Assisting with API Design through Reusing Design Knowledge

Mahsa Sadi
Department of Computer Science
University of Toronto

October 31\textsuperscript{th}, 2019

Motivation and Background Context

– A recent trend towards opening up software products to 3\textsuperscript{rd}-party applications and services

– Developing Application Programming Interfaces (APIs) has become an increasingly common practice

The Real-World Problem

– APIs expose critical data and back-end services towards their clients

– Concerns about critical non-functional requirements:
  – the security of the back-end systems
  – the confidentiality of the exchanged data
  – the performance of the provided services

Introduction

Research Question and Gap

Research Question:

- “How to address non-functional requirements in APIs?”

Research Gap:

- There is still no framework to help software developers with the above question.

Thesis Objective and Approach

Objective:

- Devising a framework that can reliably aids software developers in addressing non-functional requirements in APIs

Approach:

- Reusing API Design Knowledge

Thesis Overview

Research Step 1:

Collecting and Organizing API Design Knowledge
Objectives and Method

– Collecting and organizing the API design knowledge from various dispersed resources:
  – Expert Opinion: Books, vendor white papers, weblogs
  – Available standards and design frameworks
  – Peer-reviewed Literature

– A systematic and evidence-based review of the literature


API Non-Functional Requirements
– An Example

– Security of an API is the degree to which an API is free from external threats and attacks, internal errors and failures, and unintended access.


Outcomes and Contributions

– A structured body of API design knowledge:
  1. API non-functional requirements
  2. API design techniques
  3. The trade-offs of the API design techniques

API Design Techniques – An Example

– API access authorization mechanisms are responsible for permitting a client to access an API.

RFC 4158: Internet X.509 Public Key Infrastructure: Certification Path Building, Available at [https://tools.ietf.org/html/rfc4158], Retrieved on 21/07/2018
API Design Trade-Offs – An Example

API-KeY trade-offs:

- **API Usability - Usage Simplicity**: (+) (Strong). An API can be simply used by presenting a key to the API. There are low security barriers in order to use an API.

- **Support for the evaluation**: Qualitative reasoning and expert opinion

The table below shows the trade-offs for different aspects of API usage and security:

<table>
<thead>
<tr>
<th></th>
<th>Access Simplicity</th>
<th>Usage Simplicity</th>
<th>Latency</th>
<th>Access Confidentiality</th>
<th>Message Confidentiality</th>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>API-Key</td>
<td>+ Strong</td>
<td>+ Strong</td>
<td>+ Strong</td>
<td>+ Weak</td>
<td>- Strong</td>
<td>- Strong</td>
</tr>
</tbody>
</table>


Objectives and Method

**Objective**: Encoding the API Design Knowledge

**Method**: Describing the knowledge in the Non-Functional Requirements (NFR) multi-valued logic


Research Step 2:

Formalizing and Encoding the Collected API Design Knowledge

Outcomes

156 API Design Catalogues:

\[(G_1, \ldots, G_n) \xrightarrow{\text{Rule Type}} G_m : \text{Rule Category}\]

- \(G_i\) is a term in the form of \(\text{Type [Topic]}\)
- \(\text{Rule Type} \in \{\text{Break, SomeMinus, Hurt, Unknown, Help, SomePlus, Make}\}\)
- \(\text{Rule Category} \in \{\text{NF-REF, NF-OP, F-REF, F-OP, COR}\}\)
Research Step 3: Using the Encoded API Design Knowledge

---

1. (Access Simplicity [API], Access Duration [API], Access Rate [API]) \(\rightarrow\) Accessibility [API] : NF-REF

2. (Compatibility with Minor Changes [API], Compatibility with Major Changes [API]) \(\rightarrow\) Evolvability [API] : NF-REF

---

155. (Client-Side Two-Phase Transaction Management [ ]) \(\rightarrow\) Latency [API] : COR

156. (Client-Side Two-Phase Transaction Management [ ]) \(\rightarrow\) Throughput [API] : COR
Objectives and Method

- Developing a method to systematically use the encoded API design knowledge:
  1. A step-wise refinement procedure
  2. An evaluation procedure
     - Using the NFR forward evaluation procedure
  3. A selection procedure


Outcomes and Contributions

- A semi-formal methodology for designing requirements into APIs

Input:
- Design a mechanism to secure access to the Account API.
- Confidentiality of the Account API is Very Critical.
- Privacy of the Account API is Very Critical.
- Latency of the Account API is Critical.

