RESEARCH PROPOSAL

Automated RESTful API Documentation

S M Sohan
Supervisor: Dr. Frank Maurer
Supervisory committee members: Dr. Mea Wang, Dr. Craig Anslow
The University of Calgary
Department of Computer Science
Calgary, Alberta

October 2015

Contents

Abstract

API documentation presents both a problem and an opportunity for API usability. Most developers learn to use an API based on its documentation and at the same time faces obstacles using the APIs due to lack of necessary information in the documentation [Robillard, 2011].

RESTful APIs provide interconnectivity among network based software applications. Documentation of a RESTful API is a key information source for software developers that use these types of API. As a RESTful API evolves, the documentation needs to be maintained alongside. This implies that RESTful API developers also need to efficiently generate and maintain the documentation of their APIs with the qualities that make the API usable. This introduces the problem of interest in this research: how to document RESTful APIs? This is a problem because there is a lack of scientifically established guidelines and automated tool support for RESTful API documentation.

The key challenges are to identify the unique documentation needs of RESTful APIs, and to find and evaluate a technique to provide a cost-effective alternative compared to the currently available solutions that require manual effort or bespoke implementations. In my research, I identify the requirements for RESTful API documentation and introduce a novel technique using an HTTP proxy server. In a preliminary evaluation of a prototype implementation called *SpyREST*, I have found SpyREST can be used to auto-generate more accurate documentation of RESTful APIs compared to manually written official API documentation. I plan to conduct further usability and user evaluation of the aforementioned technique to evaluate the effectiveness of the proposed technique by involving RESTful API developers from the industry.

Chapter 1

Introduction and Motivation

Application Programming Interfaces, commonly known as APIs, are used to express a software component in terms of its operations, inputs, outputs, and their types[1]. Robillard defines an API as follows: An API is the interface to implement functionality that developers can access to perform various tasks [Robillard and DeLine, 2011] [Robillard, 2011]. APIs enable multiple software components to interact with each other.

RESTful APIs are a type of APIs that are used to integrate software components using web technologies. Fielding defined Representational State Transfer or REST as an architectural style for developing distributed hypermedia systems [Fielding, 2000]. Fielding derived the REST architectural style from several other architectural styles of developing network-based applications based on the following essential constraints: client-server, stateless, cache, uniform interface, and layered system. RESTful APIs provide an abstraction of the underlying system by using system specific resources such as documents, images, etc. and unique identifiers to access the resources. Using RESTful APIs, systems perform actions on the resources by transferring a representation of the resources between various systems. For example, the GitHub RESTful API[2] has a resource called *Repository* to denote a code repository hosted on GitHub. A repository owned by a GitHub user can be identified by the URL `/user/repos`. To create a new *Repository* for a user, the GitHub API can be invoked via HTTP *POST* at `/user/repos` using the following JSON representation of a *Repository*:

---

[1]`https://en.wikipedia.org/wiki/Application_programming_interface`
[2]`https://developer.github.com/v3/repos/#create`

```
{
  "name": "Hello-World",
  "description": "This is your first repository",
  "homepage": "https://github.com",
  "private": false,
  "has_issues": true,
  "has_wiki": true,
  "has_downloads": true
}
```

In todays world of technology, RESTful APIs have become ubiquitous and the primary choice for integrating Internet enabled applications due to its simplicity and similarity with HTTP [Mangler et al., 2010]. For example, realtor.ca[3], a real estate listing website, uses a RESTful API from walkscore.com[4] to provide the relative walk score of a selected property. To show a selected property on the map alongside other points of interest, realtor.ca uses a RESTful API from Bing Maps[5]. By incorporating the map and walk score using RESTful APIs, realtor.ca provides important information to their users. Both walkscore.com and Bing Maps have published documentation for their RESTful APIs. Most RESTful APIs, including these two examples, are often documented manually or using custom implemented tools specific to the APIs that are not publicly available. This requires effort to generate and maintain the documentation of RESTful APIs over time since there is a lack of reusable tool support.

Previous research in the area of API usability mostly focused on local APIs such as Java libraries. Researchers identified the documentation of APIs as both the primary source of information as well as the key obstacle for API usability [Robillard, 2011]. Hence the quality of the API documentation plays an important role in API usability. To this regard, researchers have identified the qualities of "good API documentation" as follows: complete, correct, includes thorough explanations and code examples, provides consistent presentation and organization [Robillard, 2011, Myers et al., 2010]. Today, there are several tools such as Junit[6], UsETeC [Zhu et al., 2014], Jadeite [Stylos et al., 2009], APIMiner [Montandon et al., 2013], Roast [Hoffman and Strooper, 2003] that can be used to document local APIs with the aforementioned qualities
.

---

[3]www.realtor.ca
[4]www.walkscore.com
[5]www.bing.com/maps
[6]http://junit.org/

While there are overlaps between the documentation requirements of local and RESTful APIs, there are significant requirements that are unique to each. For example, RESTful API documentation needs to include information about its API endpoints, HTTP headers, and request and response payloads. On the other hand, documentation of local APIs need to explain the classes and methods. Due to such differences between local and RESTful APIs, the existing technique and tools for local API documentation cannot be readily used to generate RESTful API documentation.

First, the current state of documentation of RESTful APIs needs to be analyzed to identify the qualities that define usable RESTful API documentation. Then, the existing techniques and tool support for RESTful API documentation needs to be compared against the identified qualities or requirements to solve the problems that can be addressed using new techniques and tools.
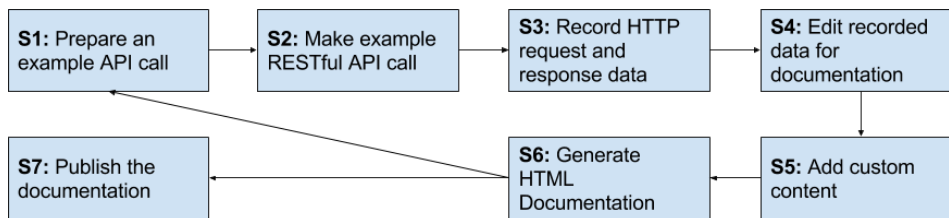

Figure 1.1: Manual RESTful API Documentation Steps

Currently, the process of documenting RESTful APIs are largely manual. Conceptually, the high level steps involved in the manual process can be identified as shown in Figure 1.1. These steps S1-S7 can be described as follows: S1 - API developer prepares an example API with the required HTTP parameters and request body, S2 - API developer uses a REST API client to make the example API call, S3 - the API developer records the HTTP data , S4 - API developer then edits the recorded data so that only relevant content is selected for documentation, S5 - API developers use any custom content to describe the API example, S6 - API developers combine the custom content with the edited content from the HTTP traffic into HTML to publish to the web, and S7 - API developer adds custom overview information to explain API concepts and general rules, and publishes the final documentation so other developers can learn the API. The steps S1-S6 are repeated for every API action that needs to be documented, and S1-S7 need to repeated for every version of the RESTful API as it evolves. As depicted here, this manual process can be both time consuming and error-

prone. Hence, tool support is required to automate the process.

