Assisting with API Design

through Reusing Design Knowledge

Mahsa Sadi

Department of Computer Science

University of Toronto

October 31th, 2019

Introduction

Motivation and Background Context

- A recent trend towards opening up software products to 3rd-party applications and services
- Developing Application Programming Interfaces (APIs) has become an increasingly common practice

- **Bosch, J. (2016).** Speed, data, and ecosystems: the future of software engineering. IEEE Software, 33(1), 82-88.

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

- **Bosch, J. (2010)**. Architecture challenges for software ecosystems. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (pp. 93-95).
- **Scacchi, W., & Alspaugh, T. A. (2013).** Processes in securing open architecture software systems. In Proceedings of International Conference on Software and System Process.

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.

5

Thesis Objective and Approach

Objective:

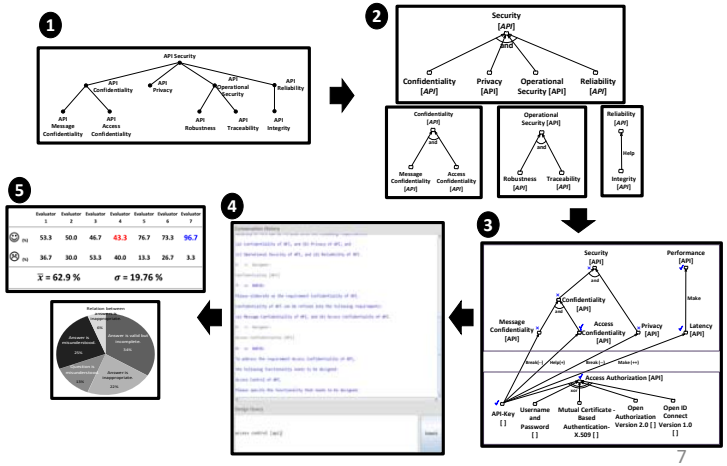
– Devising a framework that can reliably aids software developers in addressing non-functional requirements in APIs

Approach:

– Reusing API Design Knowledge

6

Thesis Overview



7

Research Step 1:

Collecting and Organizing API Design Knowledge

8

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

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele, UK, Keele University, 33(2004), 1-26.

Dyba, T., Kitchenham, B. A., & Jorgensen, M. (2005). Evidence-based software engineering for practitioners. IEEE software, 22(1), 58-65.

9

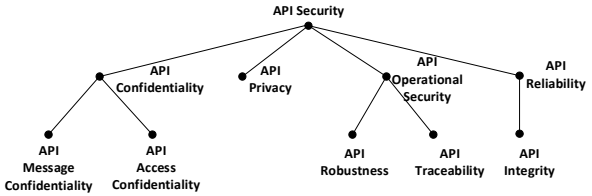
Outcomes and Contributions

- **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

10

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.



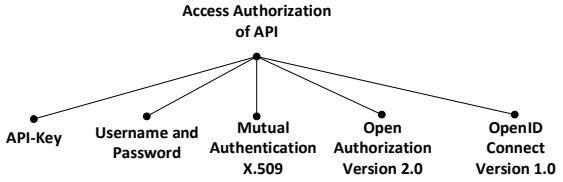
Siriwardena, P. (2014). Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE. Apress, Berkeley, CA.

De, B. (2017). API Management: An Architect's Guide to Developing and Managing APIs for Your Organization. Apress, Berkeley, CA, First edition March 2017.

11

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

RFC 6749 - The OAuth 2.0 Authorization Framework, Available at https://tools.ietf.org/html/rfc6749, Retrieved on 17 / 06 / 2018

Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., & Mortimore, C. (2014). OpenID Connect Core 1.0 incorporating errata set 1. The OpenID Foundation, specification.

12

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*

	Access Simplicity	Usage Simplicity	Latency	Access Confidentiality	Message Confidentiality	Privacy
API-Key	+ Strong	+Strong	+ Strong	+Weak	- Strong	- Strong

De, B. (2017). API Management: An Architect's Guide to Developing and Managing APIs for Your Organization. Apress, Berkeley, CA, First edition March 2017.

Research Step 2:

Formalizing and Encoding the Collected API Design Knowledge

Objectives and Method

- **Objective:** Encoding the API Design Knowledge
- **Method:** Describing the knowledge in the Non-Functional Requirements (NFR) multi-valued logic

Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2000). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media.

