Standardisierung von Web APIs durch die Kombination der Prinzipien von REST und Linked Data[1]

Markus Lanthaler[2]

Abstract: Um die exponentiell wachsende Menge an Informationen weiterhin bewältigen zu können, tauschen immer mehr Systeme riesige Mengen an Daten aus und analysieren sowie bearbeiten diese völlig autonom. Zur Kommunikation werden dabei vorwiegend World Wide Web Technologien verwendet. Da jedoch grundlegende, standardisierte Technologien noch nicht existieren, sind Entwickler gezwungen eine Reihe komplexer Designentscheidungen zu treffen. Das Resultat sind in der Regel proprietäre Services die nur von spezialisierten Clients angesprochen werden können. Diese Dissertation erörtert die Probleme aktueller Web Services sowie entsprechender Semantic Web Technologien und beleuchtet den aktuellen Stand der Technik. Des Weiteren stellt sie die Ergebnisse meiner Forschung an Web APIs der dritten Generation vor. Die erarbeiteten Technologien wurden teilweise bereits als Standards vom World Wide Web Consortium ratifiziert und werden in von Millionen von Webseiten verwendet. Die Dissertation beschreibt nicht nur wie die erarbeiteten Lösungen zum Design und zur Implementierung von Services genutzt werden können, sondern evaluiert auch ihre Praxistauglichkeit die mittlerweile auch durch die weite Akzeptanz bewiesen wurde.

Das World Wide Web hat viele Aspekte unserer Gesellschaft grundlegend verändert. Es hat unsere Leben so schnell und nachhaltig geprägt, dass sich viele ein Leben ohne Web kaum mehr vorstellen können. In der Geschichte der Menschheit waren Informationen immer eine wertvolle und knappe Ressource die nur wenigen zugänglich war. Der Erfolg des Webs hat jedoch in kürzester Zeit zu einem Überfluss an Informationen geführt, die jedem jederzeit zur Verfügung stehen. Sowohl das Konsumieren als auch Produzieren von Inhalten wurde extrem vereinfacht. Niemals zuvor haben Menschen mehr Inhalte erstellt als heute, und auch Firmen und Regierungen stellen eine nie dagewesene Menge an Daten öffentlich zur Verfügung. McKinsey [Ma13] geht in einer Studie davon aus, dass die mehr als eine Million Datensätze, die von Regierungen weltweit bereits veröffentlicht wurden, zu einer jährlichen wirtschaftlichen Wertschöpfung von mehr als drei Trillionen Dollar führen. Man muss sich angesichts dieser Zahlen jedoch stets vor Augen halten, dass ein zunehmend geringerer Teil der Daten direkt von Menschen erstellt wird. Die große Mehrheit wird von Maschinen oder Sensoren erzeugt. Jeden Tag fallen 2,5 Exabyte an—so viel, dass 90% aller heute verfügbaren Daten allein in den letzten zwei Jahren erstellt wurden [IB].

Es ist offensichtlich, dass bei derartigen Datenmengen das bloße Auffinden nicht mehr genügt, da zu viele Daten existieren um jemals von Menschen bearbeitet werden zu können. Wir Menschen sind zum Nadelöhr des Informationsprozesses geworden und brau-

[1] Englischer Titel der Dissertation:
"Third Generation Web APIs—Bridging the Gap between REST and Linked Data"

[2] Institut für Informationssysteme und Computer Medien, Technische Universität Graz
mail@markus-lanthaler.com

chen Maschinen die für uns Daten aus verschiedensten Quellen integrieren, analysieren und auf leicht verständliche Auswertungen reduzieren auf deren Basis wir dann Entscheidungen treffen können. Um das zu realisieren werden immer mehr Systeme miteinander vernetzt. Obwohl mehrere Strategien und Architekturen für eine derartige Vernetzung existieren, haben sich Services als die flexibelste Option erwiesen da sie es einzelnen Systemen erlauben eigenständig und unabhängig zu bleiben. Anstatt Systeme direkt miteinander zu verbinden, kommunizieren serviceorientierte Architekturen über standardisierte Schnittstellen und Protokollen.

