



Frank Schwarz

(E-Mail: frank.schwarz@buschmais.de),

IT-Berater bei der buschmais GbR.

MANAGING SOFTWARE DEBT

Wer kennt das nicht? Man arbeitet in einem langjährig erfolgreichen Softwareprojekt, aber die Implementierung selbst einfachster Anforderungen wird immer mühsamer. Unentwegt beschleicht einen das Gefühl, für all die „Hacks“ und „Quick Fixes“ der Vergangenheit nun teuer bezahlen zu müssen. In seinem Buch „Managing Software Debt“ greift Chris Sterling dieses Phänomen auf und zeigt Wege aus der Software-Schuldenfalle.

Softwareschulden – so die zentrale These des Buchs – sind Lücken in der Implementierung, in der Qualitätssicherung oder in anderen Bereichen, denen sich der Entwickler auf Dauer nicht entziehen kann. Es sind Abkürzungen, die auf kurze Sicht einen Vorsprung verschaffen, aber später zu immer größeren Problemen führen. Ab einem bestimmten Zeitpunkt betreiben die Entwickler fast nur noch Codepflege – für den Endanwender entsteht kaum noch Zusatznutzen. Neue Features brauchen sehr lange, bis sie sich stabilisieren.

Chris Sterling beschreibt in seinem Buch auf 288 Seiten, wie Softwareschulden entstehen, welche Auswirkungen sie haben und wie sie bekämpft werden können. Das Buch ist gespickt mit Referenzen zum aktuellen Stand der Diskussion – beispielsweise zu Aufsätzen von Martin Fowler oder Ward Cunningham. Der Autor selbst ist ein erfahrener IT-Consultant, zertifizierter Scrum-Trainer und nach eigenem Bekunden passionierter Softwareentwickler. Die im Buch diskutierten Probleme und Lösungsansätze sind gut nachvollziehbar und wirken authentisch.

Für eine Analyse des Phänomens und die Erarbeitung passender Lösungsstrategien trennt Sterling die Schulden-Metapher auf in:

- Schulden in der technischen Umsetzung
- Schulden in der Qualitätssicherung
- Schulden im Konfigurationsmanagement
- Schulden im Design
- Schulden in der Organisation der Wissensweitergabe

Jedem dieser Aspekte wird in dem Buch ein eigenes Kapitel gewidmet.

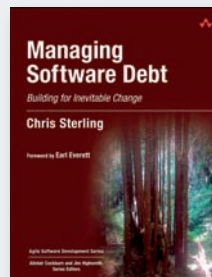
Zum Kampf gegen Softwareschulden beschreibt der Autor Techniken, die zum Teil mit tradierten Denkmustern der Softwareentwicklung brechen. Eine Technik

lautet *Architecture is S.A.I.D.* Dabei steht das „A“ für „Alignment“ und fordert dazu auf, die Architektur eines Softwaresystems mit den fachlichen Anforderungen und der

Chris Sterling

Managing Software Debt – Building for Inevitable Change

Addison-Wesley
International
Dezember 2010
288 Seiten, 35,05 €
ISBN: 978-0-3215-5413-0



Geschäftsstrategie des Unternehmens zu begründen. Die gängige Praxis hingegen betrachtet Architekturen meist als technische Herausforderung und degradiert die Fachlichkeit zur Randnotiz. So entstehen nicht selten Ungetüme, bei denen das Attribut „over-engineered“ einer Verharmlosung gleichkommt.

Ein interessantes Merkmal des Buches ist die unerschöpfliche Zahl von Aphorismen. Wer wollte widersprechen bei Sätzen wie: „Acceptance tests are not sufficient as the entire test suite for ensuring quality“ (Seite 95), „Documentation can help somewhat but is prone to misinterpretation and lacks the appropriate detail“ (Seite 167) oder „[...] then people are moved around as ‘resources’ to keep them busy and persuade stakeholders that everything is being done to get their project finished“ (Seite 200)? Als Leser möchte man hier ausrufen: „Ja, genauso ist es!“

Sterlings Lösungsstrategien reichen von Selbstverständlichkeiten, wie der Einführung einer Testautomatisierung oder der

Benutzung eines Continuous-Integration-Servers, bis hin zu eher schwierig umsetzbaren Techniken, wie „Test-Driven-Design“ oder gar „Pair-Programming“. Auch das Einreißen von Cubicle-Wänden wird thematisiert. Die Mehrheit der Vorschläge hat einen starken Einschlag in Richtung agile Softwareentwicklung und im Anhang des Buchs befinden sich zwei Abschnitte zu Scrum und XP.

Fazit

Das Buch ist eine wahre Fundgrube für all diejenigen, die nach Argumenten suchen, um ihr Management von der Einführung agiler Entwicklungsmethoden zu überzeugen. Viel mehr leistet das Buch jedoch leider nicht. Sterling schafft es weder, den Schulden-Begriff schlüssig zu definieren, liefert er nützliche Empfehlungen an das Management zur Steuerung von Softwareprojekten. Das Dilemma, das der Autor vermutlich zu lösen versuchte, zeigt sich in der originalen Begriffsprägung durch Ward Cunningham: Für ihn entstehen Softwareschulden dann, wenn der Entwickler es unterlässt, den Code seinem geänderten Verständnis der Fachdomäne anzupassen. Die Schulden-Metapher wählte er, um dem Management zu vermitteln, dass Software-Code eine innere Qualität besitzt, die unbedingt aufrechterhalten werden sollte. Wenn das Management bereits diese Auffassung teilt, bedarf es kaum der Einführung neuer Methoden.

Der andere Fall ist weitaus interessanter. Sollte das Management durch unsinnige Prozesse oder unrealistische (Zeit-)Vorgaben selbst die innere Qualität der Software untergraben, kann ein Methodenwechsel kaum Abhilfe schaffen. Scrum ist kein Allheilmittel. Eine Verbesserung entsteht hier nur durch einen organisatorischen Wandel über das Entwicklungsteam hinaus. ■