To solve the problem of RESTful API documentation, first the unique requirements for RESTful API documentation need to be identified. To this regard, I analyzed the existing literature on documentation for both local APIs and RESTful APIs. I also analyzed the current industry practices on RESTful APIs to identify challenging documentation problems that are observed in the industry [Sohan et al., 2015a]. By comparing and combining the findings from the literature review and industry practices, I have established a list of unique requirements related to RESTful API documentation. I devised a novel approach based on an HTTP Proxy server to meet the requirements. I have developed a prototype implementation of the technique called SpyREST to demonstrate the approach. The planned evaluation of the tool will provide a report on the perceived viability of the tool among RESTful API developers from the industry and a method for evaluating RESTful API documentation tools.

Through this research, I aim to achieve the following contributions:

1. Scientifically establish a list of requirements for RESTful API documentation.

2. Present a novel technique to solve the established requirements.

3. Implement and evaluate the effectiveness of the proposed technique via SpyREST.

1.1 Research Context and Scope

My research context is depicted in Figure 1.2. My dissertation topic belongs to the area of Software Engineering,"the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" [iee, 1990]. Within Software Engineering, my work is related to Service Oriented Architecture (SOA) - an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network[7]. There are several protocols that can be used to implement SOA such as RPC, SOAP, and REST. In my research, I focus on RESTful APIs because of its widespread adoption in the current industry trend [Mangler et al., 2010]. Within this area, my research specifically falls under RESTful API Documentation.

---

[7]https://en.wikipedia.org/wiki/Service-oriented_architecture

Figure 1.2: Research Context
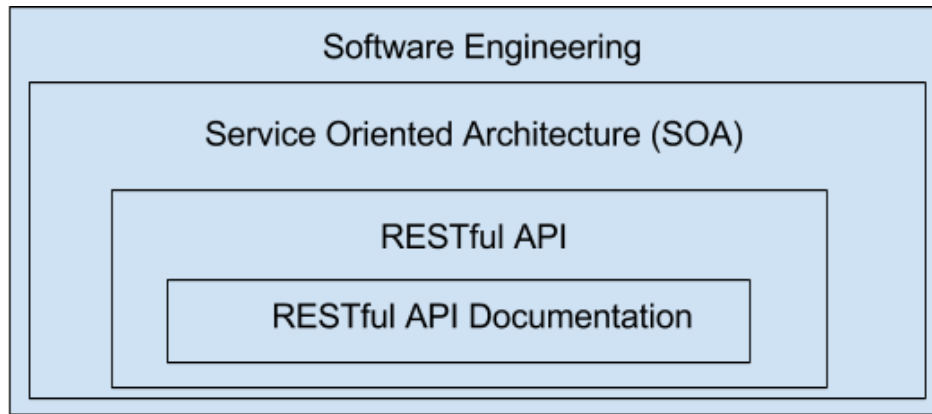
The scope of my dissertation topic is as follows: identifying the requirements related to RESTful API Documentation for API usability, and developing and evaluating a cost-effective technique and tool to solve the identified requirements. The scope for this research includes the documentation of all RESTful APIs, irrespective of the technology that may be used to implement the APIs.

Chapter 2

Background

To achieve the aforementioned goals, I begin by discussing existing literature on generic API usability and documentation. Then I provide a literature review on RESTful APIs to identify the unsolved problems related to the documentation of RESTful APIs.

2.1 API Usability and Documentation

2.1.1 Case Studies

API usability is a qualitative concept derived from the characteristics that make an API easy to use. Several papers in the existing literature have focused on identifying the characteristics that make an API usable. Robillard studied API usability by surveying 83 software developers at Microsoft [Robillard and DeLine, 2011]. Robillard found that 78% of the survey participants read API documentation to learn the APIs, 55% used code examples, 34% experimented with the APIs, 30% read articles, and 29% asked colleagues. While API documentation is used as the principal source of information about how to use an API, Robillard et al. found that the most severe API learning obstacles are related to the API documentation For API usability, Robillard suggested the following requirements as must-have for API documentation: include good examples, be complete, support many example usage scenarios, be conveniently organized, and include relevant design elements. These requirements for general purpose API documentation can also be applied to RESTful API documentation.

Zibran et al. analyzed bug repositories for 562 API usability related bugs from five different projects and found that 27.3% of the reported

bugs are API documentation bugs [Zibran et al., 2011]. Scheller et al. provided a framework for objectively measuring API usability based on the number and types of different objects and methods that the API provides [Scheller and Kühn, 2015].

Kuhn performed a user study with 19 professional software developers to understand requirements for tool development to support API learnability [Kuhn and DeLine, 2012]. Kuhn recommended the following as requirements for API documentation: trustworthiness, confidentiality, lack of information overload and the need for code examples as first-class documentation artifacts. Shi et al. studied the evolution of API documentation for five different Java projects. They presented a quantitative analysis of the types of changes with impacts on API behaviors and the frequency of the API elements that change between versions [Shi et al., 2011]. Shi et al. observed a large number of API documentation changes related to polishing the custom content, (i.e. fix typos, and API usage examples). They recommended API documentation tools to support editors for custom content to provide usage tips and simple ways to include API usage examples without syntax errors.

2.1.2 Tool Support

Several authors introduced tool support for including usage examples with API documentation. Hoffman et al. recommended using executable examples in API documentation [Hoffman and Strooper, 2003]. They introduced Roast test as tool support to combine prosaic descriptions of Java APIs along with executable code examples in a unified API documentation. Montandon et al. developed APIMiner as a search tool for Java and Android APIs and recommended providing production-like API usage examples in the API documentation [Montandon et al., 2013]. They observed that 35% of API related web searches performed on APIMiner included the term example inferring that developers search for source code examples while using API documentation. Zhu et al. developed an Eclipse plugin called UsETeC to extract API usage examples by automatically synthesizing JUnit test code of the APIs [Zhu et al., 2014]. Stylos presented Jadeite as an IDE plug-in that combines a few techniques to provide developers with faster access to relevant API documentation [Stylos et al., 2009]. Jadeite uses placeholders for API elements such as classes and method names that developers commonly expect to exist, but the actual classes or methods are named differently. When developers search for placeholder API elements, Jadeite shows links to the API documentation of the actual API elements and finds relevant usage examples from Google code search. For deployment of API

documentation, Stepalina identified several advantages of using a Software as a Service (SaaS) tool including lower cost, better quality, and reusability [Stepalina, 2010].