Die erste Realisierung solcher Services wurden nach dem so genannten Remote Procedure Call (RPC) Stil entworfen, aber es wurde relativ schnell klar dass RPC- oder konkreter SOAP-basierte Services nicht skalieren und schwierig zu warten sind [Wa94]. Aufgrund dieser Erkenntnisse wurden immer mehr Systeme an das Design des World Wide Webs, dem Representational State Transfer (REST) Architekturstil [Fi00], angenähert. Aufgrund ihrer hervorragender Skalier- und Wartbarkeit sowie ihrer Einfachheit, haben sich RESTful Services oder kurz Web APIs schnell durchgesetzt. Mittlerweile geht man, je nach Schätzung, von 10.000 bis weit über 100.000 solcher Services aus. Dies ist aufgrund der Tatsache, dass solche Services in vielerlei Hinsicht proprietäre Designs und somit nicht zu einander kompatibel sind, erstaunlich. Das Web hätte sicher nie sein exponentielles Wachstum erreicht, wenn Browser für jede einzelne Webseite hätten angepasst werden müssen. In gleicher Weise skaliert ein Ansatz, der für jede API spezialisierte Clients erfordert, nicht.

Der Hauptgrund, der zu dieser Situation führte, ist die Verwendung von proprietären Datenformaten und -modellen sowie die Verwendung von statischer, manuell erstellter und daher nicht maschinenlesbarer Dokumentation, anstatt der dynamischen Aushandlung von Formaten und Protokollen zur Laufzeit. Aus Sicht des REST Architekturstils entspricht das hauptsächlich einer Verletzung des Prinzips von selbstbeschreibenden Nachrichten und einer Vernachlässigung von Hypermedia zur Kommunikation valider Zustandsänderungen zur Laufzeit. Im Endeffekt bedeutet dies, dass wichtige Informationen nicht in maschinenlesbarer Form vorliegen und somit die Implementierung von mächtigen, generischen Clients unmöglich gemacht wird.

Technologien des semantischen Webs könnten zumindest in Hinblick auf selbstbeschreibende Nachrichten helfen, weisen aber selbst mehrere Probleme auf—unter anderem die Vernachlässigung von Hypermedia. In der Praxis jedoch stellen sich ihre (wahrgenommene) Komplexität sowie ihre „Esoterik“ als die größten Hindernisse heraus. Die Entwicklung des semantischen Webs ist für lange Zeit in die Domäne künstlicher Intelligenz abgedriftet, anstatt sich auf praxisbezogene, datenintensive Applikationen zu konzentrieren. Dies hat zur Entwicklung von Technologien geführt die, zugegebenermaßen, zwar mächtig, für die meisten Entwickler aber auch wenig vertraut und schwer verständlich sind. Zudem ist es in der Praxis sehr schwierig semantische Technologien schrittweise einzuführen, da sich das ihnen zugrunde liegende Datenmodell mit seiner „Open World Assumption“ grundlegend von dem unterscheidet was die meisten Entwickler gewohnt sind und somit verwenden. RDF/XML, das zum Höhepunkt von XML's Popularität standardisiert wurde und lange Zeit das einzig standardisierte Datenformat des semantischen

Webs war, wird sogar von XML Befürwortern vermieden. RDF/XML wurde weder für Menschen, noch für Maschinen optimiert und ist mit XML Werkzeugen nahezu nicht zu verarbeiten; was wahrscheinlich das größte Problem in der Praxis darstellt. Daher kann RDF/XML zweifellos als eine der größten Hindernisse zur Akzeptanz des semantischen Webs gezählt werden. Ein weiterer, kritischer Aspekt ist, dass das „Web“ im „semantischen Web“ wenig Aufmerksamkeit erhielt. Anstatt wie das traditionelle, dokumentenbasierte Web auf Hypermedia zu setzen und ein Netzwerk aus Daten zu schaffen, verwendet das Resource Description Framework (RDF) Internationalized Resource Identifiers (IRIs) oft als nicht-dereferenzierbare Identifikatoren anstatt voll auf URLs zu setzen. Dies hat Tim Berners-Lee 2006 dazu bewegt die sogenannten „Linked Data Principles“ [BL06] zu formulieren. Einfach gesagt, plädierte Berners-Lee für die Verwendung von dereferenzierbaren URLs und das Verknüpfen von Daten. Leider werden jedoch nach wie vor die meisten Daten im semantischen Web nicht als Linked Data oder in Form von Web APIs veröffentlicht, sondern in Form von „nur-lese“ Archiven oder zentralistischen SPARQL Endpunkten zur Verfügung gestellt.

Während es in den vergangenen Jahren mehrere Vorschläge gab um die genannten Probleme von Web APIs und des semantischen Webs zu lösen, gibt es bisher keine praxistaugliche Lösung. Eine interessante Beobachtung bei der Analyse des aktuellen Stands der Technik war, dass die meisten Lösungsvorschläge entweder zu einfach oder übermäßig komplex sind.

