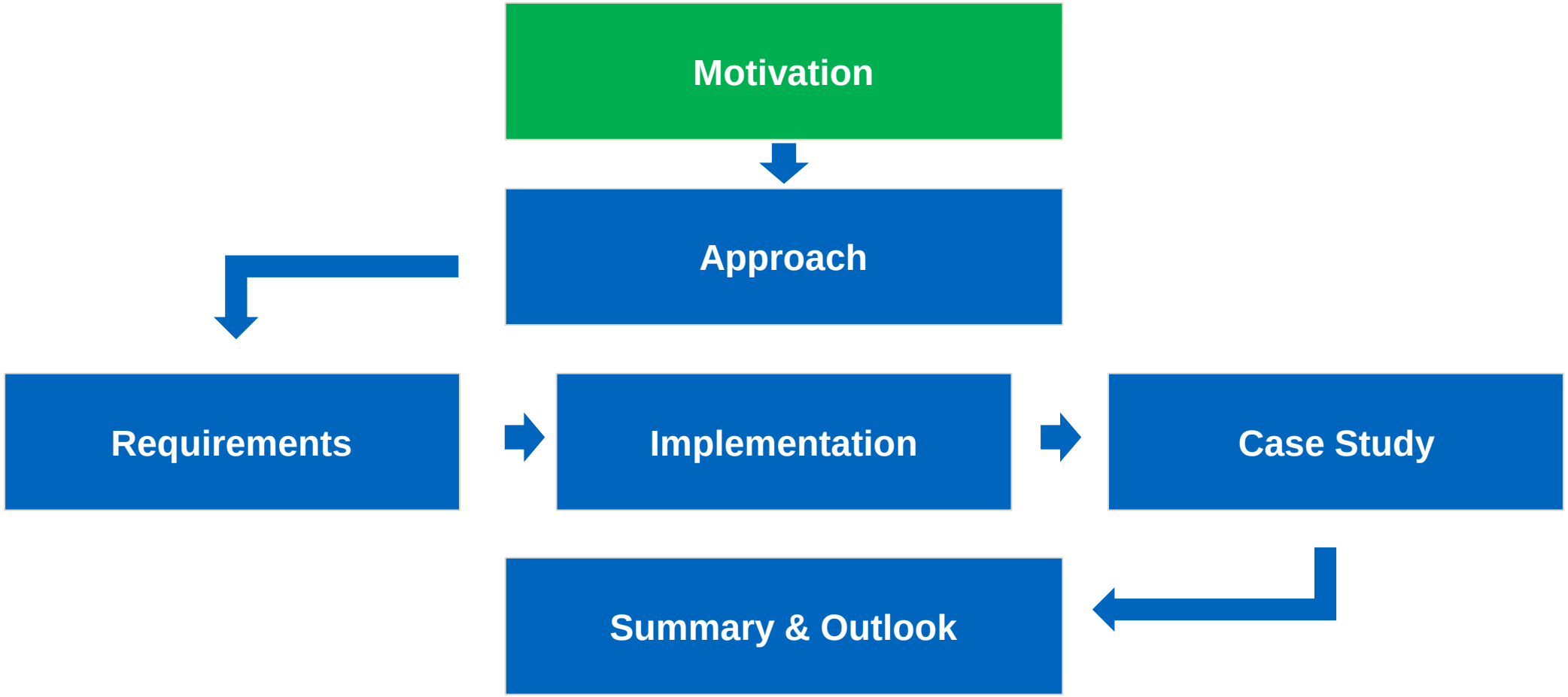
Improving the Developer Experience of API Consumers using Usage Scenarios and Examples