2.1.3 Collaboration

Several authors presented techniques for linking official API documentation with crowd-sourced API usage examples that is otherwise fragmented. Nasehi et al. recommended mining knowledge repositories such as Stack-Overflow and developer forums should be considered for retrieving useful code examples [Nasehi et al., 2012]. They also recommended providing wiki-like features so API users can contribute with relevant API examples to the documentation of the APIs in a way so that the official and crowd-sourced sections of the documentation can be easily distinguished. Parnin et al. found that examples of 87.9% of all jQuery API methods are found by searching software development blogs and forums [Parnin and Treude, 2011]. They found these crowd-sourced posts about the APIs also got a median of 8 comments per post which enables an author-reader collaboration. Wu et al. presented an Eclipse plugin called CoDocent that can automatically find code examples using various online code search engines and link with the relevant official API documentation [Wu et al., 2010]. Chen et al. presented a technique to automatically link official documentation with crowd-sourced documentation by recording the API related web searches that are performed by developers [Chen and Zhang, 2014].

2.1.4 Presentation

The presentation and organization of API documentation has been discussed in several papers. Maalej et al. identified the patterns of knowledge in API documentation by analyzing 5574 randomly sampled API documentation units [Maalej and Robillard, 2013]. Their identified patterns of knowledge are as follows: functionality, concepts, directives, purpose, quality, control, structure, patterns, examples, environment, references, and non-info. Dagenais et al. presented a technique and a tool called RecoDoc to link code-like elements with their corresponding code elements in the API documentation [Dagenais and Robillard, 2012]. They mine the API support channel data such as API mailing lists and developer forums to map code-like elements against actual API elements. Ko et al. performed a study on the role of conceptual knowledge on API usability [Ko and Riche, 2011]. They found that thorough introductions to the concepts, standards and ideas in API

documentation are a prerequisite for developers to be able to effectively use an API.

Table 2.1: Characteristics of usable API documentation

Detailed introduction	To include thorough explanations of the API elements, business models, and associated rules
Includes Examples	To help the API client developers understand various use-cases of the APIs. Executable examples are recommended so that API client developers exercise the API examples.
Automated	To be auto-generated using reusable tools for better accuracy and cost-effectiveness.
Collaborative	To augment official API documentation with relevant crowd-sourced knowledge.
Presentation	Consistent presentation and organization of the aforementioned types of contents. Reduced information overload Excluding confidential information

Table 2.1 presents a summary of the desired characteristics for API documentation of usable APIs as identified in the existing literature. These characteristics are also important for the usability of RESTful APIs. But the aforementioned techniques and tools that can be used to document local APIs to satisfy these desired characteristics are not easily applicable for documenting RESTful APIs. Through my research I aim to find an innovative solution for the documentation of usable RESTful APIs defined by these characteristics.

2.2 RESTful API Documentation

2.2.1 Case Studies

Researchers performed case studies to identify challenges associated with RESTful APIs and proposed solutions to overcome some of the identified challenges.

Maleshkova analyzed the state of RESTful APIs and found that most RESTful APIs are manually documented which results in API underspecification, and a lack of support for common tasks and reusable tools [Maleshkova et al., 2010].

Maleshkova found that more than 75% of the RESTful APIs included API examples which contributes to the ease of development work for the API client developers. Myers et al. performed a user study on the usability of a complex API for enterprise SOA to understand the obstacles faced by the target users of the API [Myers et al., 2010]. They identified the visual presentation of the API documentation played a role in API usability. Specifically, they recommended providing a consistent look-and-feel with explanation for the starting points and an overall map comprising of both text and diagrams, providing a browsing experience with breadcrumb trail following a hierarchy, an effective search interface, providing example code and a way to exercise the examples online without writing code.

In a case study of current industry practices on Web APIs, we have found the documentation of RESTful APIs are performed manually or using bespoke tools [Sohan et al., 2015a]. We observed the documentation of the studied RESTful APIs to commonly include summary information and API examples, with optional description of the structure of API requests and responses. We found some API providers also include a custom built live API explorer where API client developers can exercise the API examples on a web browser without writing any code. Because manual approaches and custom built tools are used, it is expensive to provide the same features for documenting a RESTful API.

2.2.2 RESTful API Specification Languages

Several authors proposed terminologies and specifications for standardizing how RESTful APIs are documented. Hadley proposed WADL (Web Application Description Language) as an XML based language to describe RESTful APIs [Hadley, 2006]. SOAP (Simple Object Access Protocol) based web services use WSDL (Web Service Description Language) to describe and auto generate client code. WADL was designed to achieve the same goal for RESTful APIs. Magler et al. proposed RIDDL as an extension of WADL to solve the composition and evolution of RESTful APIs [Mangler et al., 2010].

Kopecky et al. proposed hRESTS, as an HTML based mico-format to describe RESTful APIs [Kopecky et al., 2008]. Machine readable RESTful API documentation can be generated by annotating HTML documentation of RESTful APIs following hRESTS specification. Verborgh et al. proposed RESTdesc as a language to define RESTful APIs using resources and links that connects the resources [Verborgh et al., 2013]. RESTdesc provides an imperative syntax to define API actions that have pre and post conditions so that one can understand the functional requirements for using the API

that is not otherwise found when a single API action is described in isolation. Danielsen presented a vocabulary to describe RESTful APIs using WIfL(Web Interface Language) [Danielsen and Jeffrey, 2013]. If a RESTful API is described using WIfL, then automated API clients can be used to make example API calls and validate API request and response data. Lei et al. presented OmniVoke as a tool to abstract out multiple RESTful APIs under a unified interface [Li et al., 2011]. OmniVoke relies on a mapping of existing API elements against a standard RESTful API so that a single entry point can be used to call multiple APIs that are implemented using different standards.

In addition to the existing literature, several RESTful API description languages have been proposed by industry practitioners such as RAML[1], Blueprint[2], and Swagger[3]. There are tools that can automatically convert and publish the description of RESTful APIs from these languages into HTML based RESTful API documentation. In addition to the documentation, there are tools to generate RESTful API client libraries in different programming languages from these API specifications.

Table 2.2: Characteristics of usable API documentation and RESTful APIs

Property	Current State
Detailed introduction	Manually edited contents are commonly used in RESTful API documentation.
Includes Examples	RESTful API documentations commonly include manually generated API examples since the existing tools do not support automatically recorded examples.
Automated	Tools exist that can generate documentation of RESTful APIs if described using custom specifications
Collaborative	Collaboration is typically found on API forums and StackOverflow.com
Presentation	Commonly includes RESTful API access information, resources, actions, their request and response structures and API examples.