Lösungen im ersten Lager beschränken sich in der Regel auf einfache CRUD APIs (Create, Retrieve, Update, Delete). Obwohl in der Tat die meisten APIs theoretisch in so einem Stil realisiert werden könnten, ist die Semantik dieser Operationen in verteilten Systemen ohne zentrale Koordinierung zu schwach. Es reicht nicht aus zu wissen wie man eine Entität erstellt, es ist auch nötig zu wissen welche Konsequenzen eine Erstellung nach sich zieht. Zum Beispiel ist es ein wesentlicher Unterschied ob die Erstellung einer Bestellungsentität die Lieferung von Ware nach sich zieht oder nicht.

Im Gegensatz zu diesen zu sehr vereinfachten Technologien, werden die Lösungen im zweiten Lager durch in der Praxis fragwürdigen Funktionalitäten übermäßig komplex. Eine große Zahl von Lösungen der semantischen Web Forschung zum Beispiel enthält Beschreibungen der nicht-funktionalen Aspekte eines Services um die automatische Auswahl des passendsten Services zu ermöglichen. Per se scheint das eine vernünftige Entscheidung zu sein, in der Praxis jedoch wird die einheitliche Beschreibung abstrakter Eigenschaften verschiedenster Services sehr schnell zu komplex. Des Weiteren wird die Entscheidung welcher Service letztendlich verwendet wird häufig nicht aufgrund objektiver Eigenschaften getroffen, sondern aufgrund subjektiver Empfindungen wie dem Ruf des API Betreibers, das Verhältnis zwischen den verschiedenen Firmen oder anderen wettbewerbsbestimmenden Faktoren. Ich vertrete daher die Meinung, dass es essenziell ist den richtigen Kompromiss zwischen Einfachheit und Ausdrucksstärke zu finden. Diese Dissertation [La14] konzentriert sich daher auf funktionale Aspekte, stellt aber durchaus auch sicher, dass die Lösungen erweiterbar genug gehalten werden um komplexere Funktionalitäten in Zukunft einfach nachrüsten zu können. Als unerlässlich wird es auch betrachtet, dass der Ansatz stufenweise eingeführt werden kann. Bereits existierende Services sollten

mit minimalen Änderungen nachgerüstet werden können was sicherstellt, dass vorhandene Investitionen sowie existierende Werkzeuge weiter genutzt werden können.

Nach genauerer Analyse der besprochenen Probleme sowie dem aktuellen Stand der Technik lautete meine Hypothese, dass es möglich sein müsste, die Probleme durch die Standardisierung webbasierter Technologien für Maschine-zu-Maschine-Kommunikation und Verarbeitung strukturierter Daten (dessen Semantik von Maschinen „verstanden“ werden kann) lösen zu können. In anderen Worten war es das Ziel die Kluft zwischen REST und Linked Data zu schließen und Entwickler bei der Erstellung, der Dokumentation und der Benutzung von Web APIs zu unterstützen. Dabei sollte sowohl die Produktivität von Entwicklern gesteigert werden, als auch die Qualität und die Wiederverwendbarkeit der Web APIs und der durch sie zugänglich gemachten Daten verbessert werden.

Im ersten Schritt wurde die Realisierbarkeit des Zieles durch das Experimentieren mit bewährten, standardisierten Technologien überprüft. Das dadurch entstandene Design wurde SAPS [LG11c] genannt, was für „Semantic AtomPub-based Services“ steht. Es basiert, wie der Namen suggeriert, unter anderem auf dem Atom Syndication Format sowie dem dazugehörigen Atom Publishing Protocol—Technologien die häufig als Beispiele für vorbildliches RESTful Design zitiert werden. Zusammen mit OpenSearch, sind es einige der wenigen Ansätze die eine relativ große Akzeptanz erlangt haben. Wohl bewusst der Tatsache das XML, auf dem all die genannten Technologien basieren, nicht für den Austausch strukturierter Daten, sondern für die Auszeichnung von gemischten Inhalten entwickelt wurde und dass Atom oft eine zu starre Struktur verlangt, wurden dennoch diese Technologien für die Basis der ersten Experimente gewählt um rasch ein „Minimum Viable Product“ zu erstellen das semantische Technologien mit Technologien die in RESTful Services eingesetzt werden kombiniert.

