Möglichkeiten von Web API Spezifikationen im Software Reengineering

Andreas Schmietendorf

Hochschule für Wirtschaft und Recht Berlin
Email: andreas.schmietendorf@hwr-berlin.de

1. Motivation und Ziele

Moderne Integrationsarchitekturen, aber auch Microservice-basierte Anwendungen nutzen vielfältige Aspekte aus dem Umfeld webbasierter APIs (Web Services, RESTful Services, Open APIs, …). Es stellt sich die Frage, inwieweit auch das Software Reengineering davon profitieren kann. Ein offener und standardisierter Ansatz bietet sich mit den fachlich orientierten Open API Spezifikationen, welche häufig mit Swagger (JSON) realisiert werden. Aus Sicht des Autors können auch Altsysteme mit Hilfe dieser abstrakten Schnittstellenbeschreibungen entlang ihrer fachlichen Aufgabenstellungen ergänzt, geöffnet, restrukturiert und schließlich modularisiert werden. In diesem Zusammenhang wird sowohl von Aufgaben des Software Reengineerings als auch der Softwaremigration gesprochen. Im Rahmen des Beitrags soll auf Ziele, Möglichkeiten und Rahmenbedingungen derartiger Projekte eingegangen werden. In diesem Zusammenhang soll auch auf ein Framework der Telco-Branche zur domänenspezifischen Standardisierung der Spezifikation von offenen APIs eingegangen werden. Abschließend werden die Ergebnisse einer Umfrage zur Verwendung von REST-basierten Web APIs im Software Reengineering aufgezeigt.

2. Reengineering mit Open APIs

Die Idee zur Kapselung von Quellcode über Wrapper-Techniken (z.B. CORBA, WSDL) ist nicht neu. [Niiyama 2008] spricht z.B. vom Service-orientierten Software Reengineering im Zusammenhang mit der Modernisierung von Altanwendungen auf der Grundlage kompositorisch verwendeter Web Services. Ein Vorschlag zur „Objektifizierung“ eines Unternehmens findet sich bei ([Sneed 2010] S. 76). Von den so gekapselten und unabhängig testbaren bzw. wartbaren Komponenten soll sowohl das Business Reengineering als auch das Software Reengineering profitieren.

Mit der zunehmenden Nutzung des REST-basierten Architekturstils zur Entwicklung webbasierter Integrationslösungen wurde eine neue Debatte zur Gestaltung im Web verteilter und vor allem heterogener Softwarelösungen entfacht. Im Mittelpunkt dieses Gestaltungsprinzips stehen die lose Kopplung beteiligter Systeme und Services, die Gewährleistung der Interoperabilität durch Uniformität der Schnittstellen, ein weltweit einheitlichen Namensraum (URIs), die Wiederverwendung von fachlich begründeten Web APIs sowie die Performance und Skalierbarkeit durch eigenständig installierbare (deloyable) Komponenten.

Unter [Wolf 2016] wird das Aufbrechen bzw. Ergänzen von Altanwendungen mit Hilfe von Microservices thematisiert. Unter der Voraussetzung fachlich klar voneinander abgegrenzter Funktionseinheiten lässt sich eine Überführung in Microservices gut realisieren. Die nach außen genutzten Schnittstellen sollten sich an einem fachlichen Spezifikationsstandard für Prozesse, Daten und Fachservices (vgl. Abschnitt 3) orientieren, so dass ein einfaches Ersetzen der Komponenten durch interne oder auch externe Web APIs unterstützt wird. In diesem Zusammenhang spricht [Lilienthal 2017] auch von „bounded context“, d.h. Microservices sollten das Ergebnis einer fachlich spezifizierten Domäne darstellen. Mit Hilfe einer derartigen Architektur lässt sich agiler auf veränderte Anforderungen reagieren.

