RESEARCH PROPOSAL

Automated RESTful API Documentation

S M Sohan
Supervisor: Dr. Frank Maurer
The University of Calgary
Department of Computer Science
Calgary, Alberta

October 2015

0.1 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]. As a result, the documentation of API has become an active research topic to improve API usability.

RESTful APIs provide interconnectivity among network based software applications. Using RESTful APIs applications can deliver valuable features and enrich user experiences by leveraging third-party services that are otherwise expensive to develop and maintain. Documentation of a RESTful API is a key information source for software developers that use the API. As a RESTful API evolves, the documentation also needs to be maintained alongside. So, RESTful API developers need to 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.

My research is focused on finding a viable solution for RESTful API documentation so that RESTful API developers can document any RESTful API with a reusable tool. To this regard, I identify a set of requirements for usable RESTful API documentation by studying the existing literature and current industry practices on documentation of RESTful APIs. To meet these requirements I introduce a novel technique using an HTTP Proxy server to automatically generate and maintain the documentation for RESTful APIs. I demonstrate and present a preliminary evaluation of a HTTP Proxy server based technique that implements the requirements to solve the RESTful API documentation problem through SpyREST, a prototype implementation. In a preliminary evaluation, I have found the HTTP proxy server based technique can be used to generate more accurate documentation of RESTful APIs compared to manually written official API documentation. I plan to conduct usability and user evaluation of the aforementioned technique to validate the proposed technique by involving RESTful API developers from the industry.

0.2 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. APIs enable multiple software components to interact with each other. RESTful APIs are a subclass 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. In todays world of technology, RESTful APIs have become mainstream and the primary choice for integrating internet enabled applications due to its simplicity and similarity with HTTP [Mangler et al., 2010].

RESTful APIs are used by a wide variety of software applications to integrate and deliver enhanced customer value. For example, realtor.ca, a real estate listing website, uses a RESTful API from www.walkscore.com to provide the relative walk score of a selected property. To show a selected property on the map alongside other points of interest, www.realtor.ca uses a RESTful API from www.bing.com/maps. By incorporating the map and walk score using RESTful APIs, www.realtor.ca provides important information to their users. Both www.walkscore.com and www.bing.com/maps have published documentation for their RESTful APIs. Most RESTful APIs, including these two examples, are often documented manually or using custom implemented tools. 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. 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. Today, there are several tools that can be used to document local APIs with the aforementioned qualities.

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 infor-

mation 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 the 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 a usable RESTful API documentation. Then, the existing techniques and tool support for RESTful API documentation needs to be compared against the identified qualities to solve the problems that can be addressed using new techniques and tools.

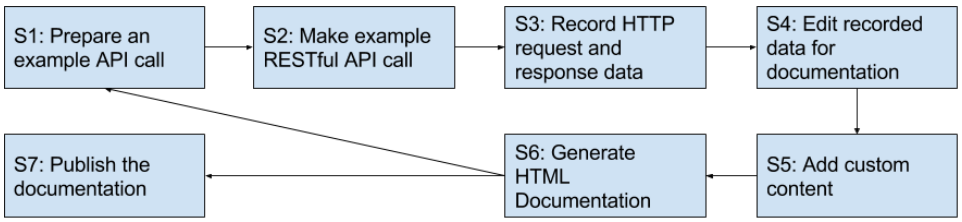


Figure 1: Manual RESTful API Documentation Steps

Currently, the process of documenting RESTful APIs are largely manual. The high level steps involved in the manual process is shown in Figure 1. These steps S1-S7 can be described as follows: S1 - API developer prepares an example API will 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 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 is needs to repeated for every version of the RESTful API as it evolves. As depicted here, this manual process can be both expensive and error-prone and 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 prac-

tices on RESTful API documentation to identify challenging problems that are observed in the industry. 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. A prototype implementation of the technique called SpyREST is developed 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 framework 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 a prototype of the identified technique.
4. Evaluate the effectiveness of the presented technique for RESTful API documentation.

0.3 Research Context and Scope

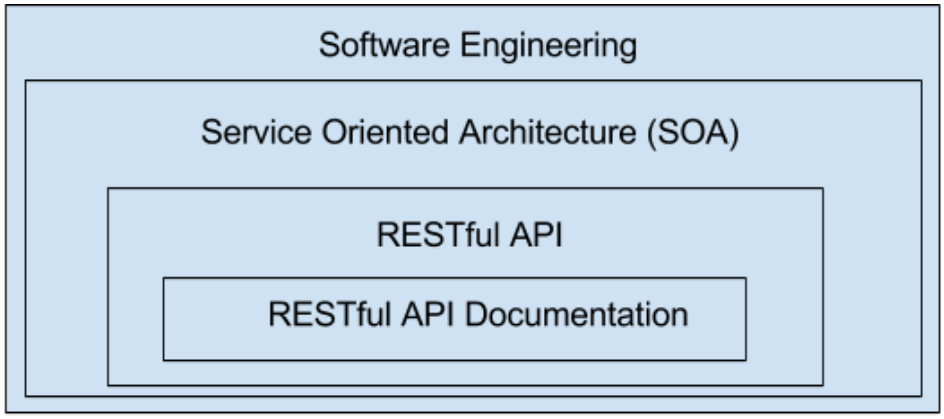


Figure 2: Research Context

My research context is depicted in Figure 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. 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. 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. Within this area, my research specifically falls under RESTful API Documentation.