Durch die Spezifikation von nur zwei Attributen bildet SAPS ein erweiterbares Framework existierender Technologien das die Erstellung einfacher CRUD-basierter APIs ermöglicht. Diese kommen ohne zusätzliche Dokumentation aus, da die Interaktionen bereits durch die verwendeten Technologien standardisiert wurden. Für anspruchsvollere APIs müssen jedoch zusätzliche Interaktionen definiert werden was bedeutet, dass für solche APIs sehr wohl zusätzliche Dokumentation nötig ist. In der Praxis stellte sich die von Atom diktierte starre Struktur als noch problematischer heraus als ursprünglich angenommen. Zusätzlich erwies sich das Design, das auf XML, XML Schema, Atom usw. basierte, als zu komplex und schwergewichtig. Die Rückmeldungen von Entwicklern machten klar, dass in der Praxis ein leichtgewichtigeres und flexibleres Design nötig ist. Die grundlegende Idee der Kombination von semantischen Technologien und Technologien von RESTful Web APIs erwies sich jedoch als aussichtsreich und somit wurde mit der Verfeinerung des Designs begonnen. Bei der Suche nach alternativen Serialisierungsformaten wurde dann relativ schnell auf JSON gesetzt, das zu der Zeit zunehmend an Popularität gewann.

Der Vorteil von JSON liegt darin, dass es ist den meisten Programmiersprachen direkt in native Speicherstrukturen deserialisiert werden kann und somit die gleiche „Struktur“ behält, was in Folge die Verarbeitung erleichtert. Es werden also keine komplexen Schnittstellen benötigt um mit JSON zu arbeiten. Dies zieht natürlich das Risiko einer engen Kopplung zwischen Clients und Servern nach sich wenn intern beide auf dieselbe Daten-

struktur setzen. Deshalb wurde versucht Möglichkeiten zu finden diese enge Kopplung aufzubrechen. SEREDASj—eine Sprache zur Beschreibung von *SEmantic REstful DAta Services*—war das Ergebnis dieser Bemühungen.

Ähnlich wie SAPS wurde SEREDASj [LG11a] zunächst für einfache CRUD-basierte Services optimiert, aber anstatt Entwickler zu zwingen ihre Services mit Atom und XML (neu) zu implementieren, konzentriert sich SEREDASj auf das Beschreiben von existierenden, auf JSON basierenden Services. SEREDASj ist daher kein Datenformat (oder eine Erweiterung eines solchen), sondern eine Beschreibungssprache. SEREDASj Beschreibungen dokumentieren die Semantik der JSON Repräsentationen und die Beziehungen zwischen den verschiedenen Ressourcen. Es wurden auch Algorithmen entwickelt die es erlauben mittels SEREDASj beschrieben Daten von Web APIs nach RDF zu konvertieren, mit SPARQL zu bearbeiten und die Änderungen zurück an den Service zu schicken [LG11b]. Das verschiebt die Kopplung von den ausgetauschten JSON Strukturen auf die semantische Ebene der Daten die sich nicht nur sehr viel seltener ändert, sondern auch einfacher zwischen verschiedenen Services geteilt werden kann.

Obwohl ein großes Augenmerk auf die Einfachheit von SEREDASj gesetzt wurde, musste in der Praxis festgestellt werden, dass der Ansatz suboptimal war. Entwickler hatten Schwierigkeiten mit der Trennung der Daten in JSON Repräsentationen und ihren Beschreibungen. Der separate Graph von SEREDASj Beschreibungen, der über den von der Web API ausgegeben JSON Dokumenten liegt, erhöhte die kognitive Belastung von Entwicklern beträchtlich. Entwickler beschwerten sich auch, dass die Syntax, die dem Vorbild von JSON Schema folgt, nicht kompakt genug sei und dass die Verwendung von semantischen Annotationen, die für die Realisierung von Web APIs, die über die CRUD Funktionalität hinausgehen, nötig sind, problematisch und schwer verständlich ist, da weder ein Vokabular noch Leitfäden dazu existierten. Das führte zur Einsicht, dass es im Generellen besser ist mehr Informationen in die Nachrichten (Repräsentation) selbst zu stecken als in komplexe Beschreibungen oder Mappings zu investieren. Es wurde auch klar, dass eine Trennung der Lösung in ein generisches Datenformat und ein Vokabular, das die nötigen Konzepte eindeutig definiert, von Entwicklern bevorzugt wird. Aufgrund der Erfahrungen und Erkenntnisse, die durch die Experimente mit SAPS und SEREDASj gewonnen werden konnten, war es schlussendlich möglich die zwei finalen Technologien JSON-LD und Hydra zu entwickeln.