Table 2.2 shows the current state of RESTful API documentation in

---

[1]http://raml.org/
[2]https://apiblueprint.org/
[3]http://swagger.io/

comparison to the aforementioned desired characteristics of usable API documentation. In summary, practitioners and researchers have attempted to solve the problem of RESTful API documentation by proposing candidate specifications to standardize the vocabulary and format of describing RESTful APIs. The process of generating the documentation for a RESTful API following these custom specifications is largely manual since no automated tools are available. These specification formats support describing the structure of different API elements, but no support is provided for API usage examples and executable API examples, an important feature for API learnability as identified by the researchers. Thus, there exists a need for further research on RESTful API documentation to satisfy the identified characteristics of API documentation for API learnability. Through my research, I aim to fulfill this need by learning from the currently available solutions and finding innovative techniques and tools for RESTful API documentation.

Chapter 3

Research Questions

Within the context of RESTful API documentation and the scope as discussed before, my research aims to address the following three high level research questions.

3.1 RQ1: What are the requirements for usable RESTful API documentation?

Researchers have identified several important requirements for API usability primarily focusing on local APIs. RESTful APIs have unique characteristics compared to local APIs. As a result, the recommendations for local API documentation needs to be adapted and extended for RESTful API documentation to account for the differences between the two. A research challenge is to identify the differences in the documentation needs of RESTful and local APIs. To develop this understanding, I look for answers to the following research questions:

1. What are the essential requirements for usable RESTful API documentation?

2. How to adapt and extend the recommended characteristics for usable local API documentation to support the documentation of RESTful APIs?

3.2 RQ2: How to provide a cost-effective and reusable tool support so that RESTful API developers can create and maintain usable RESTful API documentation?

Researchers have introduced API documentation tools primarily focusing in local APIs. Several researchers and practitioners have proposed languages and specifications for describing RESTful APIs. There is a lack of reusable tool support that RESTful API developers can use to cost-effectively generate the custom specifications to document their RESTful APIs. This introduces a research challenge to find a viable automated technique for RESTful API documentation to find a cost-effective alternative to the manual process or the need for bespoke implementation. To improve the tool support, I investigate the following research questions:

1. What technique can be used to automatically document RESTful APIs?

2. What techniques can be used to provide reusable tool support for the documentation of RESTful APIs?

3.3 RQ3: How to evaluate a technique and tool for RESTful API documentation?

In the absence of the necessary tool support for RESTful API documentation, it is also not clear how to evaluate such a tool. A research challenge is to establish a scientific evaluation method for RESTful API documentation techniques to assure RESTful API developers about the merits and drawbacks of a technique and related tool supports. In the literature, several papers discussed the topic of empirical evaluation of software tools and techniques [Runeson and Höst, 2009, Sjøberg et al., 2002, Lethbridge et al., 2005, Kitchenham et al., 2008, Easterbrook et al., 2008]. Based on the evaluation methods identified by the researchers, specific evaluation scenarios need to be designed for RESTful API documentation tools. This introduces the following questions:

1. How to evaluate the cost-effectiveness of a RESTful API documentation tool?

2. How to evaluate the usability of a RESTful API documentation tool?

Chapter 4

Research Contributions and Methods

I aim to add to the body of existing knowledge by finding answers to the aforementioned research questions. Here, I explain the expected contribution of my research against the identified research questions RQ1-3.

4.1 RQ1: Literature reviews of the requirements for usable RESTful API documentation.

I start by reviewing the existing literature on API usability to understand the recommended characteristics of usable API documentation. I also review the existing literature on RESTful APIs to identify the problems and challenges specific to the documentation of RESTful APIs and their currently available solutions. In addition to reviewing the literature, I perform a case study involving multiple RESTful APIs to understand the current industry practices around RESTful API documentation. I combine the outcomes of the literature reviews and the case study to identify the essential requirements for usable RESTful API documentation. In particular, I aim to contribute with identifying the following information so that RESTful API developers can provide usable documentation for their APIs:

1. The types of content that are essential for usable RESTful API documentation.

2. The presentation and organization of usable RESTful API documentation.

3. The approaches that are required to generate RESTful API documentation.

4. The approaches that are required to maintain the documentation of evolving RESTful APIs.

In addition to listing the requirements for RESTful API documentation, I will identify differences between the requirements of usable local and RESTful API documentation. This will help tool developers to adapt and extend the currently available tool support for local API documentation to meet the requirements for RESTful APIs.

4.2   RQ2: Present a novel technique for automatically generating and maintaining RESTful API documentation with reusable tool support.

I investigate the currently available solutions to meet the identified requirements of usable RESTful API documentation. The currently available solutions are found from the existing literature as well as the tools found in the industry. I start by comparing currently available solutions to find the unmet requirements (RQ1) that require an adaptation of the available techniques or the introduction of new techniques. Then, I introduce the new techniques along with the adaptation of existing techniques that can be used to develop tool support for automated RESTful API documentation. The resultant is a mapping of the identified requirements and new techniques to solve the requirements for usable RESTful API documentation.

To demonstrate the newly proposed techniques, I develop a prototype API documentation tool that can be used by RESTful API developers. The development of the tool and the newly identified techniques are interdependent. Based on the challenges and opportunities that are discovered during the development phase, the proposed techniques are further adjusted and refined. I compare the output of the developed tool against the official documentation of multiple RESTful APIs from different providers to demonstrate the advantages of the newly proposed techniques.

4.3 RQ3: I will evaluate the proposed technique for RESTful API documentation.

I will explore the literature to understand the scientifically accepted methods of empirically evaluating software tools and techniques. I will also investigate the research methods that have been used by researchers for the evaluation of API documentation tools. Based on my findings, I will propose a research method that can be used to evaluate RESTful API documentation tools and techniques.

Once the research method is identified, I will propose a plan for performing an evaluation of RESTful API documentation tool. The plan will include both the specific goals and required study setup involving people, project, timeline, and feedback collection for the evaluation.

Following the plan, I will perform an evaluation to find the effectiveness of my proposed techniques and the prototype tools. I will use the feedback from the evaluation to refine and extend the essential requirements, and their associated proposed techniques and tool for usable RESTful API documentation. The evaluation will help RESTful API developers to use an improved tool support and the researchers to perform future research to advance the state of tool support for RESTful API documentation.

Chapter 5

Current Progress

In my first two years of the PhD program, I have made progress towards achieving the proposed contributions. The current progress can be explained through five different research projects that have been carried out during this time.