Master's Thesis – Final Presentation

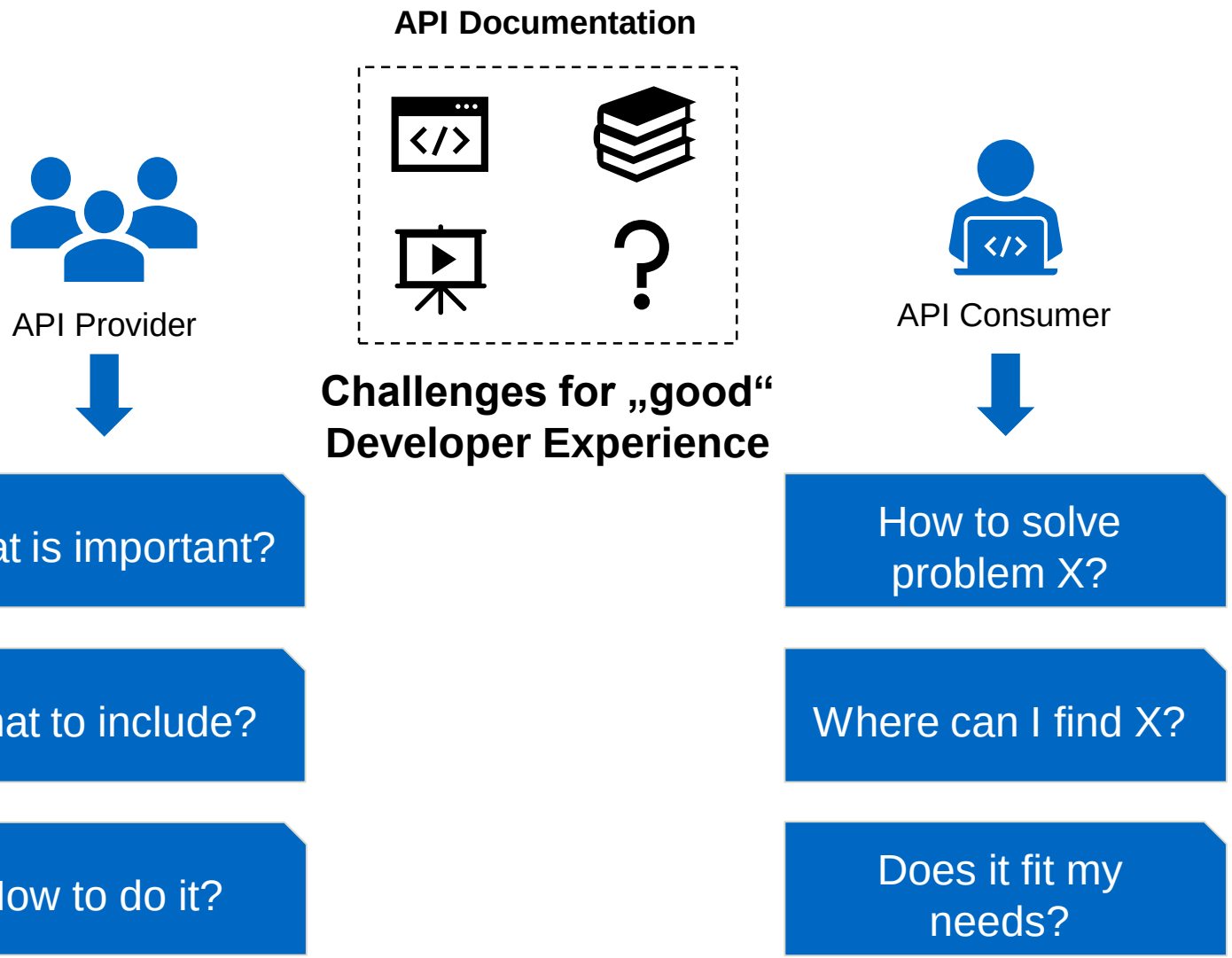
Arif Cerit | 19.08.19 | Garching

Chair of Software Engineering for Business Information Systems (sebis)
Faculty of Informatics
Technische Universität München
wwwmatthes.in.tum.de