Zusätzlich zu den Funktionalitäten die JSON bereits mit sich bringt, unterstützt JSON-LD [SKL14] Hyperlinks, einheitliche Identifikatoren für Entitäten und deren Attributen in Form von IRIs, Internationalisierung, die Definition und den Gebrauch von beliebigen Datentypen, die Unterscheidung zwischen unsortierten Mengen und geordneten Listen sowie die Möglichkeit Daten in mehrere benannte Graphen aufzuteilen. Diese Funktionalitäten vereinfachen nicht nur die Integration von Daten, was eines der grundlegenden Probleme in der Verwendung von Web APIs darstellt, sondern ermöglichen es Entwicklern auch ihre Daten mit einer viel expliziteren, maschinenlesbaren Semantik auszudrücken. Somit ist JSON-LD im Grunde ein neues Serialisierungsformat für das semantische Web. Um der traditionellen Abneigung von Entwicklern gegenüber semantischer Technologien gegenüber zu treten, war es ein grundlegendes Designkriterium den

Aufwand zum Erstellen und Verstehen von JSON-LD Dokumenten für Entwickler so gering als möglich zu halten. Sehr viel Zeit wurde daher in die Einfachheit, die Knappheit und die Lesbarkeit von JSON-LD investiert. Des Weiteren wurden Funktionalitäten in JSON-LD integriert, die es mit sehr wenig Aufwand ermöglichen existierende JSON Dokumente in selbstbeschreibende JSON-LD Dokumente zu verwandeln.

Neben der Spezifikation des Serialisierungsformats, wurden auch mehrere Algorithmen und eine Programmierschnittstelle die die Arbeit mit JSON-LD Dokumenten vereinfacht entworfen. Die „JSON-LD 1.0 Processing Algorithms and API“ Spezifikation [LKS14] definiert auch wie JSON-LD zu RDF und RDF zu JSON-LD konvertiert werden kann. Da beide Spezifikationen am World Wide Web Consortium (W3C) als offizielle Internet Standards verabschiedet wurden, besteht die Möglichkeit, dass JSON-LD, ähnlich wie HTML, das ebenfalls am W3C standardisiert wurde und die dominierende Sprache im dokumentenbasierten Web ist, die Lingua franca für Web APIs wird. Die Erfahrungen aus der Verwendung von JSON-LD in einem großangelegten „Internet der Dinge“ Projekt sowie das positive Feedback der zahlreichen frühzeitigen Anwendern bestätigen, dass JSON-LD ein praxistauglicher Ansatz ist. In der Tat könnte JSON-LD ein erster Schritt auf dem Weg zur Standardisierung von semantischen, RESTful Web Services werden und die Basis für die verschiedenen Anstrengungen bilden, die bisher keinen gemeinsamen Nenner zu finden schienen. In diesem Kontext ist es auch nochmals erwähnenswert, dass JSON-LD selbst keine vollständige Lösung darstellt, sondern zusätzlich ein Vokabular, das die nötigen Konzepte zur Realisierung einer Anwendung definiert, benötigt.

Hydra [LG13] ist so ein Vokabular. Sein Ziel ist es die Konzepte die häufig in Web APIs benötigt werden zu definieren um damit die Basis für hypermediabasierte Services, die von komplett generischen Clients verwendet werden können, zu bilden. Die grundlegende Idee dabei war es ein leichtgewichtiges Vokabular zu schaffen, das es einem Server ermöglicht seinen Clients zur Laufzeit die möglichen Operationen und Navigationspfade mitzuteilen. Ein Client interagiert mit einem Service dann in einem Prozess der in jedem Schritt die vom Server angezeigten Operationen analysiert und diejenige auswählt die ihn näher an sein Ziel bringt. Hydra, dessen Konzepte in Abbildung 1 dargestellt sind, erlaubt es dem Client in maschinenlesbarer Form mitzuteilen wie er die nötigen HTTP Nachrichten erstellen kann um den Status des Servers so zu beeinflussen, dass er sein anwendungsspezifisches Ziel erreichen kann. Da all diese Informationen dynamisch zur Laufzeit ausgetauscht werden anstatt direkt im Client eingebaut zu werden, wird die Kopplung von Clients und Servern aufgelöst—was auch die gewünschte Generalisierung von Clients mit sich bringt. Auf zusätzliche Dokumentation oder die Erstellung von Vokabularen kann weitgehend verzichtet werden da zusätzlich Konzepte aus beliebigen existierenden Vokabularen wie zum Beispiel Schema.org wiederverwendet werden können. Dies reduziert nicht nur den Arbeitsaufwand für Entwickler, sondern verbessert auch die Interoperabilität und reduziert die Preisgabe von Implementierungsdetails, was das Risiko einer engen Kopplung zwischen Client und Server maßgeblich reduziert. Um diese Eigenschaften zu verifizieren, wurde eine komplette Entwicklungsumgebung für JSON-LD und Hydra implementiert.