5.1 Research Project 1: Systematic Literature Review

I started my research by performing a systematic literature review following the guidelines as suggested by Kitchenham on the topic of web API evolution to understand the related challenges and their available solutions [Kitchenham and Charters, 2007]. In particular, the goals of the literature review was to understand the following:

1. Why do web services need to evolve?

2. How to evolve the web services?

3. What are the key challenges for evolving web services?

From the literature review I identified a list of open research problems related to evolving web APIs in the areas of documentation, implementation, deployment, and communication of API changes. Among these identified areas, I observed the documentation related challenges included the following problems: automatic API documentation, support for multiple versions, auto-generating the changelog between versions, and supporting interactive

exercisable API explorers for multiple API versions. I observed several papers in the literature proposed techniques to solve these problems primarily focusing on SOAP based Web APIs that have limited applicability to RESTful APIs. Based on this observation from the literature review, I selected my dissertation research topic to be the documentation of RESTful APIs.

5.2 Research Project 2: Case Study

To better understand the current industry practices around the documentation of RESTful APIs, I performed a case study on the evolution of nine RESTful APIs representing different industry domains, popularity levels, maturity levels, and open vs. closed source applications [Sohan et al., 2015a]. Through this case study, I have summarized and identified opportunities for improving the current industry practices about the versioning, documentation, and communication of API changes for evolving RESTful APIs.

From the case study, I found the RESTful APIs change frequently, often every week, and a lack of commonly used standards for versioning RESTful APIs. I also found the documentation of the RESTful APIs were largely manual and often only documented the most recent version even through multiple versions were in use. The changelogs are manually generated and disconnected from the API documentation. I observed multiple communication channels exist between RESTful API developers and the developers that use the APIs for collaboration and notification that are disconnected from the API documentation. Overall, I identified an opportunity for future research to focus on finding tools and techniques for RESTful API documentation to solve the observed problems through the case study.

5.3 Research Project 3: Finding a Technique for RESTful API documentation

To find a technique for RESTful API documentation, first, I compared the identified problems from the literature review and case study on web API evolution against the existing literature on API usability. The outcome of this comparison is an initial list of requirements for usable RESTful API documentation towards achieving proposed contribution on RQ1. I identified the following high level requirements as must-have for the usable documentation of RESTful APIs:

R1 Automated. The documentation needs to be accurately generated and

updated automatically to reduce cost and improve accuracy.

R2 Example based. The API features need to be explained using examples.

R3 Executable documentation. Developers that use the API should be able to execute the API examples.

R4 Version aware. There needs to be documentation for all the API versions that are supported.

R5 Customizable. The API documentation needs to include custom content for introductions and explaining any complex business rules.

R6 Reusable tools. Reusable tools need to be used so that any RESTful API can be documented without needing bespoke implementation.

R7 Collaborative. Documentation needs to be linked with crowd-sourced content to reduce knowledge fragmentation.

In this research, I proposed a technique to solve these initial set of requirements for RESTful API documentation [Sohan et al., 2015b]. At the heart of the technique is a customized HTTP proxy server. If example API calls are made against a RESTful API via the HTTP proxy server, it can automate the steps S3, S4, S6, and S7 of the manual process as described in Figure 1. The result is a smaller workflow using this technique as shown in Figure 5.1.
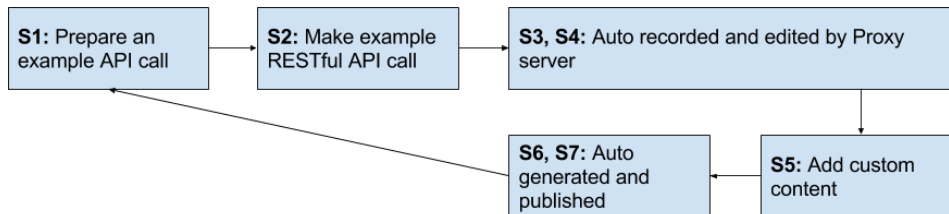
Figure 5.1: RESTful API Documentation Steps Using Proxy Server

The use of an HTTP proxy server for RESTful API documentation is a novel idea and it offers several advantages over existing solutions for both initial generation and ongoing maintenance of RESTful APIs. Table 5.1 summarizes the key differences between existing solutions and the proposed HTTP Proxy server based techniques:

Table 5.1: Comparison of existing techniques vs. proposed technique for RESTful API documentation

Requirement	Existing Solutions	HTTP Proxy server based solution
R1. Automated	Partially automated. Requires manual effort from API developers to describe RESTful APIs using custom intermediate API specification formats.	Automated. HTTP proxy server records HTTP traffic from example API calls and automatically generates the documentation without needing an intermediate specification of the API.
R2. Example Based	Manual. API developers need to manually add examples to the documentation.	Automated. The HTTP proxy server automatically adds the recorded API examples.
R3. Executable Documentation	Manual. API developers need to manually provide executable API examples.	Automated. The auto-recorded API examples are converted to executable documentation.
R4. Version Aware	Partially automated. The intermediate API specifications support API versions, for a new version the API developers need to manually generate the specifications.	Automated. API examples run against a new version via the HTTP proxy automatically generates the documentation for the new version.
R5. Customizable	Manual.	Manual.
R6. Reusable tools	Partially automated. RESTful API documentation can be generated and published using reusable tools if an API is defined using supported specifications formats.	Automated. RESTful API documentation can be auto-generated for any RESTful API by running example API calls using the proxy.
R7. Collaborative	Developers discuss API related issues across a variety of mediums including API forums, and StackOverflow.com.	The user interface of the auto-generated API documentation provides collaboration features.

5.4   Research Project 4: Prototype Tool Development

To find the feasibility and perceived effectiveness of the devised technique, I have implemented SpyREST as a prototype tool [Sohan et al., 2015c]. SpyREST is composed of three different components as shown in Figure 5.2.
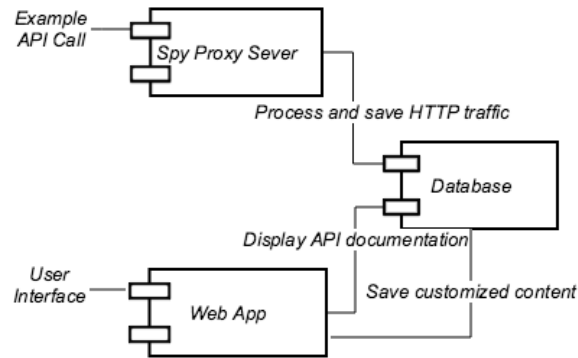

Figure 5.2: SpyREST Components

5.4.1   The Spy

This component is a special purpose HTTP proxy server that can record and post-process HTTP traffic from example JSON based RESTful API calls into usable API documentation. Based on Figure 5.1, the Spy is responsible for S3 and S4. The Spy saves the recorded HTTP traffic in a structured database that can be used to render the documentation for RESTful APIs. The Spy automatically filters HTTP authorization headers to prevent the leakage of API access credentials into the documentation. To reduce information overload, the Spy automatically truncates large arrays in API example responses. As a result, the Spy automates the repeated parts of the RESTful API documentation process. For evolving APIs, a rerun of the same API examples update the auto-generated documentation. The Spy component satisfies the following requirements: automated, example based, version-aware, customizable, and a reusable solution for RESTful API documentation.