Outline


Motivation
Approach
Requirements
Implementation
Case Study
Summary & Outlook

Motivation



API providers **do not know** their API consumers

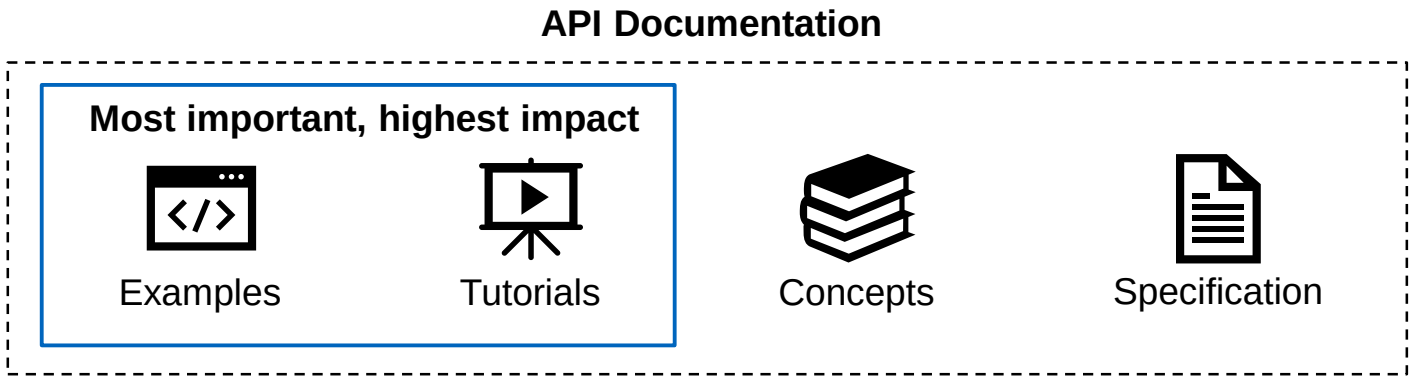
Information asymmetry
&
Unkown needs

Working with APIs is **common** but **hard**

Developer experience
&
Right information
Right format

Fagerholm & Münch (2012) | Meng et al. (2019) | Robillard (2009) | Uddin & Robillard (2015)

Motivation



Previous work

How is API documentation used?

Are examples **effective**?

Technical recommendations (headers, types, error codes …)

Lack of research

Requirements coming from **“both sides”**

What is the contribution of **individual decisions**?