Bei über Jahrzehnte gewachsenen Softwarelösungen findet sich allerdings zumeist eine eher technisch orientierte Aufteilung der involvierten Komponenten. Eine komplette Restrukturierung entlang der fachlichen Domänen verbietet sich aufgrund der einhergehenden Aufwände. Zweckmäßig ist es hier, Altanwendungen bezüglich benötigter Prozesse und Daten bedarfsorientiert über Web APIs zu öffnen oder auch eine Ergänzung mit Hilfe parallel zur Altanwendung betriebenen Microservices vorzunehmen. Diese sollten dabei möglichst unabhängig gewartet und installiert werden können. Der letztgenannte Ansatz wurde durch den Autor z.B. im Rahmen telekommunikationsspezifischer Softwarelösungen genutzt, um so schrittweise die Funktionsblöcke innerhalb der Altanwendung abzuschalten. Die fachliche Spezifikation dieser Funktionsblöcke sollte sich an Standards orientieren, wie z.B. denen der Open APIs für die Telekommunikationsbranche.

3. Telco spezifische Open APIs

Die Telekommunikationsbranche musste sich frühzeitig mit der benötigten Interoperabilität von Altsystemen auseinandersetzen. Durch das industriegetriebene TMF Forum werden seit ca. 3 Jahren auch Open API Spezifikationen für fachlich orientierten Domänen dieser Branche zur Verfügung gestellt [TMF 2018].

Domänen der Open API Spezifikationen (Stand: Frameworkx Release 17.5)	Current	Planned	Future
MARKET/SALES	-	-	3
PRODUCT	3	1	2
CUSTOMER	8	4	5
SERVICE	8	1	-
RESOURCE	4	3	1
ENGAGED PARTY	8	-	1
ENTERPRISE	-	-	3
COMMON BUSINESS ENTITIES	7	4	4

Abb. 1: Open APIs des TM Forums [TMF 2018]

Neben der domänenspezifischen Klassifikation (farblich markierte Zeilen – z.B. Markt, Produkt, Kunde, Service) werden die Open APIs nach existenten, geplanten und zukünftig vorgesehene Spezifikationen unterschieden. In der Abbildung kann die Anzahl der Open API Spezifikationen nachvollzogen werden. In der Originalquelle finden sich darüber hinaus Bezüge zu den durch das TM Forum modellhaft beschriebenen Ressourcen, welche durch Prozess- (vgl. eTOM) und Datenmodelle (vgl. SID) untersetzt werden. [Garcia et al. 2009]

Bei den konkreten Open APIs handelt es sich um mit Hilfe von Swagger dokumentierte Spezifikationen, d.h. hier wird aus fachlicher Sicht beschriebenen, was entsprechende Schnittstellen (z.B. Microservices) an Funktionen/Daten anbieten, nicht aber, wie diese konkret implementiert werden. Werden derartige Spezifikationsstandards innerhalb des Reengineerings existierender Anwendungen berücksichtigt, lassen sich die eingangs erwähnten Aufgaben vereinfachen.

- *Ergänzung* – Einfaches Einbinden von externen Web APIs zur Nutzung von Daten und Funktionen.
- *Öffnung* – Semantisch abgestimmte Prozesse und Daten lassen sich aus der Altanwendung anbieten.
- *Restrukturierung* – Identifikation fachlicher Services und Vorbereitung potentieller Migrationen.
- *Modularisierung* – Überführung der Altanwendung in eine flexible Microservice-Architektur.

So kann eine schnelle und kostengünstige Integration in ein globales Connectivitäts-Ökosystem gewährleistet werden, aber auch die Leistungsfähigkeit durch die Verwendbarkeit skalierbarer Betriebsinfrastrukturen sichergestellt werden. Ein aktuelles Beispiel für ein Reengineering-Projekt unter Verwendung der Open API Spezifikationen des TM Forums findet sich z.B. unter [Végső 2018]. Hier erfolgte die Harmonisierung von 5 national betriebenen „Customer Self Service“-Lösungen unter Verwendung eines Abstraktions-Layers für den Zugriff auf die Daten/Funktionen der Altsysteme.