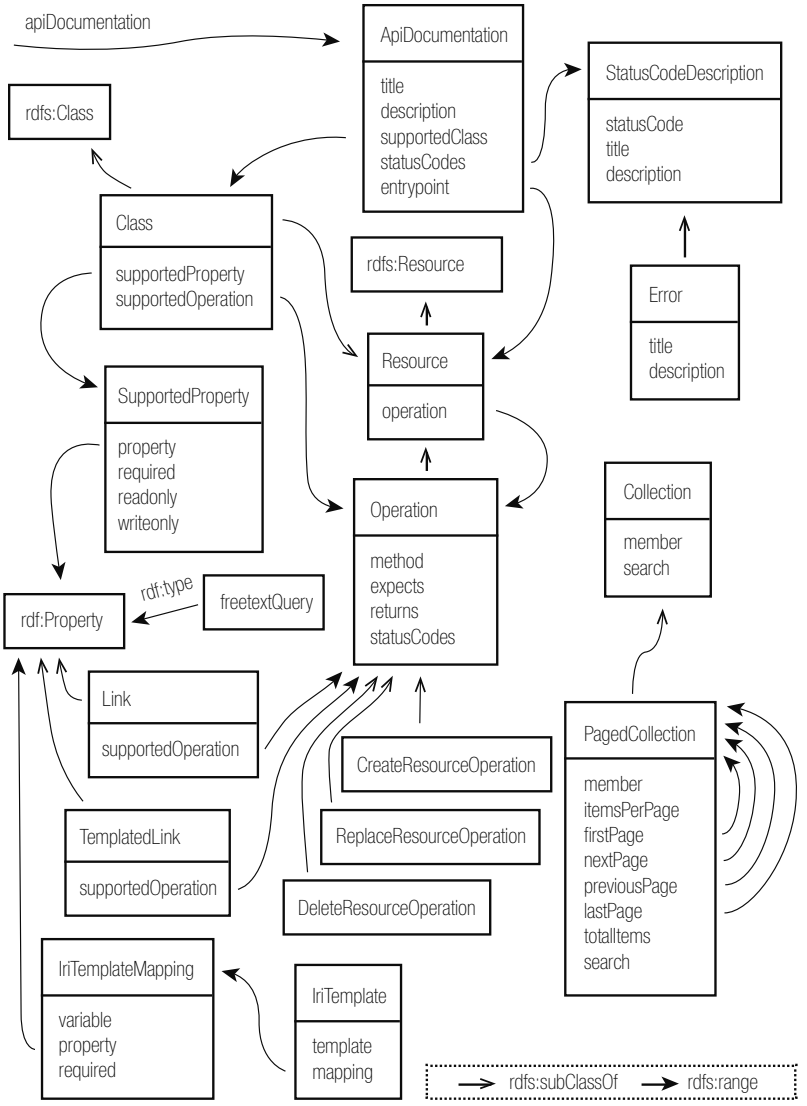

Abb. 1: Das Hydra Core Vokabular

Neben der Integration beider Technologien in ein populäres Webframework wurde ein komplett generischer API Browser, der es auch Laien ermöglicht mit solchen Web APIs zu interagieren, sowie ein generischer API Client für den programmatischen Zugriff entwickelt. Diese Werkzeuge wurden dann benutzt um exemplarisch mehrere Web APIs zu implementieren um somit die Benutzbarkeit der Technologien zu überprüfen und um zu evaluieren ob die eingangs erwähnten Probleme mit ihnen gelöst werden können. Es konn-

te gezeigt werden, dass JSON-LD und Hydra lose gekoppelte Lösungen für die identifizierten Probleme bieten. Da beide Ansätze auf RDF, einem mächtigem Datenmodell basieren, müssen sich Entwickler weder um die Definition von Datenmodellen noch von Datenformaten kümmern. Stattdessen können sie sich voll auf die Modellierung der anwendungsspezifischen Funktionalität konzentrieren und diese mit standardisierten und interoperablen Technologien umsetzen. Falls gewünscht, öffnen JSON-LD und Hydra auch die Tür zu weiteren semantischen Technologien wie Quad Stores, SPARQL Query Engines und Reasonern. Im Gegensatz zu anderen semantischen Technologien zwingen die vorgestellten Lösungen Entwickler aber nicht dazu. Sie können durchaus auch mit populären NoSQL Datenbanken wie MongoDB oder Elasticsearch verwendet werden. Dies macht sie zu sehr flexiblen Technologien die den meisten Entwicklern vertraut wirken und eine graduelle Aktualisierung bereits existierender Infrastrukturen ermöglichen. Der gesamte Quellcode den ich im Rahmen meiner Forschung erstellt habe, wurde als Open Source freigegeben. Mittlerweile hat die Entwicklergemeinde zahlreiche weitere Werkzeuge und Programmbibliotheken für JSON-LD und Hydra erstellt, so dass diese nun in den meisten Programmiersprachen umfassend verwendet werden können.[3]

Die klare Trennung der Belange (separation of concerns) hilft die Komplexität bei gleich bleibender Funktionalität so gering als möglich zu halten. Entwickler die zum Beispiel die Funktionalität von Hydra nicht benötigen, da sie ausschließlich einen nur-lese Zugriff auf ihre Daten realisieren wollen, können JSON-LD ohne Hydra verwenden. Analog dazu kann Hydra, falls nötig oder gewünscht, mit anderen Serialisierungsformaten wie Turtle oder RDFa verwendet werden.