Practical evaluations with **observable** impact and benefits

Goals

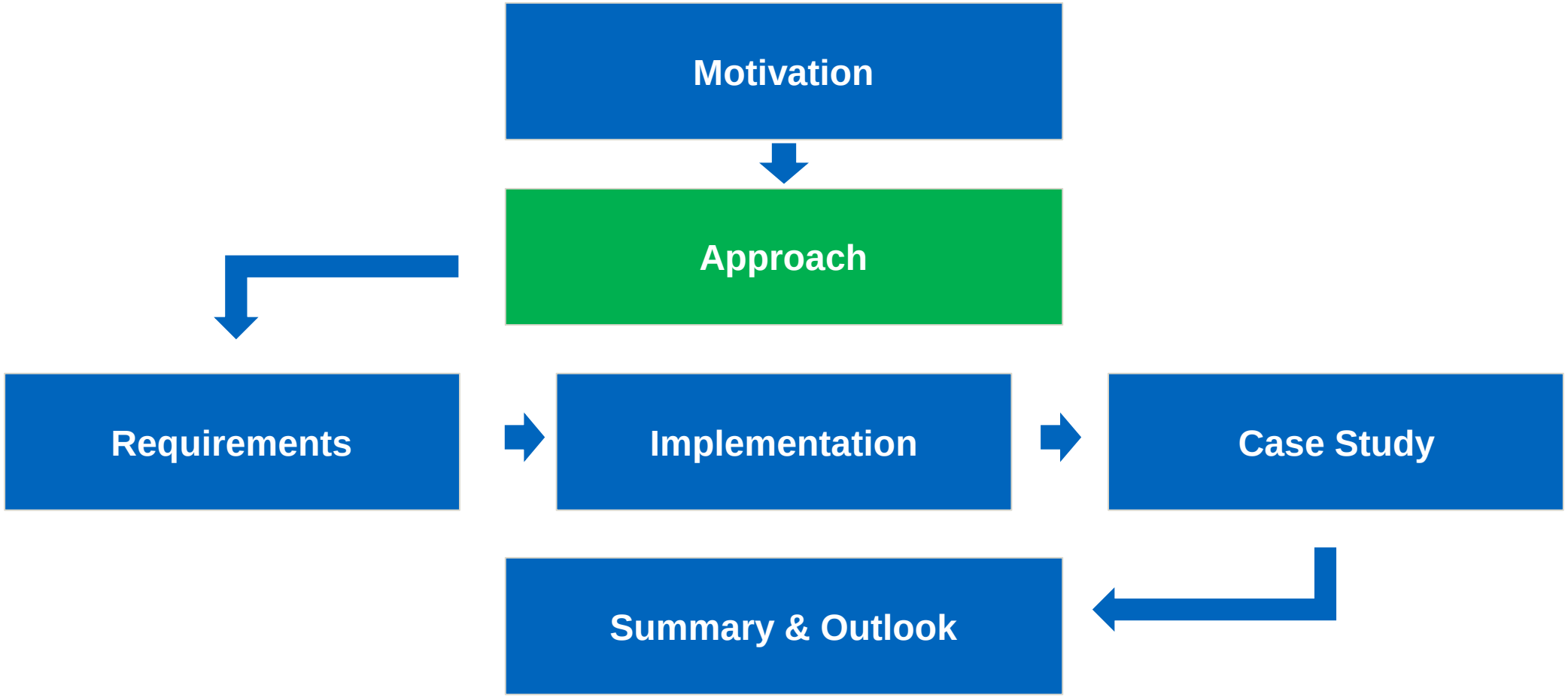
Elicit **requirements** with experts from **both sides**

Find **improvements** for developer experience

Study API consumers in a **natural setting** with **multiple data sources**

Robillard (2009) | Sohan et al. (2017) | Zhang et al. (2019)

Outline


Motivation
Approach
Requirements
Implementation
Case Study
Summary & Outlook

Approach

Research Questions

RQ 1 What are the **approaches** and **concepts** to create and publish API usage scenarios and examples?