Output:
- Either Username and Password, or
  Open Authorization version 2.0

Component 1: Refinement Procedure - 1

“Design a mechanism to secure access to the Account API.”

Component 1: Step-Wise Refinement Procedure - 2

1- Security [API]

2- Security [API] and
(Confidentiality [API], Privacy [API], Operational Security [API], Reliability [API])

...and...

10- Access Authorization [API]

(API-Key [ ], Username and Password [ ], Mutual Certificate-Based Authentication X.509 [ ], Open Authorization Version 2.0 [ ], Open-ID Connect Version 1.0 [ ])
Component 2: Evaluation Procedure

Component 3: Selection Procedure

Research Step 4:

Tool Support for Using the Encoded API Design Knowledge

Objectives and Method

– Developing a tool that supports the use of the API catalogues

– Designed and implemented a rule-based knowledge-based system in Java
### Method – Development of the Tool

<table>
<thead>
<tr>
<th>Design Step</th>
<th>Related Rule</th>
<th>English Translation</th>
</tr>
</thead>
</table>
| Requirement Refinement | $G_i \rightarrow \text{Refine} \rightarrow G_j$ | “Elaborate on the requirement $G_i$. The requirement $G_i$ can be refined into the requirement $G_j$."
| Requirement Refinement | $G_i \rightarrow \text{and} \rightarrow \{G_p, \ldots, G_n\}$ | “Elaborate on the requirement $G_i$. The requirement $G_i$ can be refined into the following requirements: $G_p, \ldots, G_n$."

### Outcomes

---

**–RAPID an Interactive design assistant**

**Source Code:** [https://github.com/m-h-s/RAPID](https://github.com/m-h-s/RAPID)

---

**Research Step 5:**

**Evaluating the developed Framework**
Objectives and Method

Research Question:

“How valid and reliable are the design guidelines of the framework?”

Method:

a) Seating the tool in an API design exam

b) Asking 7 experienced developers to blindly evaluate the accuracy of the provided answers

Objectives and Method (2)

Research Question:

“Why some answers have been considered as unacceptable by some of the evaluators?”

Method:

a) Open Coding:

Categorizing the comments of the evaluators

---

Measuring the Validity of the Design Guidelines

Accuracy Measure:

\[
\frac{\# \text{ Acceptable Answers}}{\# \text{ Answers} (= 30)}
\]

- An acceptable answer:
  - is accepted by the majority of the evaluators
  - \# Evaluators = 7 \Rightarrow \text{Majority} : n > 3

How valid are the design guidelines?

<table>
<thead>
<tr>
<th>Evaluator</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>☹ (%)</td>
<td>53.3</td>
<td>50.0</td>
<td>46.7</td>
<td>43.3</td>
<td>76.7</td>
<td>73.3</td>
<td>96.7</td>
</tr>
<tr>
<td>☹ (%)</td>
<td>36.7</td>
<td>30.0</td>
<td>53.3</td>
<td>40.0</td>
<td>13.3</td>
<td>26.7</td>
<td>3.3</td>
</tr>
</tbody>
</table>

\[ \bar{x} = 62.9 \% \quad \sigma = 19.76 \% \]

Accuracy = \[\frac{\# \text{ Acceptable Answers} (= 22)}{\# \text{ Answers} (= 30)} = 73.3\% \]
Why are some answers unacceptable?

Summary and Conclusions

Summary – Motivation and Objectives

Problem:

– Addressing non-functional requirements in APIs is crucial considering the trade-offs to be made

Objective:

– Devising a framework that assists software engineers with addressing these requirements

Summary - Method

1. Collecting and organizing API design knowledge
2. Formalizing API design knowledge
3. Using the encoded API design knowledge
4. Developing a tool that supports the use of the encoded design knowledge
5. Evaluating the reliability of the provided design assistance
Summary – Research Questions (1)

– RQ 1. What non-functional requirements should be considered in designing APIs?
– RQ 2. What techniques are suggested to address these requirements in APIs?
– RQ 3. What are the trade-offs of these techniques?
– RQ 4. How to represent and formalize design knowledge?

Summary – Research Questions (2)

– RQ 5. How to design a tool that can process design knowledge?
– RQ 6. How to evaluate a framework that assists with the task of software design?

Conclusions – Thesis Statement

It is possible to devise an assistant that can reliably assist software developers with addressing non-functional requirements in APIs.

Future Work

– Evaluating the usefulness and effectiveness of RAPID in assisting software developers with API design
E-mail: mhsadi@cs.toronto.edu