4. Empirische Umfrage

Im Rahmen des beim Autor durchgeführten Forschungsprojekts erfolgte neben der fachlichen und technologischen Auseinandersetzung auch eine empirische Umfrage zum Einsatz von Web APIs im Software Reengineering. An dieser beteiligten sich 14 Projektpartner aus dem industriellen Diskurs. Im Folgenden finden sich einige ausgewählte Ergebnisse. Das Alter der betreuten Anwendungen verteilte sich wie folgt:

- weniger als 5 Jahre (50%)
- 5 bis 10 Jahre (21,4%)
- 10 bis 20 Jahre (28,6%).

Ein Großteil wurde mit Java (50%) bzw. C/C++ (35%) implementiert. Fast 80% gaben an, dass die Altanwendungen schlecht bzw. nicht anpassbar seien. Mängel wurden ebenso hinsichtlich schlechter Dokumentation und „ausgestorbenen Technologien“ benannt.

57 % beschrieben den Kern ihrer Aufgabenstellungen mit Wartung und Pflege des Altsystems. Die Idee der extern nutzbaren Web APIs war 64 % geläufig, weshalb die folgende Frage zu den Gründen eines Web API Einsatzes lediglich von 8 befragten Projekten beantwortet wurde, wobei Mehrfachantworten zugelassen waren:

- Erweiterungszwang/neue Funktionen (75%)
- Nutzung externer Web APIs (75%)
- Verbesserung Modularisierung/Wartbarkeit (37%)
- Angebot von eigenen Web APIs (25%)

Die schrittweise Ersetzung (sanfte Migration) von aus monolithischen Anwendungen „herausgelösten“ Komponenten durch korrespondierende Web APIs wurde von 57% aller Befragten bestätigt.

Als Gründe, die gegen den Einsatz von Web APIs sprechen, wurden die folgenden benannt. Die Reihenfolge (beginnend mit 1) korrespondiert mit der Häufigkeit:

1. Sicherheitsbedenken
2. Hoher Arbeitsaufwand
3. Ohne erkennbaren Mehrwert
4. Bedarf zusätzlicher Dokumentationen
5. Pflicht zur nachhaltigen Wartung.

Probleme bei Nutzung von Web APIs wurden hinsichtlich der Dokumentation und Verfügbarkeit benannt.

6. Fazit

Es steht außer Frage, dass Web API Spezifikationen einen Beitrag zum Software Reengineering leisten können. Dennoch bleiben viele Aspekte, wie z.B. die Transaktionssteuerung, die Verwaltung von Zuständen oder auch Fragen des Tests und schließlich des Managements umfangreicher Web API Kompositionen, eine Herausforderung für die Entwicklung und den Betrieb.

7. Quellenverzeichnis

[Garcia et al. 2009] Garcia, S.; Gramatikoff, I.; Wilmes, J.: Business Transformation with TM Forum Solution Frameworks and SOA, TM Forum, März 2009

[Lilienthal 2017] Lilienthal, C.: Langlebige Softwarearchitekturen, dpunkt.verlag, Heidelberg 2017

[Niiyama et al. 2008] Niiyama, C.; Chung, S.; Chinn, D.; Davolos, S.: Service-Oriented Software Reengineering, TCSS 702 Design Project in Computing and Software Systems (FINAL, Winter 2008)

[Sneed et al. 2010] Sneed, H.; Wolf, E.; Heilmann, H.: Softwaremigration …, dpunkt.verlag, Heidelberg 2010

[TMF 2018] Open API Map, TM Forum, https://www.tmforum.org/resources/posters/open-api-poster-2018, abgerufen März 2019

[Wolf 2016] Wolff, E.: Microservices, dpunkt.verlag, Heidelberg 2016

[Végső 2018] Végső, C.: A Case Study …, Nov. 2018, (abgerufen 14. 02. 2019), https://medium.com/mito/a-case-study-on-how-we-built-deutsche-telekoms-new-self-care-application-4d5e69d80975