Ein formal fundierter Entscheidungs-Ansatz zur Behandlung von Technical Debt

Klaus Schmid

Institut für Informatik
Universität Hildesheim
Marienburger Platz 22
31141 Hildesheim
schmid@sse.uni-hildesheim.de

Zusammenfassung: Der Begriff Technical Debt bezeichnet Entwicklungsaltlasten, die in der Weiterentwicklung von Software zu Problemen führen können. Diese Problematik ist von großer praktischer Bedeutung, vor allem in inkrementellen Entwicklungsprozessen, und gewinnt aktuell auch in der wissenschaftlichen Community stark an Bedeutung. Hier betrachten wir die Frage welche Bedingungen eine optimale Strategie zur Behandlung von Technical Debt erfüllen muss, ob eine solche Strategie überhaupt möglich ist und wie realistische Strategien aussehen können. Dabei ergibt sich, dass eine optimale Strategie auf fundamentale Probleme stößt. Wir führen daher Approximationen ein, die eine optimale Auswahl ermöglichen. Daraus leiten wir einen Ansatz zur Auswahl von Technical Debt items ab. Dieser erlaubt die effiziente Behandlung von Technical Debt in beliebigen, insbesondere iterativen und gar agilen Entwicklungsprozessen.

1 Technical Debt

Mit dem Begriff *Technical Debt* werden Kostentreiber in der Softwareentwicklung bezeichnet, die auf frühere Entwicklungsentscheidungen zurückgehen. Oft haben diese Entscheidungen zuerst zu einer Vereinfachung und Beschleunigung der Entwicklung geführt, jedoch führen sie in der weiteren Evolution zu Zusatzaufwänden. Cunningham führte für solche kurzfristig beschleunigten, aber langfristig behindernden Entscheidungen den Begriff *Technical Debt* (technische Schulden) ein [Cun92]. Beispiele dafür können sein: die Nutzung eines Frameworks, das den Anforderungen im Verlauf der Weiterentwicklung nicht gewachsen ist, oder die Verwendung eines vereinfachten Designs, welches im Rahmen der Weiterentwicklung nicht mehr hinreichend skaliert. *Technical Debt* lässt sich zurückzahlen, indem die entsprechenden Entscheidungen revidiert werden, bspw. durch Ersetzen des Frameworks oder durch Refactoring des Designs.

Das Erkennen und Behandeln von *Technical Debt* ist ein Thema von hoher praktischer Relevanz: sowohl in allen inkrementellen Entwicklungsprozessen (wie bpsw. agilen Ansätzen) als auch allgemein in der Evolution von Software. Seit kurzer Zeit erhält das Thema auch intensive Aufmerksamkeit in der Forschungsgemeinde (Veröffentlichungen, Sonderhefte, Workshops).

2 Ein Ansatz zur Entscheidungsfindung

Es gibt bereits eine Vielzahl von Arbeiten, die sich mit dem Thema *Technical Debt* beschäftigen. Oft werden dabei ad-hoc Kriterien und Ansätze verwendet [Sch13a]. In dem in [Sch13b, Sch13c] dargestellten Ansatz wurde daher der Fokus auf eine systematische Ableitung der Kriterien und des Entscheidungsverfahrens gelegt. Der Ansatz führt eine detaillierte und formale Definition von Technical Debt ein, die sich an den Kostenunterschieden zwischen Einsparungen durch das Eingehen von Technical Debt und den verursachten Kosten im weiteren Entwicklungsverlauf orientiert. Dabei werden die Kosten als Erwartungswert zukünftiger, möglicher Entwicklungspfade definiert. Auf dieser Basis ist eine präzise Definition möglich, was auch die Ableitung eines optimalen Entwicklungspfads erlauben würde. Eine praxisorientierte Umsetzung der Theorie stößt jedoch auf fundamentale Schwierigkeiten, da bspw. alle möglichen alternativen Architekturen bzgl. aller möglichen zukünftigen Entwicklungspfade bewertet werden müssten. Dies ist für realistische Systeme prinzipiell unmöglich. Man kann entsprechend davon ausgehen, dass die Bestimmung eines optimalen, praktisch anwendbaren Ansatzes zur Behandlung von Technical Debt fundamental unmöglich ist.

Ausgehend von der Unvereinbarkeit eines präzisen Ansatzes zur Technical-Debt-Behandlung und den Bedürfnissen eines praxisnahen Ansatzes werden verschiedene Approximationen eingeführt: der Vergleich mit einem optimalen Referenzsystem wird durch den Vergleich mit einem fixen Referenzsystem ersetzt, die Annahme der Unabhängigkeit der einzelnen Entwicklungsschritte wird eingeführt und die Betrachtung einer beliebig großen Menge von Entwicklungssequenzen wird durch die Betrachtung einer endlichen (beliebig kleinen) Repräsentantenmenge ersetzt. Prinzipiell lassen sich diese Approximationen beliebig kombinieren und nur einzelne davon verwenden. Wir betrachten jedoch beispielhaft den Ansatz, der entsteht wenn alle Approximationen gleichzeitig angewendet werden. Der resultierende Ansatz erweist sich als einfach anwendbar und in grundlegender Weise mit dem ursprünglichen Ansatz verwandt. Insbesondere besitzt der Ansatz eine systematische Grundlage mit definierten, einzelnen Approximationen, die explizit identifiziert und formalisiert sind. Damit hat der Ansatz fundamentale Vorteile gegenüber anderen publizierten Ansätzen. Weiterhin kann er einfach in inkrementelle, insbesondere agile Ansätze integriert werden.

Literaturverzeichnis

[Cun92] W. Cunningham. *The WyCash portfolio management systems.* In Object-oriented programming systems, languages, and applications (OOPSLA'92, Addend.), Seite 29–30, 1992. www.c2.com/doc/oopsla92.html.

[Sch13a] K. Schmid. *On the Limits of the Technical Debt Metaphor: Some Guidance on Going Beyond*, Proceedings of the 4th International Workshop on Technical Debt (MTD) at ICSE, Seite 63-66, 2013. DOI: 10.1109/MTD.2013.6608681

[Sch13b] K. Schmid. *A Formal Approach to Technical Debt Decision Making*, Proceedings of the 9th international ACM Sigsoft conference on Quality Of Software Architectures (QoSA'13), Seite 153-162, ACM, 2013. DOI: 10.1145/2465478.2465492.

[Sch13c] K. Schmid. *Technical Debt — From Metaphor to Engineering Guidance: A Novel Approach based on Cost Estimation.* Technical Report, University of Hildesheim, No. 1/13, 2013.