

In dieser Ausgabe:

Die Rolle der Wartung in der Softwareentwicklung *Entwicklungsmethodologien bestimmen und definieren Wartungskonzepte*

Agile und Waterfall konfrontieren den Softwareentwickler mit dem Problem der Wartung und Pflege im Entwicklungsprozess. Gibt es dafür eine effektive Lösung? Hier finden Sie Tipps und Ideen für die erfolgreiche Entwicklungsarbeit und das Projektmanagement dieser Frage.

Sicherheitsaspekte als integraler Bestandteil der Softwarearchitektur

Der Trend zu webbasierenden Software as a Service (SaaS)-Lösungen bringt eine wachsende Relevanz der Sicherheitsfrage mit sich. Von großer Bedeutung ist die Berücksichtigung der Angriffsgefahren und der Grundsätze der Sicherheitsgewährleistung.

Die Rolle der Wartung in der Softwareentwicklung *Entwicklungsmethodologien bestimmen und definieren Wartungskonzepte*



Anton Dechko,
 Director of Business
 Development

Die beliebtesten Prozessmodelle der Softwareentwicklung sind momentan zweifellos Agile und Waterfall. Diese in unzähligen Projekten bewährten Modelle oder Methodologien dürften sich an zahlreichen Stellen und in vielen Details unterscheiden. Der wesentliche und fundamentale Unterschied besteht jedoch darin, dass Agile Projekte in Abschnitte eingeteilt werden, die klein genug sind, um überwiegend verbal kommuniziert zu werden. Feedback kommt in einem derartig strukturierten Prozess dann ebenfalls direkt im laufenden Arbeitsprozess. Waterfall dagegen ist auf eine detaillierte Dokumentation angewiesen. In dieser geht es dann natürlich primär um die festgelegten Anforderungen und deren

technische Umsetzung. Dem liegt die Vorstellung zugrunde, für alle Fälle ein Papier vorzufinden, mit dem sich eine Applikation optimal erstellen und umsetzen lässt.

Lassen wir die Frage einmal beiseite, welcher Zugang für welches Projekt der praktischere ist. An diesem Punkt muss man jedoch feststellen, dass beide Methoden eine fundamentale Gemeinsamkeit besitzen. Beide betrachten den Prozess der Softwareentwicklung unter dem Aspekt der Neuentwicklung, der Erstellung einer von Grund auf neuen Software. In der Realität dann sind die meisten Softwareprojekte, zumindest für am Markt erfolgreich etablierte Applikationen, immer auch mit der laufenden Wartung und Pflege konfrontiert. Und dies parallel zur Erweiterung und der Entwicklung neuer Funktionen. Beschäftigen wir uns mit diesem Punkt etwas genauer, kommen wir zu einer wichtigen Frage: Wie lässt sich die Entwicklung neuer Funktionen effektiv mit der permanenten Behebung von Fehlern, mit Aufgaben der Wartung und Pflege kombinieren?

Existieren zwei Teams, eines für die Entwicklung und eines für die Wartung, wird das Problem entschärft, aber nicht vermieden oder gar behoben. Zur Sicherstellung einer effizienten Wartung benötigt man Mitarbeiter die den Code wirklich kennen. Und zwar auch den der neu entwickelten Funktionen. Nun könnte man auf die Idee kommen, das Wartungsteam einfach „on demand“, also je nach Bedarf zu diesem Zweck um neue Mitglieder zu erweitern. Um Spitzen in der Arbeitsbelastung abzufedern oder die Geschwindigkeit im Team zu steigern, um Termine doch noch zu halten. In den meisten Fällen ist es damit aber nicht getan. Und effektiv ist ein derartiges Vorgehen in den meisten Fällen auch nicht. Der Einsatz des ganzen Entwicklungsteams für drängende Aufgaben der Fehlerbehebungen ist ebenfalls eine umstrittene

Option. Denn so leidet immer die Planung und Umsetzung neuer Funktionen. Also die eigentliche Aufgabe dieses Teams.

Ein Lösungsansatz (nicht nur) für Waterfall

In der konkreten Entwicklungsarbeit von SaM Solutions haben sich einige praktische Hinweise für die Abstimmung und Aufteilung der Arbeit in Entwicklungs- und Wartungsteams bewährt. Dies gilt insbesondere, aber nicht nur, für nach der Waterfall-Methode organisierte Projekte.

