Technical Debt-Propagation in koadaptiven Systemen

Johann Schütz
OFFIS – Institut für Informatik, Oldenburg, Deutschland
johann.schuetz@offis.de

Zusammenfassung Traditionelles Technical Debt-Management betrachtet Technical Debt vorrangig als eine geschlossene Systemeigenschaft und behandelt dieses zumeist als eine isolierte kurz- oder langfristige Trade-Off-Entscheidung. Moderne Softwaresysteme sind jedoch i. d. R. komplexe, offene und hochgradig vernetzte soziotechnische Systeme mit gegenseitigen Abhängigkeiten. Diese miteinander verwobenen Systeme gehen trotz ihrer Unabhängigkeit eine koadaptive und damit verbundene koevolutionäre Beziehung ein. Dieser Kurzbeitrag beschreibt, wie der Effekt des Technical Debt-Propagation dabei systemübergreifende Synergien in Dysergien transformiert.

1 Einleitung

Technical Debt (TD), z. dt. technische Schulden, beschreibt ein alltägliches, unvermeidliches und zunehmend disruptives Problem in der Software-Entwicklung. Seit Cunningham [1] 1992 erstmals die mögliche Auswirkung von TD beschrieb, wurden Softwaresysteme analog zu Lehmans Gesetze der Software-Evolution [2] zunehmend komplexer. Ursprünglich geschlossene technische Systeme entwickelten sich in einem stetig anhaltenden Evolutionsprozess zu offenen und hochgradig vernetzten soziotechnischen Systemen. Mit der zunehmenden Größenordnung der Systemgrenzen verschieben sich jedoch die beim Software- und Systems- Engineering zu berücksichtigenden Charakteristiken des Systems. Bisherige Qualitäts-modelle wie bspw. SQuaRE der ISO/IEC 25010 Standard für „Systems and Software Engineering -- Systems and Software Quality Requirements and Evaluation (SQuaRE) -- System and Software Quality Models“ betrachten die Systemqualität vorrangig als eine geschlossene Systemeigenschaft. Sobald aber zwei oder mehrere Systeme miteinander interagieren, insb. wenn diese untereinander Abhängigkeiten aufweisen, werden die Qualitätseigenschaften des einen Systems jedoch durch die Qualitätseigenschaften des anderen Systems beeinflusst [3].

2 Koadaptive und koevolutionäre Systeme

Schließen sich unabhängige Systeme zu kollaborativen System-of-Systems (SoS) zusammen, lassen sich zwar durch Realisierung von wechselseitigen Synergie-effekten neue Potenziale zwischen allen Subsystemen erschließen [3]. Jedoch gehen die somit miteinander verwobenen Systeme trotz ihrer Unabhängigkeit eine koadaptive und damit verbundene koevolutionäre Beziehung ein. So können Entscheidungen in einem System unmittelbar zum Aufbau von TD in anderen Systemen führen und auf diesem Wege die Gesamtqualität des SoS kompromittieren [4, 3].

3 Technical debt in koadaptiven Systemen

TD in koadaptiven Systemen ist weniger in der Art als an der Ursache-Wirkungs-Kette durch die sich gegenseitig verstärkende systemübergreifende Akkumulation und den sich daraus ergebenden Folgen von „klassischem“ TD zu unterscheiden. So lässt sich TD stets in eine der in **Abbildung 1** dargestellten neun Kategorien einordnen und ist i. d. R. auf bewusste oder unbewusste Entscheidungen einer am Software Engineering-Prozess beteiligten Rolle zurückzuführen. Während einem solchen systeminternen TD durch mehr oder minder kostspielige Re-Engineering Maßnahmen zur Wiederherstellung der internen Qualität beizukommen ist, entwickelt sich zusätzliches TD aufgrund gegenseitiger Adaptionen koevolutionär in den daran geknüpften Systemen [5, 6, 4].

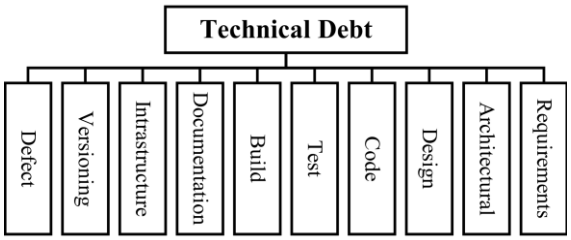

***Abbildung 1**: Technical Debt-Klassifikation nach [7]*

So stellt insb. TD, welches an den Außenschnittstellen eines Systems liegt bzw. diese beeinflusst, durch eine sog. TD-Propagation den größten Risikofaktor dar. Wird z. B. in einer Organisation ein Softwaresystem eingeführt, welches zur Kommunikation ein proprietäres Datenformat (Design Debt) verwendet, ergeben sich grundsätzlich zwei Möglichkeiten:

1. Die Schnittstellen der kommunizierenden Systeme werden (mit zusätzlichem Aufwand) angepasst. Dies führt jedoch zu einer stärkeren Kopplung der Systeme.

2. Es wird ein Adapter eingeführt. Dieser führt jedoch zu einer zusätzlichen Steigerung der Komplexität, der Heterogenität und einer ggf. anhaltenden Erweiterung an neue/weitere Schnittstellen.