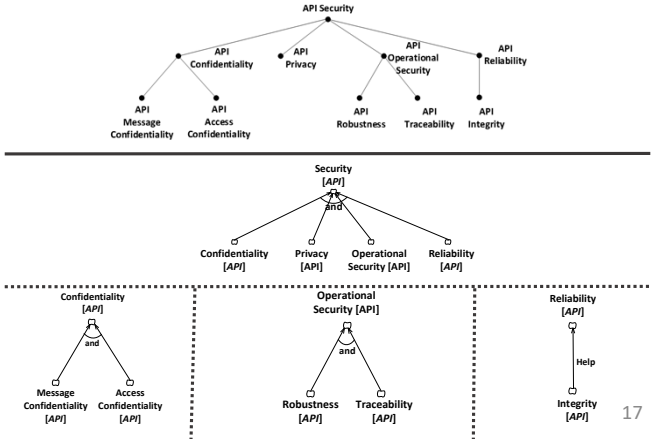
Outcomes

156 API Design Catalogues:

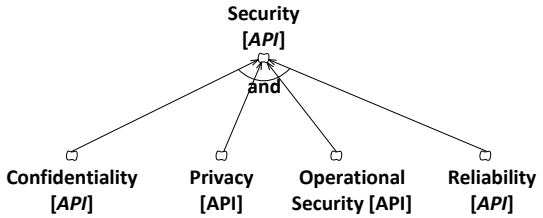
$$(G_1,\ldots, G_n) \xrightarrow{Rule\ Type} G_m : Rule\ Category$$

- G_i is a term in the form of ***Type [Topic]***
- ***Rule Type*** ∈ {Break, SomeMinus, Hurt, Unknown, Help, SomePlus, Make}
- ***Rule Category*** ∈ {NF-REF, NF-OP, F-REF, F-OP, COR}

API Design Catalogues – Example



API Design Catalogues – Example



(Confidentiality [API], Privacy [API], Operational Security [API], Reliability [API]) $\xrightarrow{\text{and}}$ Security [API] : NF-REF

API Design Catalogues

1- (Access Simplicity [API], Access Duration [API], Access Rate [API]) $\xrightarrow{and}$ Accessibility [API] : NF-REF

2- (Compatibility with Minor Changes [API], Compatibility with Major Changes [API]) $\xrightarrow{and}$ Evolvability [API] : NF-REF

……..

……..

155- (Client-Side Two-Phase Transaction Management [ ]) $\xrightarrow{Break--}$ Latency [API] : COR

156 - (Client-Side Two-Phase Transaction Management [ ]) $\xrightarrow{Break--}$ Throughput [API] : COR

Research Step 3:

Using the Encoded API Design Knowledge

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

Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2000). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media.

21

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:

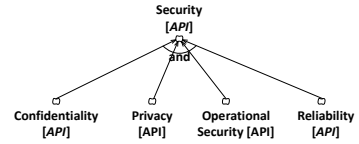
Either
- Username and Password,
or
- Open Authorization version 2.0

22

Component 1: Refinement Procedure - 1

"Design a mechanism to **secure** access to the Account **API**."

1- Security [API]



2- Security [API] $\xleftarrow{and}$

(Confidentiality [API], Privacy [API], Operational Security [API], Reliability [API])

23

Component 1: Step-Wise Refinement Procedure - 2

1- Security [API]

...

2- Security [API] $\xleftarrow{and}$

(Confidentiality [API], Privacy [API], Operational Security [API], Reliability [API])

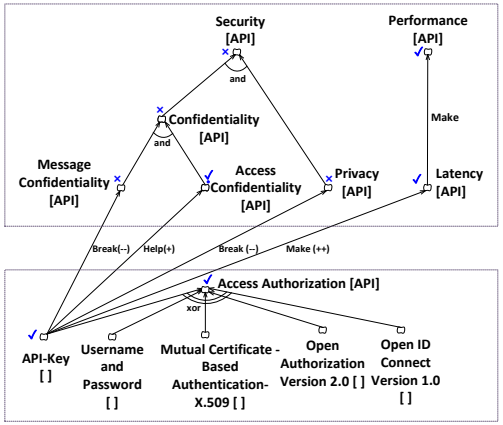
...

10- Access Authorization [API] $\xleftarrow{xor}$

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

24

Component 2 : Evaluation Procedure



25

Component 3 : Selection Procedure

"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**."

Requirements Specification	**Requirement**	**Confidentiality [API]**	**Privacy [API]**	**Latency [API]**	**Score**
	Priority	**High**	**High**	**Medium**	
	Expected Satisfaction Value	**Sat**	**Sat**	**Sat**	**20**
Requirements Satisfaction in the Access Authorization Design Mechanisms Available in the Catalogues	**API-Key**	Den	Den	Sat	-12
	Username and Password	**PSat**	**PSat**	**PDen**	**6**
	Mutual Authentication	Sat	Den	PDen	-2
	OAuth 2.0	**PSat**	**PSat**	**PDen**	**6**
	OpenID connect	PSat	PSat	Den	4

Research Step 4:

Tool Support for Using the Encoded API Design Knowledge

27

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

28

Method – Development of the Tool

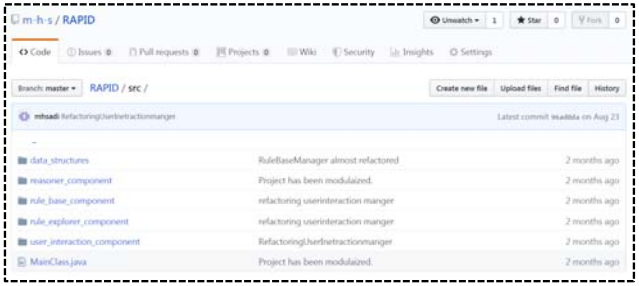
Design Step		Related Rule Category	English Translation
Step	**Step Form**		
Requirement Refinement	$G_i \xleftarrow{Help} G_j$	NF-REF	"Elaborate on the requirement G_i. The requirement G_i can be refined into the requirement G_j."
	$G_i \xleftarrow{and} (G_j, \dots, G_n)$	NF-REF	"Elaborate on the requirement G_i. The requirement G_i can be refined into the following requirements: G_j, and … , and G_n."