The scope of my dissertation topic is as follows: identifying the requirements related to RESTful API Documentation for API usability, and developing and evaluating techniques and tools 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. Tool development is scoped to proof of concept only. As such, my research will not focus on developing production-ready tools.

0.4 Background

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

0.4.1 API Usability and Documentation

Robillard defines 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]. 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 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 [Robillard and DeLine, 2011] Żibran 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]. 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.

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 a tool support to combine prosaic description 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].

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 StackOverflow 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].

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, 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.

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].

Table 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.
Consistent Presentation	Consistent presentation and organization of the aforementioned types of contents.

Table 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.

0.4.2 RESTful API Documentation

Researchers have 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 into API underspecification, and a lack of support for common tasks and reusable tools [Maleshkova et al., 2010]. Maleshkova found that more than 75

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 layer of adapter [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 researchers, 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.

Myers et al. performed a user study on the usability of a complex API for enterprise service oriented architecture 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 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 exercisable 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.

0.5 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.

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

0.5.1 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. There is a lack of understanding regarding what and where to adapt and extend the local API specific recommendations for usable API documentation. 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?

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

The tool support for RESTful API documentation is rather limited. Researchers have identified the commonly used and important content for usable API documentation, and introduced 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 automatically generate the custom specifications to document their RESTful APIs. This results into a manual process or bespoke implementation of tools that are expensive to develop and maintain over time. 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?

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

In absence of the necessary tool support for RESTful API documentation, it is also not clear how to evaluate such a tool. A scientific evaluation method is required 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. 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 problems:

1. What research method needs to be followed to evaluate a RESTful API documentation technique?

2. How to conduct an evaluation of a RESTful API documentation tool?

0.6 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 problems RQ1-3.

0.6.1 RQ1: I will perform literature reviews to identify 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.

0.6.2 RQ2: I will 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 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.

0.6.3 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 to state of tool support for RESTful API documentation.

0.7 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 four different research projects that have been carried out during this time.

0.7.1 Research Project 1: Systematic Literature Review

I started my research by performing a systematic literature review on the topic of web API evolution to understand the related challenges and their available solutions. 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.

0.7.2 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. 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 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.

0.7.3 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 doc-

umentation towards achieving proposed contribution on RQ1. I identified the following high level requirements as must-have for the usable documentation of RESTful APIs:

1. Automated. The documentation needs to be generated and updated automatically to reduce cost and improve accuracy.
2. Example based. The API features need to be explained using examples.
3. Executable documentation. Developers that use the API should be able to exercise the API examples.
4. Version aware. There needs to be documentation for all the API versions that are supported.
5. Customizable. The API documentation needs to include custom content for introductions and explaining any complex business rules.
6. Reusable tools. Reusable tools need to be used to eliminate the need for bespoke tool development.
7. 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. At the heart of the technique is a customized HTTP proxy server called Spy. 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 resultant is a smaller workflow using this technique is as shown in Figure 3.

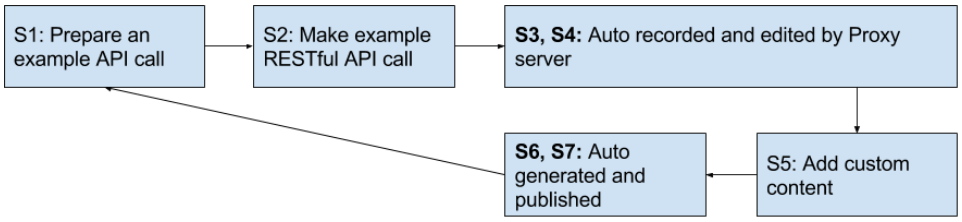


Figure 3: RESTful API Documentation Steps Using Proxy Server

This reduced workflow introduces automation to the process of RESTful API documentation. The steps for creating good example API calls

and adding custom content are best done manually since it requires human judgement. This novel technique offers several advantages over the existing solutions for RESTful API documentation. For example, unlike existing techniques that require manual effort to create an intermediate representation of the API using custom specifications for generating final HTML documentation, the proxy server directly converts the recorded HTTP data into the documentation. Because this is a HTTP proxy server, the technique is applicable to all RESTful APIs irrespective of the technology that is used to implement the APIs or the API clients that are used to make the example API calls. By automating the process of recording and processing HTTP traffic for documentation, the documentation can be organically updated as the API evolves by replaying the examples. Overall, this technique can satisfy the identified must-have requirements for RESTful API documentation.

0.7.4 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. SpyREST is composed of three different components as follows:

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 3, 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.