Bei einer i. d. R. stetig anhaltenden Evolution führen beide Varianten durch gegenseitige Koadaption (sofern nicht aktiv angegangen) zu einem Wildwuchs der Enterprise Architektur (EA) bzw. des SoS [4, 6]. Ist der Point-of-no-Return erst einmal erreicht, ist TD nicht mehr durch Re-Engineering eines einzelnen Systems, sondern nur noch über ein EA-umfassendes beizukommen.

Unter Berücksichtigung von Maiers‘ [8] Klassifikation für SoS-Architekturen ergibt sich darüber hinaus die größte Herausforderung im Falle einer management- und operativen-Unabhängigkeit der einzelnen Systeme (z. B. bei Smart Grids oder Smart Cities). Die konstituierenden Systeme unterliegen angesichts der Abwesenheit eines zentralen Managements individuellen Anforderungen, Rahmenbedingungen und Stakeholdern [8, 3]. So entwickeln sich die einzelnen Systeme zwar koevolutionär, verfolgen aber unabhängige, u. U. konträre Ziele.

4 Konklusion

Das Phänomen der TD-Propagation ist kein grundsätzlich neues. Doch obwohl TD aufgrund der potenziell disruptiven Eigenschaften sowohl in Forschung als auch Wirtschaft zunehmend Anklang findet, bleiben die netzwerkartigen Effekte weitestgehend unberücksichtigt. So werden zwar insb. in der Praxis selbstverständlich mögliche Auswirkungen auf Schnittstellenänderungen berücksichtigt, jedoch Entscheidungen i. d. R. aufgrund von Short-term-Benefits nur in Ausnahmefällen überorganisatorisch ausgerichtet. Diese Entscheidungen sind oftmals auf die Grenzen der TD-Metapher [9] zurückzuführen. Denn, ob TD zu einer Schuld wird, hängt von der Zukunft ab und kann nur im Nachhinein beurteilt werden [9]. Infolgedessen werden Methoden des Risikomanagements als Bewertungsgrundlage herangezogen [5]. Indem TD-Propagation auf diese Weise jedoch gemäß den Gesetzen der Software-Evolution [2] zu einer zusätzlichen stetigen Abnahme der SoS-Gesamtqualität führt, erstrecken sich die möglichen Schäden und damit verbundenen Risiken von einer einfachen systemübergreifenden Fehlerweitergabe bis hin zu einem Kaskadenausfall des SoS mit unvorhersehbaren Netzwerkeffekten und damit verbundenen Kosten über alle Beteiligten hinweg [3].

Während die Beseitigung von TD bereits auf der EA-Ebene eine Herausforderung für sich darstellt, ist der gemeinsame und organisationsübergreifende Abbau von TD in vielen Fällen nicht mehr möglich. So sollte die Software- und/oder System-Entwicklung mit externen Abhängigkeiten, insb. im Falle eines SoS im KRITIS-Umfeld, stets ein globales Optimum ohne temporäre Trade-Offs anstreben. Alternativ erreicht ein System (dargestellt als roter Punkt in **Abbildung 2**) aufgrund von Short-term-Benefits zwar ein lokales Optimum, entrückt jedoch weiter vom globalen Optimum.

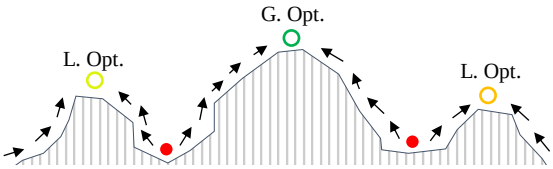

***Abbildung 2**: Fitness-Landschaft*

Indem jede Iteration des Evolutionszyklus zumeist mit zusätzlichen Aufwänden für alle beteiligten Systeme und dazugehörigen Akteure verbunden ist, ist das globale Optimum aus einem lokalen Optimum heraus i. d. R. nicht mehr zu erreichen.

Literaturverzeichnis

[1] W. Cunningham, „The WyCash Portfolio Management System,“ *Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum)}, OOPSLA '92*, pp. 29--30, 1992.

[2] M. M. Lehmann, „Laws of Software Evolution Revisited,“ *Proceedings of the 5th European Workshop on Software Process Technology (EWSPT '96)*, pp. 108-124, 1996.

[3] Office of the Deputy Under Secretary of Defense for Acquisition and Technology, „Systems and Software Engineering. Systems Engineering Guide for System of Systems, V 1.0,“ 2008.

[4] T. Suovuo, J. Holvitie, J. Smed, V. Leppänen, „Mining Knowledge on Technical Debt Propagation,“ *14th Symposium on Programming Languages and Software Tools*, pp. 281-295, 2014.

[5] N. Ramasubbu, C. F. Kemerer, „Technical Debt and the Reliability of Enterprise Software Systems: A Competing Risks Analysis,“ *Management Science Vol. 62, No. 5*, pp. 1487-1510, 5 2016.

[6] J. D. McGregor, J. Y. Monteith, J. Zhang , „Technical debt aggregation in ecosystems,“ *2012 Third International Workshop on Managing Technical Debt (MTD)*, pp. 27-30, 2012.

[7] Z. Li, P. Avgeriou, P. Liang, „A Systematic Mapping Study on Technical Debt and its Management,“ *Journal of Systems and Software Volume 101*, pp. 193-220, 2015.

[8] M. W. Maier, „Architecting Principles for System-of-Systems,“ *Systems Engineering 1*, 1998.

[9] K. Schmid, „On the limits of the technical debt metaphor some guidance on going beyond,“ *2013 4th International Workshop on Managing Technical Debt (MTD)*, pp. 63-66, 5 2013.