Literature + Interviews

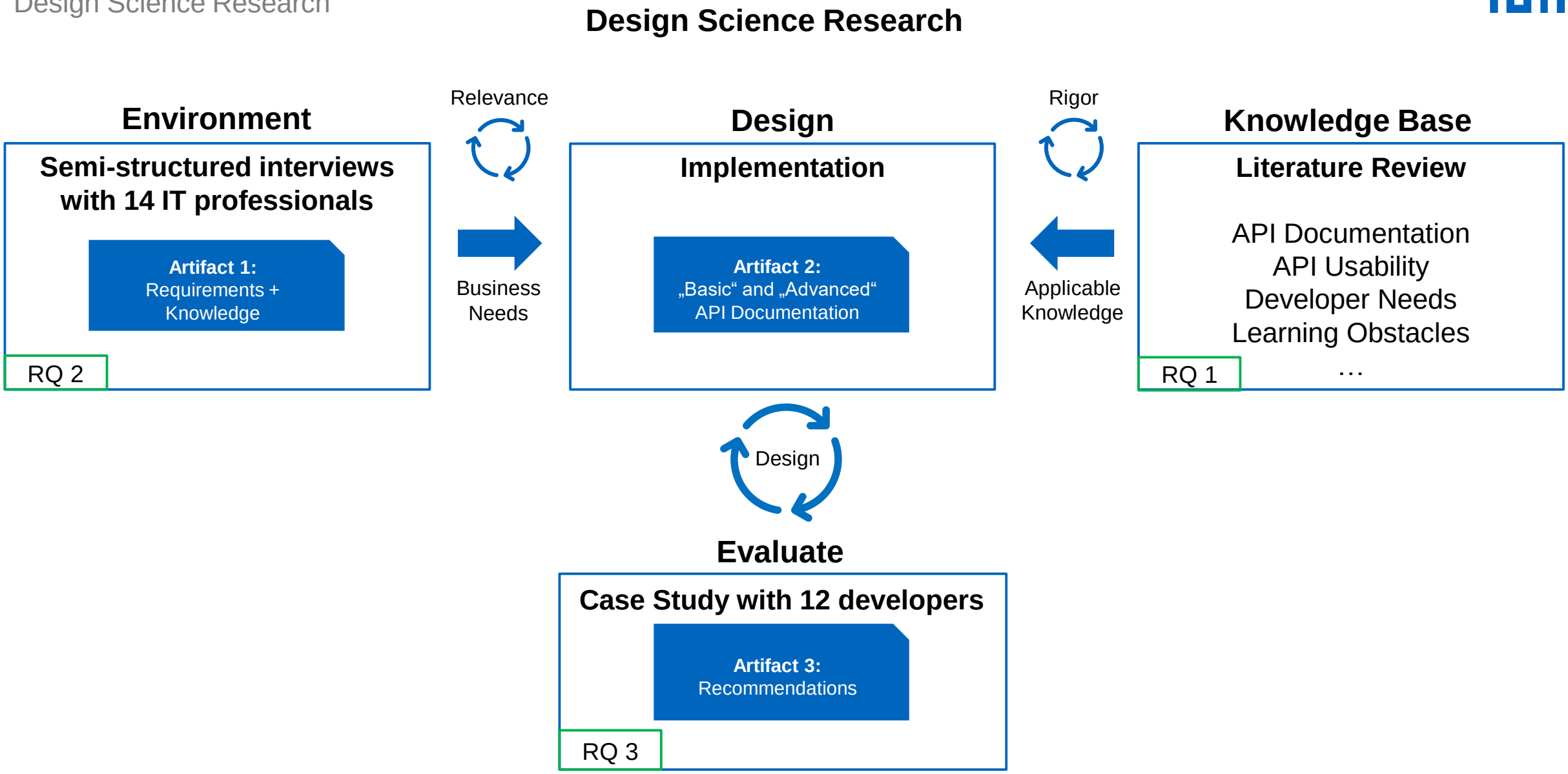
RQ 2 What are the **requirements and knowledge types** that must be included in usage scenarios and examples?

Literature + Interviews

RQ 3 How can usage scenarios and examples be leveraged to **improve the developer experience** for API consumers?

Implementation + Case Study

Approach
Design Science Research



Hevner et al. (2004) | Lethbridge et al. (2005) | Peffers et al. (2007) | Runeson & Höst (2007)