5.4.2   The Web App

The web app complements the features of the Spy to fulfill the rest of the identified must-have requirements for RESTful API documentation. The web app displays RESTful API documentation based on the data recorded by the Spy and implements the steps S5, S6, and S7 as shown in Figure 5.1. Additionally, the web app allows RESTful API developers to add custom content via a wiki-like editor to describe overview information and thorough explanations about business rules related to the APIs that are not derivable solely based on example API calls. For each API example, the web app uses the recorded data from the Spy and displays a cURL[1] command that the API client developers can use to execute the API without writing any client code.

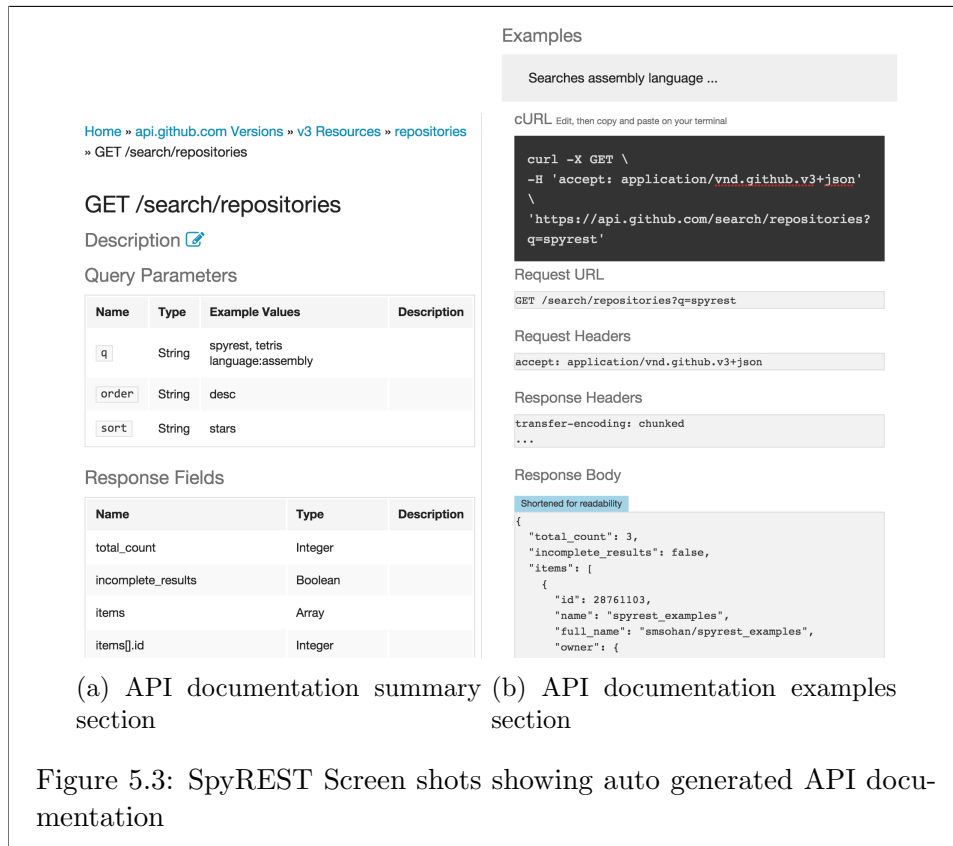(a) API documentation summary section

(b) API documentation examples section

Figure 5.3: SpyREST Screen shots showing auto generated API documentation

To facilitate collaboration, the web app provides in-page commenting

---

[1] http://curl.haxx.se/

throughout the site. The web app component implements solutions for the following requirements: executable examples, customizable, reusable, and collaborative documentation for RESTful APIs. Figure 5.3 shows a screenshot of the web app.

5.4.3   Database

The database component plays a supporting role so that the Spy and the web app can leverage from a single database. The database stores all the auto-recorded information from the Spy and custom edited content from the web app.

5.5   Research Project 5: Preliminary Evaluation

I have performed a preliminary evaluation of SpyREST and found it to produce more accurate documentation of RESTful APIs compared to the official documentation. In this preliminary evaluation, I have used SpyREST to auto-generate the documentation of 25 RESTful API actions from three different products. To run the example API calls for this evaluation, I have used automated test code that provides both test coverage as well as updated API documentation. In 5 of the 25 API actions, SpyREST documented API response fields that were not included the official API documentations, resulting into more up to date documentation of the APIs.

This prototype implementation has been productized and is being used as a RESTful API documentation tool at Cisco Systems, Inc.[2]. In the remainder of my research, I plan to conduct additional evaluations of SpyREST from this production usage and involving RESTful API developers from the industry to better understand the effectiveness of the HTTP Proxy server based approach of automated RESTful API documentation.