29

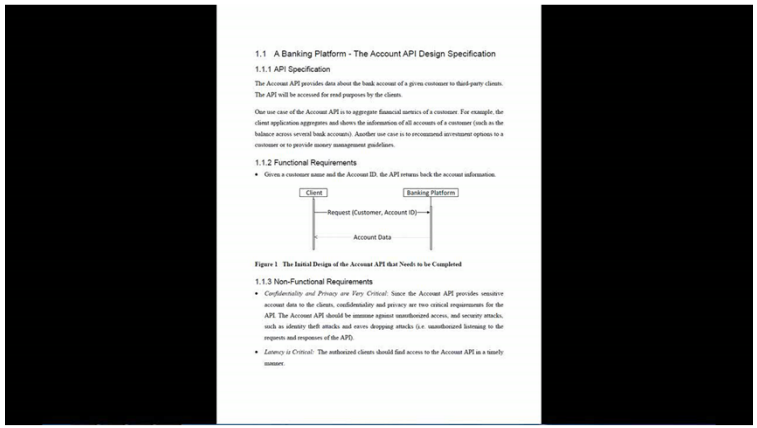
Outcomes

–RAPID an Interactive design assistant

Source Code: https://github.com/m-h-s/RAPID



30



1.1 A Banking Platform - The Account API Design Specification

1.1.1 API Specification

The Account API provides data about the bank account of a given customer to third-party clients. The API will be accessed for read purposes by the clients.

One use case of the Account API is to aggregate financial metrics of a customer. For example, the client application aggregates and shows the information of all accounts of a customer (such as the balance across several bank accounts). Another use case is to recommend investment options to a customer or to provide money management guidelines.

1.1.2 Functional Requirements

- Given a customer name and the Account ID, the API returns back the account information.

Figure 1 The Initial Design of the Account API that Needs to be Completed

1.1.3 Non-Functional Requirements

- *Confidentiality and Privacy are Very Critical:* Since the Account API provides sensitive account data to the clients, confidentiality and privacy are two critical requirements for the API. The Account API should be immune against unauthorized access, and security attacks, such as identity theft attacks and eaves dropping attacks (i.e. unauthorized listening to the requests and responses of the API).
- *Latency is Critical:* The authorized clients should find access to the Account API in a timely manner.

31

Research Step 5:

Evaluating the developed Framework

32

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

33

Measuring the Validity of the Design Guidelines

Accuracy Measure:

$$\frac{\#\ Acceptable\ Answers}{\#\ Answers\ \ (=30)}$$

– An acceptable answer:

– is accepted by the majority of the evaluators

– # Evaluators = 7 ⇒ Majority : n > 3

34

How valid are the design guidelines?

	Evaluator 1	Evaluator 2	Evaluator 3	Evaluator 4	Evaluator 5	Evaluator 6	Evaluator 7
☺ (%)	53.3	50.0	46.7	43.3	76.7	73.3	96.7
☹ (%)	36.7	30.0	53.3	40.0	13.3	26.7	3.3

$\bar{x}$ = 62.9 % σ = 19.76 %

Accuracy = # Acceptable Answers (= 22) / # Answers (= 30) = 73.3%

Objectives and Method (2)

Research Question:

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

Method:

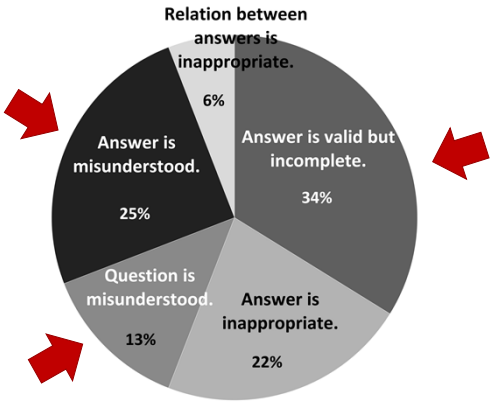
– Open Coding:

Categorizing the comments of the evaluators

- Flick, U. (2009). An introduction to qualitative research. Sage Publications Limited.

36

Why are some answers unacceptable?



37

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

39

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

40

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?
- **RQ4.** How to represent and formalize design knowledge?

41

Summary – Research Questions (2)

- **RQ 5**. How to design a tool that can process design knowledge?
- **RQ6**. How to evaluate a framework that assists with the task of software deign?

42

Conclusions – Thesis Statement

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

43

Future Work

- Evaluating the usefulness and effectiveness of RAPID in assisting software developers with API design

44

?

E-mail: mhsadi@cs.toronto.edu