Diese lose Kopplung war von enormer Bedeutung für die Akzeptanz der Technologien—eine Kenngröße die in Forschungsprojekten leider viel zu häufig vernachlässigt wird. In meiner Dissertation ist die Akzeptanz hingegen ein kritischer Bestandteil der Evaluierung der erarbeiteten Lösungen. Die meisten Technologien—und ganz besonders Web Technologien—sind weitgehend nutzlos, falls sie nicht verwendet werden. Ich bin daher in den verschiedenen Entwicklungsstadien sehr aktiv an verschiedene Benutzergruppen herangetreten um Feedback zu sammeln und Entwickler über die Technologien zu informieren. Dies war von enormer Bedeutung um sicherzustellen, dass praxisrelevante Lösungen erarbeitet werden. Die ständige Interaktion mit sogenannten Innovatoren und Early Adoptern [Ro03] half dann schlussendlich auch eine kritische Masse an Interesse zu bilden, die JSON-LD bei seiner Fertigstellung dann schließlich zum Durchbruch verhalf.

JSON-LD wurde mittlerweile in mehreren Forschungsprojekten verwendet—von großangelegten EU-Forschungsprojekten wie dem Interactive Knowledge Stack [Eu], das eine Referenzarchitektur für semantische Content Management Systeme erarbeitete, zu kleineren Projekten wie dem der Educational System Group[4] der Galileo Universität Guatemala, die an einem „Cloud Educational Interoperability Service“ forscht der Web APIs populärer Online-Tools mit JSON-LD und Hydra nachrüstet um es so auch Laien zu ermöglichen personalisierte Lernumgebungen zu erstellen. Neben diesen Anwendungen in der For-

[3] Eine Liste frei verfügbarer JSON-LD Prozessoren kann auf http://json-ld.org/ gefunden werden, Hydra-spezifische Software ist hingegen auf http://www.hydra-cg.com/ aufgelistet.

[4] http://ges.galileo.edu/geswiki/research

schung, wurde JSON-LD auch bereits in Produkte von Konzernen wie Google, Microsoft, Yahoo, Yandex oder der BBC, um nur einige zu nennen, integriert. Diese Produkte bringen die erarbeiteten Technologien in die Hände von hunderten Millionen von Benutzern die sie tagtäglich—und zwar meist ohne ihr Wissen—verwenden. Google zum Beispiel benutzt JSON-LD unter anderem um die Websuche zu verbessern und Benutzern von Gmail produktivitätssteigernde Funktionen anzubieten. Eine einfache Überschlagsrechnung hilft dabei das Potenzial zu illustrieren. Geht man sehr konservativ davon aus, dass die durch die erarbeiteten Technologien ermöglichten Funktionalitäten einem Gmail Benutzer auch nur ein Minute pro Monat sparen helfen, so werden für die über 500 Millionen Gmail Nutzer in Summe pro Monat weit über 4.000 Personenjahre eingespart.

Nachdem JSON-LD mittlerweile ein offizieller Standard des World Wide Web Consortiums (W3C) ist, bauen auch bereits mehrere Standardisierungsprojekte darauf auf [La14]. Eine so rasche Verbreitung und Standardisierung wäre wohl nicht zu erreichen gewesen, wenn die beiden Ansätze in eine Technologie verschmolzen worden wären. Hydra, die jüngere der beiden Technologien, ist noch nicht so weit verbreitet wie JSON-LD, gewinnt aber rasch an Boden. Die W3C Community Group in die Hydra's weitere Entwicklung ausgelagert wurde, ist äußerst aktiv und wächst stetig. Mittlerweile ist sie mit weit mehr als hundert Teilnehmern unter den zehn größten W3C Community Groups. Zudem konnte Hydra das Design wichtiger Aspekte von Schema.org beeinflussen.

