-IT-Sicherheit sich darauf zu beschränken, die nächsten Angriffsideen zu antizipieren und die von Hackern anvisierten Löcher zu stopfen.

Doch das Vorhersehen von Angriffstechniken ist schwierig. Es würde einen offeneren Erfahrungsaustausch und vermehrte geheimdienstliche Aufklärungsarbeit verlangen. Beides ist mit der Geschäftskultur vieler Firmen inkompatibel: Zum einen glauben noch immer viele Praktiker an Security durch Obscurity, und zum anderen birgt ein Mehr an Schnüffelstaat und international vernetzter Schnüffelei viele diffuse Risiken. Ausserdem würde eine ausschliesslich auf Antizipation setzende Sicherheitsstrategie die Initiative völlig den Angreifern überlassen und das Risiko schaffen, Architektur-Flickwerk hervorzubringen.

Sinnvoller ist es deshalb, Investitionen in die IT-Sicherheit so auszuwählen, dass sie in möglichst vielen zukünftigen Angriffsszenarien von Nutzen sind und ein schnelles Reagieren ermöglichen. Dafür müssen erstens transparente Architekturen implementiert werden, die die Designprinzipien für verteilte Systeme konsequent umsetzen. Zweitens muss die Infrastruktur zur Systemüberwachung und zur effizienten und effektiven Aufbereitung und Nutzung der Überwachungsdaten ausgebaut werden. Ein gutes Anschauungsbeispiel für die Folgen des Fehlens einer solchen Infrastruktur ist der SBB-Blackout. Drittens sind organisatorische Massnahmen notwendig: Das Top-Management muss Prozesse zum Management des operationellen Risikos installieren. Die Verantwortlichkeiten im mittleren Management müssen klar definiert werden. Und die Sicherheitskultur im ganzen Unternehmen muss permanent weitergebildet werden, durch regelmässige Schulungen und die systematische Kontrolle des konkreten Verhaltens. Alle weiteren Massnahmen sollten dann nach der ökonomischen Bedeutung dessen, was geschützt wird, priorisiert werden.

Und es gibt noch eine weitere Investition, die sehr lohnenswert sein kann: Um die vorhandenen formalen Methoden nutzen zu können, sollte man mehr in die Ausbildung von Mitarbeitern zu investieren, die gut mit formalen Modellierungswerkzeugen umgehen können. Bislang scheitert dies daran, dass einerseits die Theoretiker zu wenig Fachbereichsverständnis und andererseits die Informatiker und Fachexperten zu wenig Theorieverständnis besitzen. Die Konsequenz ist, dass die einen viel zu feingranular modellieren und die beiden anderen die formalen Instrumente ablehnen. Sie berufen sich darauf, dass man auch bisher ohne Methoden sicher war. Hier sind auch die Ausbildungszentren aufgefordert, statt der gerne gewünschten theorieabstinenten Praxisnähe praxistaugliche Theorie zu vermitteln.

Reinhard Riedl ist Mathematiker und leitet die E-Government Research Group am Institut für Informatik der Universität Zürich.