IT-Projekte im Saldo

Messen und Koordinieren der technischen Schulden

Jedes Entwicklerteam kennt die Fallstricke bei der Erschaffung neuer Software. Einer davon sind technische Schulden. Wenn Software entwickelt wird, werden technische Schulden aufgenommen. Es beginnt bei der Wahl der Technologie und mit jeder neuen Zeile Quellcode erhöht sich diese Schuld.

Technische Schulden können ganze Projekte lahmlegen, sie können aber auch – als bewusste Entscheidung – einen signifikanten Vorteil am Markt schaffen. Doch wie können diese clever gemanagt werden?

Der Begriff „Technical Debt“ wurde erstmals 1992 von Softwareentwickler Ward Cunninghan verwendet. Er wollte seinem Projektleiter anhand einer verständlichen Metapher klarmachen, wieso es wichtig ist, den Quellcode sauber zu halten:

„Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation, object-oriented or otherwise.“

Mit dieser Metapher wurde eine monetäre bzw. betriebswirtschaftliche Dimension geschaffen, welche für alle Stakeholder eines Softwareprojektes verständlich ist. Mit technischen Schulden beschreibt man demnach mögliche Konsequenzen einer schlechten technischen Umsetzung von der Programmierung. Wenn die Schuld zu hoch wird, steigt die Gefahr, dass die Software nicht mehr wart- oder erweiterbar ist. Die erhöhte Entwicklungszeit zur Fehlerbehebung gleicht hierbei dem Bezahlen der Zinsen.

Hinzu kommt, dass die Anzahl der Fehler steigt. Dies können sichtbare Fehler wie unvorhergesehene Programmabstürze oder Fehlerverhalten bei der Bedienung sein. Aspekte wie Sicherheit und Performance werden von technischen Schulden nicht minder tangiert. Dagegen spricht für die Aufnahme technischer Schulden eine Verkürzung der Time-to-Market.

Martin Fowler, Experte für Software-Architektur, hat mit dem Technical-Debt-Quadranten eine Hilfestellung für den Umgang mit den technischen Schulden geschaffen:

Es wird zwischen 4 verschiedenen Arten von Schulden unterschieden. Im oberen Bereich des Quadranten ist allen Beteiligten bewusst, dass technische Schulden aufgenommen werden und welche Konsequenzen diese Handlung nach sich zieht. Im unteren Bereich fehlt dieses Wissen. Für das Projekt entstehen damit neue ungeplante Aufwände. Ziel des Teams müsste es demnach sein, sich immer im oberen Bereich des Quadranten zu befinden.

Wie kann man die technischen Schulden in einem Projekt messen?

1. Auf die Softwareentwickler hören!
Die Softwareentwickler sind sehr nahe an der Software und sie wissen sehr genau, ob sie etwas sauber und stabil entwickelt haben oder ob hier noch Nacharbeiten notwendig sind.

2. Das Ökosystem auf dem aktuellen Stand halten.
Meistens beinhaltet Software, neben den selbstentwickelten Teilen, auch Bibliotheken oder Infrastrukturelemente, die außerhalb des eigenen Projektteams weiterentwickelt wer-

	Reckless	Prudent
Deliberate	*„We don't have time for design"*	*„We must slip now and deal with consequences"*
Inadvertent	*„What's Layering?"*	*„Now we know how we should have done it"*

Abbildung 1 Technical Debt Quadrant

den. Man sollte sich im Projekt darum bemühen, diese Elemente auf dem aktuellsten Stand zu halten.

3. Auf die Roboter hören!

Mittels statischer Source-Code-Analyse können Mängel in der Software identifiziert werden. Hierfür gibt es zahlreiche Werkzeuge aus der Open-Source-Gemeinde oder von namhaften Herstellern wie IBM oder HP. Letzteres sogar aus der Cloud.