Zusammenfassend ist die Kombination aus JSON-LD und Hydra meines besten Wissens die erste praxistaugliche und akzeptierte Lösung die die Lücke zwischen semantischen Technologien und Technologien, die in RESTful Services eingesetzt werden, schließt. Die erarbeitete Lösung vereint die Stärken beider Welten und ist somit in der Lage die Probleme, auf die Entwickler von Web APIs häufig stoßen, zu lösen. Dies verbessert hoffentlich nicht nur Web APIs, sondern das Web als Ganzes und geht damit weit über meine bescheidenen Beiträge zur Verbesserung existierender Standards wie HTTP/1.1 [FR14] und RDF 1.1 [CWL14] hinaus. JSON-LD wird mittlerweile in Fachkreisen als die „Einstiegsdroge" des semantischen Webs gehandelt und hat durchaus das Potenzial diesem nach über einer Dekade Forschung zum Durchbruch zu verhelfen. Die technische und wissenschaftliche Bedeutung der erarbeiteten Technologien wird durch deren Einsatz in Millionen von Websites und Produkten von im Web marktführenden Konzernen sowie der Standardisierung am W3C belegt.

Literaturverzeichnis

[BL06] Berners-Lee, Tim: Linked Data. In: Design Issues for the World Wide Web. 2006.

[CWL14] Cyganiak, Richard; Wood, David; Lanthaler, Markus: RDF 1.1 Concepts and Abstract Syntax. In: W3C Recommendation. 2014.

[Eu] European Commission: IKS: Interactive knowledge stack for small to medium CMS/KMS providers. In: Community Research and Development Information Service (CORDIS).

[Fi00] Fielding, Roy Thomas: Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine, 2000.

[FR14] Fielding, Roy T.; Reschke, Julian F.: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. In: Internet Engineering Task Force (IETF) Draft, Jgg. 2616. 2014.

[IB] IBM: Bringing big data to the enterprise.

[La14] Lanthaler, Markus: Third Generation Web APIs—Bridging the Gap between REST and Linked Data. Doctoral dissertation, Graz University of Technology, Austria, 2014.

[LG11a] Lanthaler, Markus; Gütl, Christian: A Semantic Description Language for RESTful Data Services to Combat Semaphobia. In: Proceedings of the 2011 5th IEEE International Conference on Digital Ecosystems and Technologies (DEST). IEEE, Daejeon, Korea, 2011.

[LG11b] Lanthaler, Markus; Gütl, Christian: Aligning Web Services with the Semantic Web to Create a Global Read-Write Graph of Data. In: Proceedings of the 9th IEEE European Conference on Web Services (ECOWS 2011). IEEE, Lugano, Switzerland, 2011.

[LG11c] Lanthaler, Markus; Gütl, Christian: SAPS: Semantic AtomPub-based Services. In: Proceedings of the the 11th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT). IEEE, Munich, Germany, 2011.

[LG13] Lanthaler, Markus; Gütl, Christian: Hydra: A Vocabulary for Hypermedia-Driven Web APIs. In: Proceedings of the 6th Workshop on Linked Data on the Web (LDOW2013) at the 22nd International World Wide Web Conference (WWW2013). CEUR-WS, Rio de Janeiro, Brazil, 2013.

[LKS14] Lanthaler, Markus; Kellogg, Gregg; Sporny, Manu: JSON-LD 1.0 Processing Algorithms and API. In: W3C Recommendation. 2014.

[Ma13] Manyika, James; Chui, Michael; Groves, Peter; Farrell, Diana; Kuiken, Steve Van; Doshi, Elizabeth Almasi: Open data: Unlocking innovation and performance with liquid information. McKinsey Quarterly, (October), 2013.

[Ro03] Rogers, Everett M.: Diffusion of Innovations. Free Press, New York, NY, USA, 5. Auflage, 2003.

[SKL14] Sporny, Manu; Kellogg, Gregg; Lanthaler, Markus: JSON-LD 1.0 - A JSON-based Serialization for Linked Data. In: W3C Recommendation. 2014.

[Wa94] Waldo, Jim; Wyant, Geoff; Wollrath, Ann; Kendall, Sam: A Note on Distributed Computing. Bericht, Sun Microsystems Laboratories, Inc., Mountain View, California, USA, 1994.

Markus Lanthaler wurde 1984 in Italien geboren. Nach dem Abschluss der Hochschulreife zog er 2003 nach Österreich um an der Technischen Universität Graz Telematik zu studieren. Nach einem Auslandsaufenthalt in Helsinki, Finnland schloss Lanthaler sein Masterstudium 2009 mit Auszeichnung ab und begann ein Doktoratsstudium in Informatik. Dieses führte Lanthaler an die Curtin University in Perth, Western Australia bevor er 2014 an der Technischen Universität Graz mit Auszeichnung promovierte. Markus Lanthaler ist Autor mehrerer wissenschaftlicher Publikationen und weitgenutzter Internetstandards sowie Vorsitzender der Hydra W3C Community Group.