1. Zuweisung eines dedizierten Wartungsteams

Ressourcen für die Softwarewartung müssen auf irgendeine Art und Weise vorgehalten werden. Die Erfahrung von SaM Solutions zeigt, dass der praktikabelste Weg darin besteht, ein separates Team als feste Größe abzutrennen. Mindestens ein bis zwei Personen, wobei sich die genaue Größe natürlich nach dem Umfang des konkreten Projektes richtet. Im Zusammenhang mit einem kalkulierten Festpreis kann man hier von einem Risikobudget sprechen. Möglicherweise wird es nicht vollständig genutzt, schlimmstenfalls reicht es nicht aus. Letzteres ist üblicherweise Kennzeichen für ein echtes technisches Problem, nicht nur für ein Problem im Management des Projektes. Der Rest des Teams fährt damit fort, neue Funktionen und Erweiterungen für das eigentliche Softwarebasisprojekt zu entwickeln. Regelmäßig, beispielsweise monatlich oder wöchentlich, werden die Aufgaben und die Belastungen des Wartungsteams dann analysiert. Je nach den anstehenden Aufgaben kann es dann entsprechend reduziert oder vergrößert werden.

2. Entwicklungsteam und Wartungsteam sollten eng beieinander arbeiten

Dabei besonders wichtig: Die Mauern zwischen den Teams einreißen! Nur bei einer funktionierenden wechselseitigen Kommunikation mit konstantem Feedback kann die Zusammenarbeit funktionieren. Festgestellte Probleme im Softwarecode sind kritische Informationen für anstehende Entscheidungen bezüglich des technischen Designs oder Änderungen im geplanten Testablauf.

3. Rotation von Entwicklern zwischen verschiedenen Teams

Es ist nützlich und hilfreich, Teammitglieder aus Entwicklungs- und Wartungsteams bisweilen rotieren zu lassen. Beispielsweise alle drei oder sechs Monate. Dies ist gut für die Motivation. Und noch besser ist es für den Wissenstransfer.

Der hier skizzierte und vorgeschlagene Zugang eignet sich auch für Projekte, die nach der Agile- und Scrum-Methodologie organisiert sind. Obwohl das Vorgehen zwei Scrum-Prinzipien verletzt. Erstens sollte es hier multifunktionale Teams geben, also keine

Spezialisten. Und zweitens sollten die Teams sich selbst organisieren, die Entwickler also entscheiden, wer welche Aufgabe übernimmt. Um dem Rechnung zu tragen, bietet sich folgendes Vorgehen an:

1. Es sollte keine Teilung der Teams geben. Stattdessen sollten Ressourcen, Personen und ein Budget, für die generellen Aufgaben der Wartung vorgehalten werden. Idealerweise in Form von reservierten Stunden.

2. Zweitens sollte es eine Regelung zur umgehenden Reaktion auf entstandene Probleme geben. Dies ist ein unumgänglicher Punkt. Es sollte zudem eine Regelung sein, nach der sich jemand sofort um das Problem kümmert. Und nicht erst dann, wenn und wie das Team entschieden hat. Gefragt ist hier also eine Art von Service Level Agreements (SLA) zwischen den Entwicklern und dem Management des Supports.

Die Kehrseite eines derartigen Agile-Anspruchs an die Wartung ist auch klar. Es ist nicht immer leicht, jederzeit und schnell zwischen verschiedenen Aufgaben zu wechseln. Von der laufenden Entwicklungsarbeit zur Fehlerbehebung und wieder zurück. Leicht kann es dabei daher zu Ineffizienzen kommen. Eine andere

typische Herausforderung ist die Handhabung eines wiederkehrenden Problems durch den immer gleichen Entwickler. Geschieht dies regelmäßig, führt es zu technischer Spezialisierung oder der Hoheit über bestimmten Code und widerspricht den Agile-Prinzipien. Will man dies nicht, muss man akzeptieren, dass Entwickler eine Weile brauchen ehe sie Fahrt aufnehmen, erst ein Problem kennenlernen, sich anschauen was bereits gemacht wurde und warum dies nicht funktioniert hat.

Wie man es dreht und wendet, die effiziente Organisation der Wartung in einem Projekt führt zu „technischen“ Fragen, die nicht durch das Prozessmodell abgedeckt sind. Und dies wesentlich öfter als bei der Anpassung des Codes an unterschiedliche Versionen, Plattformen oder Anwendergruppen, um nur einige Beispiele aus dem Entwicklungsprozess zu nennen.

Letztlich lassen sich alle diese Fragen nur in einem konkreten Projekt und einem konkreten Unternehmensumfeld beantworten. Denn die Komplexität und die Geschichte eines Softwareproduktes, das Geschäftsumfeld und die ganz konkreten Entwicklungspraktiken einer Organisation müssen immer berücksichtigt werden. SaM Solutions beschäftigt sich mit der Beantwortung dieser Fragen für seine Kunden im täglichen Geschäft seit mittlerweile mehr als 16 Jahren. ■

Sicherheitsaspekte als integraler Bestandteil der Softwarearchitektur



Denis Koloshko
Chief .NET Technologist

Zur Beurteilung der Qualität von Architektur und Design einer Softwareanwendung dienen üblicherweise so umfassende Kriterien wie die Benutzer- und Supportfreundlichkeit oder die Zuverlässigkeit. Oft wird auch ganz allgemein von der Leistungsfähigkeit gesprochen. Ein Aspekt jedoch, die Sicherheit, wird an dieser Stelle eher selten genannt. Ein bisweilen teurer Fehler.