For every API example, the Spy automatically extracts and saves the API version, RESTful resource, API action, HTTP query parameters, request and response headers, and request and response bodies. The Spy recognizes the commonly used standards for RESTful APIs to auto-extract the aforementioned properties for each example. While executing the example API calls, the RESTful API developers can use several SpyREST specific HTTP headers to provide custom descriptions for the API examples, and override autodetected versions, resources and API actions.

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 to only show sample items instead of the whole array. As

a result, the Spy automates the repeated parts of the RESTful API documentation process using intelligent defaults. For evolving APIs, a rerun of the same API examples will update the auto-generated documentation. The Spy component provides satisfies the following requirements: automated, example based, version-aware, customizable, and a reusable solution for RESTful API documentation.

The Web App

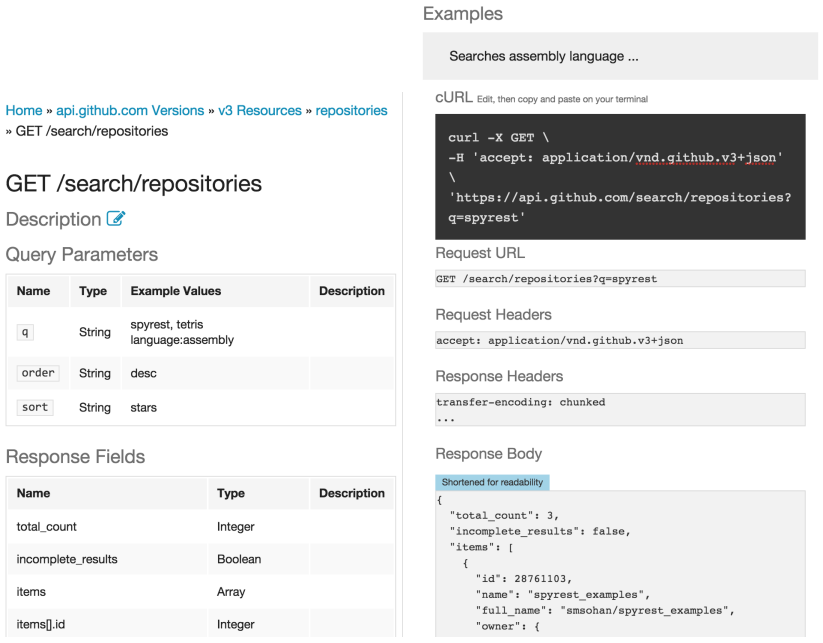
The web app complements the features of the Spy to fulfill the rest of the identified must-have requirements for RESTful API documentation. The web app displays RESTful API documentation based on the data recorded by the Spy and implements the steps S5, S6, and S7 as shown in Figure 3. Additionally, it 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 command that the API client developers can use to exercise the API without writing any client code. cURL is a general purpose utility tool that can be used as a RESTful API client to make API calls.

To facilitate collaboration, the web app provides in-page commenting throughout the site. Developers can discuss and learn from past discussions on API related matters on the API documentation site where the documentation of specific API elements and related collaboration can co-exist. The web app component implements solutions for the following requirements: executable examples, customizable, reusable, and collaborative documentation for RESTful APIs. Figure 4 shows a screenshot of the web app. The contents and presentation on Figure 4 has been adapted for brevity.

Database

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

In addition to satisfying the must-have requirements, SpyREST provides several other advantages. For example, SpyREST can be used to document RESTful APIs irrespective of the technology that is used to implement the APIs, since a HTTP proxy server is used and access to source code of the API is not required. SpyREST provides reusability as a tool, and also as a



(a) API documentation summary section (b) API documentation examples section

Figure 4: SpyREST Screen shots showing auto generated API documentation

platform by providing cloud based software as a service solution for RESTful APIs where the documentation for multiple RESTful APIs can be generated and published on a shared platform.

I have performed a preliminary evaluation of SpyREST and found it produced 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. In 5 of the 25 API actions, SpyREST documented API response fields that were not included the official API documentations, resulting into more accurate documentation of the APIs. In the remainder of my research, I plan to conduct additional evaluations of SpyREST involving RESTful API developers to better understand the effectiveness of the HTTP Proxy server based approach of automated RESTful API documentation.

0.8 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 generate and maintain usable documentation for their RESTful APIs in a cost-effective manner.

0.9 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 preliminary 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
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

Semester	Tasks
Winter 2016	Extending the Systematic Literature Review on Web API Evolution Performing an evaluation of SpyREST Planned Publications: ICSME 2016 Systematic Literature review on Web API evolution.
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 2: Milestones

Bibliography

[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 *Software Engineering (ICSE), 2012 34th International Conference on*, 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.

[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.

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

[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 *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 *Reverse Engineering (WCRE), 2013 20th Working Conference on*, 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 *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 '11, 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.

[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.

[Stepalina, 2010] Stepalina, E. (2010). SaaS support in software documentation systems. In *Software Engineering Conference (CEE-SECR), 2010 6th Central and Eastern European*, pages 192–197.

[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 *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, 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 *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, 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 *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 151–155. IEEE.