Steuerung von technischen Schulden

Zur Steuerung technischer Schulden könnte in agilen Entwicklungsteams eine mögliche und bewährte Variante Anwendung finden: Entwickler kommunizieren bei den Daily-Stand-ups, ob sie technische Schulden aufgenommen oder identifiziert haben. Der Quellcode wird regelmäßig mittels statischer Source-Code-Analyse geprüft: jeweils in der Nacht oder bei jedem Build. Bevor dies stattfindet, muss ein Regelsatz vereinbart werden. Dieser definiert, was geprüft wird. Diese Informationen sollten den Entwicklern bereitgestellt werden. Je nach verwendetem Tool kann man hierfür eine Weboberfläche verwenden oder eine direkte Integration in die Entwicklungsumgebung. Die Entwicklungsleitung koordiniert die Upgrades der verwendeten Softwarebestandteile, wobei die Identifizierung von veralteten Bestandteilen durch automatisierte Tools unterstützt wird. Alle identifizierten Schulden werden als Arbeitsaufträge erfasst, quantifiziert und qualifiziert. Ungefähr 15% der Entwicklungszeit jedes Sprints wird für die Beseitigung von technischen Schulden verwendet. Für die Priorisierung der Schulden kann die Qualifizierung der Schuld als Indikator dienen. Es kann aber auch sinnvoll sein, die Schulden im Kontext von anderen Arbeitspaketen zu adressieren.

Schulden müssen zurückbezahlt werden – in der Software-Entwicklung gilt das nicht immer

Bei der Planung der Reduzierung der Schulden sollte stets bedacht werden: Nicht alle Schulden müssen bezahlt werden. Wenn ein Bereich der Software sehr viele Schulden hat und in Zukunft überarbeitet werden soll, sollte man keine Arbeitszeit für die Beseitigung dieser technischen Schulden einplanen.
Zudem lassen sich andere organisatorische Maßnahmen treffen, um die Qualität des Quellcodes zu erhöhen, beispielsweise Code Reviews, Pair-Programming, Test-Driven-Development.
In der Praxis wird dieses Thema von den großen IT-Firmen sehr ernst genommen. Allerdings gehen die Unternehmen unterschiedlich damit um. Hier sind einige Beispiele:

Amazon hat für die erste Plattform sehr viele technische Schulden aufgenommen und diese niemals bezahlt. Die gesamte technische Basis wurde mehrfach fundamental erneuert, hierbei wurde sehr viel Quellcode „weggeschmissen". Somit sind auch die Schulden verfallen. Die letzte Lösung von Amazon ist technisch so potent, dass sie sich hiermit sogar ein weiteres Geschäftsfeld erschlossen haben: Amazon Web Services.

Microsoft hat in den ersten Tagen sehr viele technische Schulden aufgenommen, um die Time-to-Market zu reduzieren und sich aufgrund des zeitlichen Faktors gegenüber den Mitbewerbern durchzusetzen. Lange Zeit waren die Betriebssysteme von Microsoft sehr fehlerbehaftet und unsicher. Bill Gates hat das Thema technische Schulden 2002 in seinem Memo „Trustworthy Computing" zur Chefsache gemacht und als Ergebnis u.a. Windows 7 hervorgebracht.

Jedes IT-Projekt baut technische Schulden auf. Der Umgang damit hat entscheidenden Einfluss auf den Erfolg des Projektes. Um bessere Entscheidungen treffen zu können, sollte man also immer technische Schulden messen und koordinieren! ■

QUELLEN
http://martinfowler.com/bliki/TechnicalDebtQuadrant.html
http://blog.crisp.se/2013/10/11/henrikkniberg/good-and-bad-technical-debt
http://www.wired.com/2002/01/bill-gates-trustworthy-computing/
https://www.sogeti.de/dienstleistungen/software-qualitatssicherung-und--test/application-security-testing/

Toni Gansel ist Testmanager bei Sogeti. Er hat mehr als ein Jahrzehnt IT-Projekte dabei unterstützt, qualitativ hochwertige Software zu entwickeln.

► Facebook kauft WhatsApp.

EIN HUNDERT TAUSEND

► iSQI nimmt die 100.000 Prüfung ab.