---

[2]`https://api-docs.amp.sourcefire.com/`

Chapter 6

Conclusion

Through my dissertation research, I plan to improve the body of knowledge on RESTful API documentation to help software developers and researchers build usable RESTful APIs. This research is positioned based on the existing literature on API usability and RESTful APIs, to identify and solve unique challenges observed in the current industry practices related to RESTful API documentation. My research will help understand the requirements for usable RESTful API documentation and provide novel techniques and tool support with their evaluation to solve the identified requirements. This has implications for both researchers and RESTful API developers. With the extended knowledge, researchers will be able to perform future research to find alternate techniques and tool support for RESTful API documentation. Software developers can use the developed tool from this research to cost-effectively generate and maintain the documentation for their RESTful APIs.
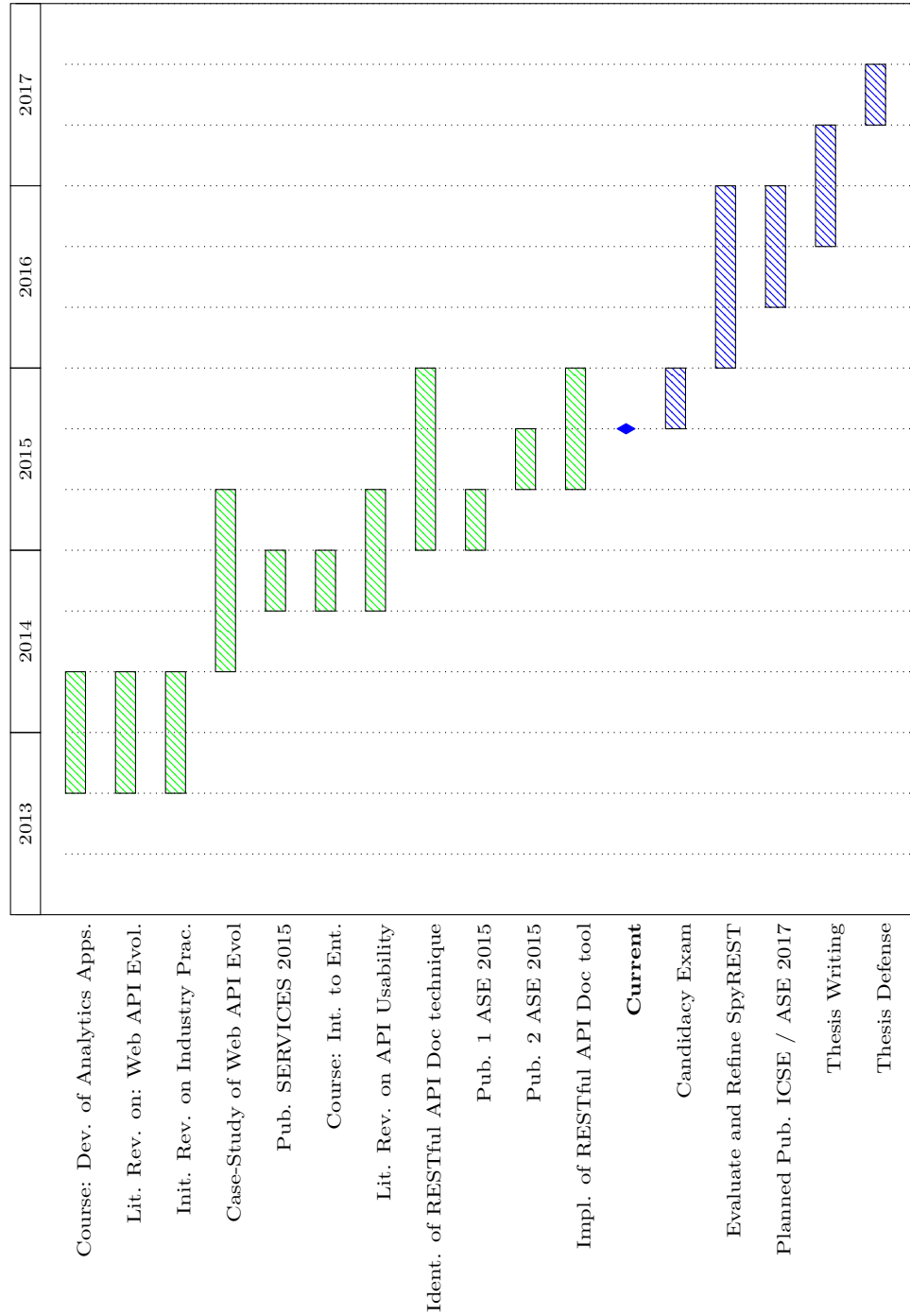
Chapter 7

Milestones

The following list summarizes the important milestones that I have completed and plan to complete during the period of my dissertation research. The future milestones are subjected to change based on the early feedbacks found from the evaluation of the proposed technique and SpyREST.

Semester	Tasks
Fall 2013 - Winter 2014	Courses (Development of Analytics Applications)
	Literature review on evolving Web APIs
	Initial review of current industry practices on Web API evolution
Spring 2014 - Summer 2014	Case study of multiple Web API evolution
Fall 2014	Courses (Introduction to Entrepreneurship)
	Literature review on API usability
	Summarize the case-study of Web API Evolution
	Published: SERVICES 2015
	A Case-Study of Web API Evolution
Winter 2015	Identification of a technique for RESTful API documentation
	Initial implementation of SpyREST
	Preliminary evaluation of SpyREST

Semester	Tasks
Spring - Summer 2015	Courses (Patterns in Software Engineering)
	Refinement of prototype tool implementation
	Ethics approval for evaluating SpyREST
	Published: ASE 2015
	SpyREST: Automated RESTful API Documentation using an HTTP Proxy Server
	SpyREST in Action: An Automated RESTful API Documentation Tool
Fall 2015	Reading list for candidacy examination
	Writing research proposal document for candidacy examination
	Candidacy examination
Winter 2016	Performing an evaluation of SpyREST
Summer - Fall 2016	Summarizing the results of evaluation
	Refining SpyREST based on evaluation
	Planned Publications: ICSE / ASE 2017
	Experience report on using SpyREST
	User-study evaluation of an automated RESTful API documentation technique
Winter - Spring 2017	Thesis writing
Summer 2017	Thesis defense

Table 7.1: Milestones

Bibliography

[iee, 1990] (1990). IEEE standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84.

[Chen and Zhang, 2014] Chen, C. and Zhang, K. (2014). Who asked what: Integrating crowdsourced faqs into api documentation. In *Companion Proceedings of the International Conference on Software Engineering*, ICSE Companion 2014, pages 456–459. ACM.

[Dagenais and Robillard, 2012] Dagenais, B. and Robillard, M. P. (2012). Recovering traceability links between an api and its learning resources. In *Proc. of 2012 International Conference on Software Engineering (ICSE)*, pages 47–57. IEEE.

[Danielsen and Jeffrey, 2013] Danielsen, P. and Jeffrey, A. (2013). Validation and interactivity of web API documentation. In *International Conference on Web Services*, pages 523–530. IEEE.

[Easterbrook et al., 2008] Easterbrook, S., Singer, J., Storey, M.-A., and Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*, pages 285–311. Springer.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

[Hadley, 2006] Hadley, M. J. (2006). Web application description language (wadl).

[Hoffman and Strooper, 2003] Hoffman, D. and Strooper, P. (2003). API documentation with executable examples. *Journal of Systems and Software*, 66(2):143 – 156.

[Kitchenham et al., 2008] Kitchenham, B., Al-Khilidar, H., Babar, M. A., Berry, M., Cox, K., Keung, J., Kurniawati, F., Staples, M., Zhang, H., and Zhu, L. (2008). Evaluating guidelines for reporting empirical software engineering studies. *Empirical Software Engineering*, 13(1):97–121.

[Kitchenham and Charters, 2007] Kitchenham, B. A. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

[Ko and Riche, 2011] Ko, A. J. and Riche, Y. (2011). The role of conceptual knowledge in api usability. In *Proc. of 2011 Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 173–176. IEEE.