Die Ausgangssituation

In den letzten Jahren hat es eine sehr starke Zunahme der Internetaktivitäten gegeben. Verbunden damit ist die Anwendung von Applikationen im Web stark gestiegen.

Die Zahl der Transaktionen über das Web mit sensitiven Inhalten oder Finanzinformationen hat sich drastisch erhöht. Und der Trend zu webbasierenden Software as a Service (SaaS)-Anwendungen wird die Volumina weiter steigen lassen. Auch soziale Netzwerke tragen ihren Teil dazu bei. Zudem handelt es sich bei diesen um Online-Plattformen, auf denen die Menschen zahlreiche private und vertrauliche Informationen hinterlegen.

Mit dem Anstieg der Internetaktivitäten sind auch die Gefährdungen und der Missbrauch gewachsen. Jede populäre Website dürfte in der Vergangenheit schon erfolgreiche Attacken durch kriminelle Nutzer erlebt haben. Und auch die Größen der Branche mussten immer wieder die Verwundbarkeit und den Missbrauch ihrer Softwarelösungen und Online-Produkte zugestehen.

Diese Verwundbarkeit der Software und ihre Sicherheitsmängel haben unterschiedliche Gründe. Zudem nutzen Angreifer bisweilen auch sehr anspruchsvolle und leistungsstarke Technologien für ihre Attacken. Aber 90 Prozent der Angriffe basieren auf fünf einfachen Basisverfahren:

1. SQL-Injection: Eine Technik zur Injektion von Schadcode, die Sicherheitsmängel auf der Datenbankebene einer Anwendung ausnutzt.

2. XSS-Injection: Eine Art der Verwundbarkeit, die sich typischerweise bei Web-Anwendungen findet. Bösartiger Script-Code wird auf der Client-Seite eingeschleust.

3. Cross-Site-Request-Forgery: Das Ausnutzen einer Sicherheitslücke in Webanwendungen. Dem Anwender wird ein vertrauenswürdiges Umfeld vorgetäuscht, um an Informationen von ihm zu gelangen.

4. Url-Injection: Schaden dem Anwender, der sensible Daten unter vorgetäuschten, falschen URL-Parametern preisgibt.

5. Generelle Verwundbarkeit von Applikationen: Bösartiger Code nutzt Fehler in der Logik einer Software.

Die Kosten der Fehlerbehebung wachsen in allen Fällen exponentiell mit der vergangenen Zeit nach einem Angriff. Unabhängig davon: Sicherheitsmängel führen nicht nur zu beträchtlichen Entwicklungskosten. Viel wichtiger und für ein Unternehmen belastender, sind die damit verbundenen finanziellen und geschäftlichen sowie rechtlichen Probleme und Folgen.

Was Unternehmen berücksichtigen sollten

Die Sicherheit von Softwareanwendungen sollte als einer der kritischen Faktoren bei Architektur, Design und laufender Kontrolle im Entwicklungsprozess betrachtet werden. Um eine sichere Anwendung zu erhalten, müssen die wichtigsten Standards identifiziert und auf allen Stufen der Entwicklungsarbeit beachtet, durchgesetzt und überwacht werden. Für .NET-Anwendungen finden sich Hinweise zu sehr empfehlenswerten Prozessen und Praktiken schon bei Microsoft selber (etwa unter: <http://www.microsoft.com/security/sdl/default.aspx>). Diese Prozesse können angepasst und vereinfacht werden, um auch bei kleineren Projekten praktikabel zu bleiben oder dem Anspruch der Softwareentwicklung nach Methodologien wie Agile, nach Scrum oder XP, zu entsprechen.

Die Gewährleistung der Sicherheit einer Softwareapplikation gleicht aber immer der Verfolgung eines beweglichen Ziels. Die regelmäßige Weiterbildung der Entwickler sowie die Übernahme aktueller und empfehlenswerter Praktiken und Standards sind daher eine kontinuierliche Aufgabe. Eine Aufgabe, zu deren Erfüllung Entwicklungsspezialisten wie SaM Solutions einen beachtlichen Aufwand betreiben. So gibt es hier die firmeninterne Position des sogenannten „Chief-Technologist“. Dieser überwacht die Umsetzung von Technologiestandards, kümmert sich um Weiterbildungsmaßnahmen und generell um das Thema Sicherheit im Source-Code. SaM Solutions unterscheidet an dieser Stelle zwischen Verantwortlichen für .Net und Java. Die prinzipielle Vorgehensweise in den beiden Welten ist aber identisch. ■