[Kopecky et al., 2008] Kopecky, J., Gomadam, K., and Vitvar, T. (2008). hRESTS: An HTML microformat for describing RESTful web services. In *Proc. of International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 1, pages 619–625.

[Kuhn and DeLine, 2012] Kuhn, A. and DeLine, R. (2012). On designing better tools for learning APIs. In *Search-Driven Development - Users, Infrastructure, Tools and Evaluation (SUITE), 2012 ICSE Workshop on*, pages 27–30.

[Lethbridge et al., 2005] Lethbridge, T. C., Sim, S. E., and Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. *Empirical software engineering*, 10(3):311–341.

[Li et al., 2011] Li, N., Pedrinaci, C., Maleshkova, M., Kopecky, J., and Domingue, J. (2011). OmniVoke: A framework for automating the invocation of web APIs. In *Proc. of IEEE International Conference on Semantic Computing*, pages 39–46.

[Maalej and Robillard, 2013] Maalej, W. and Robillard, M. P. (2013). Patterns of knowledge in api reference documentation. *Software Engineering, IEEE Transactions on*, 39(9):1264–1282.

[Maleshkova et al., 2010] Maleshkova, M., Pedrinaci, C., and Domingue, J. (2010). Investigating web APIs on the world wide web. pages 107–114.

[Mangler et al., 2010] Mangler, J., Beran, P. P., and Schikuta, E. (2010). On the origin of services using RIDDL for description, evolution and composition of RESTful services. In *Proc. of International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 505–508.

[Montandon et al., 2013] Montandon, J. E., Borges, H., Felix, D., and Valente, M. T. (2013). Documenting APIs with examples: Lessons learned with the APIMiner platform. In *Proc. of 2013 Working Conference on Reverse Engineering (WCRE)*, pages 401–408. IEEE.

[Myers et al., 2010] Myers, B. A., Jeong, S. Y., Xie, Y., Beaton, J., Stylos, J., Ehret, R., Karstens, J., Efeoglu, A., and Busse, D. K. (2010). Studying the documentation of an api for enterprise service-oriented architecture. *J. Organ. End User Comput.*, 22(1):23–51.

[Nasehi et al., 2012] Nasehi, S. M., Sillito, J., Maurer, F., and Burns, C. (2012). What makes a good code example?: A study of programming Q&A in StackOverflow. In *Proc of. IEEE International Conference on Software Maintenance*, pages 25–34.

[Parnin and Treude, 2011] Parnin, C. and Treude, C. (2011). Measuring API documentation on the web. In *Proceedings of the International Workshop on Web 2.0 for Software Engineering*, (Web2SE), pages 25–30. ACM.

[Robillard, 2011] Robillard, M. (2011). What makes APIs hard to learn? the answers of developers. *Software, IEEE*, PP(99):1–1.

[Robillard and DeLine, 2011] Robillard, M. and DeLine, R. (2011). A field study of API learning obstacles. *Empirical Software Engineering*, 16(6):703–732.

[Runeson and Höst, 2009] Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164.

[Scheller and Kühn, 2015] Scheller, T. and Kühn, E. (2015). Automated measurement of api usability: The API concepts framework. *Information and Software Technology*, 61:145–162.

[Shi et al., 2011] Shi, L., Zhong, H., Xie, T., and Li, M. (2011). An empirical study on evolution of API documentation. In *Proc. of Conference on Fundamental Approaches to Software Engineering (FASE)*, volume 6603, pages 416–431. Springer.

[Sjøberg et al., 2002] Sjøberg, D. I., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A., Koren, E. F., and Vokác, M. (2002). Conducting realistic experiments in software engineering. In *Proc. of 2002 International Symposium on Empirical Software Engineering*, pages 17–26. IEEE.

[Sohan et al., 2015a] Sohan, S., Anslow, C., and Maurer, F. (2015a). A case study of web api evolution. In *Proc. of 2015 IEEE World Congress on Services (SERVICES)*, pages 245–252. IEEE.

[Sohan et al., 2015b] Sohan, S., Anslow, C., and Maurer, F. (2015b). Spyrest: Automated restful api documentation using an http proxy server. In *Tp Appear in Proc. of International Conference on Automated Software Engineering.* IEEE/ACM.

[Sohan et al., 2015c] Sohan, S., Anslow, C., and Maurer, F. (2015c). Spyrest in action: An automated restful api documentation tool. In *To Appear in Proc. of International Conference on Automated Software Engineering.* IEEE/ACM.

[Stepalina, 2010] Stepalina, E. (2010). SaaS support in software documentation systems. In *Proc. of 2010 Central and Eastern European Software Engineering Conference (CEE-SECR)*, pages 192–197.

[Stylos et al., 2009] Stylos, J., Myers, B. A., and Yang, Z. (2009). Jadeite: Improving API documentation using usage information. In *'09 Extended Abstracts on Human Factors in Computing Systems*, (CHI EA), pages 4429–4434. ACM.

[Verborgh et al., 2013] Verborgh, R., Steiner, T., Van Deursen, D., De Roo, J., Walle, R. d., and Gabarr Valls, J. (2013). Capturing the functionality of web services with functional descriptions. *Multimedia Tools and Applications*, 64(2):365–387.

[Wu et al., 2010] Wu, Y.-C., Mar, L. W., and Jiau, H. C. (2010). Codocent: Support api usage with code example and api documentation. In *Proc. of 2010 International Conference on Software Engineering Advances (IC-SEA)*, pages 135–140. IEEE.

[Zhu et al., 2014] Zhu, Z., Zou, Y., Xie, B., Jin, Y., Lin, Z., and Zhang, L. (2014). Mining api usage examples from test code. In *Proc. of 2014 IEEE International Conference on Software Maintenance and Evolution (ICSME),*, pages 301–310. IEEE.

[Zibran et al., 2011] Zibran, M. F., Eishita, F. Z., and Roy, C. K. (2011). Useful, but usable? factors affecting the usability of APIs. In *Proc. of 2011 Working Conference on Reverse Engineering (WCRE)*, pages 151–155. IEEE.

Appendix

- Sohan, S., Anslow, C., and Maurer, F. (2015). SpyREST: Automated RESTful API Documentation using an HTTP Proxy Server. To appear in Proc. of IEEE/ACM International Conference on Automated Software Engineering.

- Sohan, S., Anslow, C., and Maurer, F. (2015). SpyREST in Action: An Automated RESTful API Documentation Tool. To appear in Proc. of IEEE/ACM International Conference on Automated Software Engineering.

- Sohan, S., and Maurer, F. (2014). Practical Challenges with Web Service Evolution: A Review of Literature and Current Industry Practices