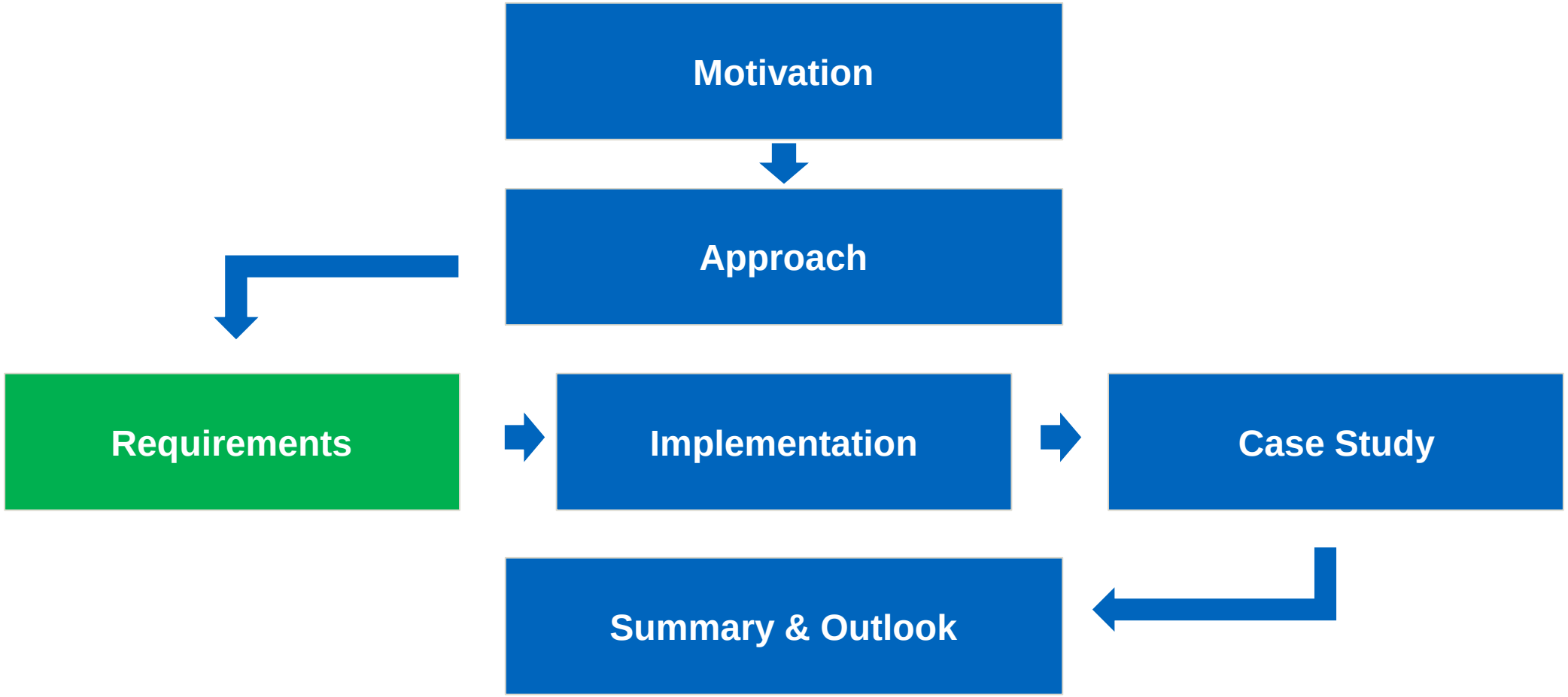
Approach

Industry Partner

- **SAP Customer Experience – Cloud Business Group**
- **Locations:** Walldorf, Munich, Gliwice (Poland)

- **Why?**
  - Large **API projects** with many software developers
  - Awareness for **developer experience**
  - Access to teams of API **providers and consumers**

Outline


Motivation
Approach
Requirements
Implementation
Case Study
Summary & Outlook

Requirements

Semi-structured Interviews

- **14 IT professionals**
  - Roles: development, architecture
  - Experiences with APIs

- **Questions**
  - Challenges
  - Requirements
  - Knowledge needs

- **Integrated coding approach**
  - Deductive → knowledge type taxonomy
  - Inductive → emerging patterns

ID	Role	Experience [years]	Duration [hh:mm]
A1*	Software Architect	12	01:15
A2	Product Owner	15	00:32
A3	Enterprise Architect	12	-
A4	Enterprise Architect	10	00:30
A5	Software Architect	29	00:36
A5	Enterprise Architect	7	00:44
A7	Product Owner	12	00:45
D1*	Lead Developer	20	01:05
D2*	Senior Developer	15	01:31
D3	Senior Developer	6	00:40
D4	Developer	4	00:50
D5	Developer	2	00:36
D6	Senior Developer	8	00:41
D7	Senior Developer	20	00:38
	Mean	**12,3**	**00:39**

** Participant was interviewed twice (prestudy)*
Cruzes & Dyba (2011) | Maalej & Robillard (2013)

Requirements

Functional & Non-functional

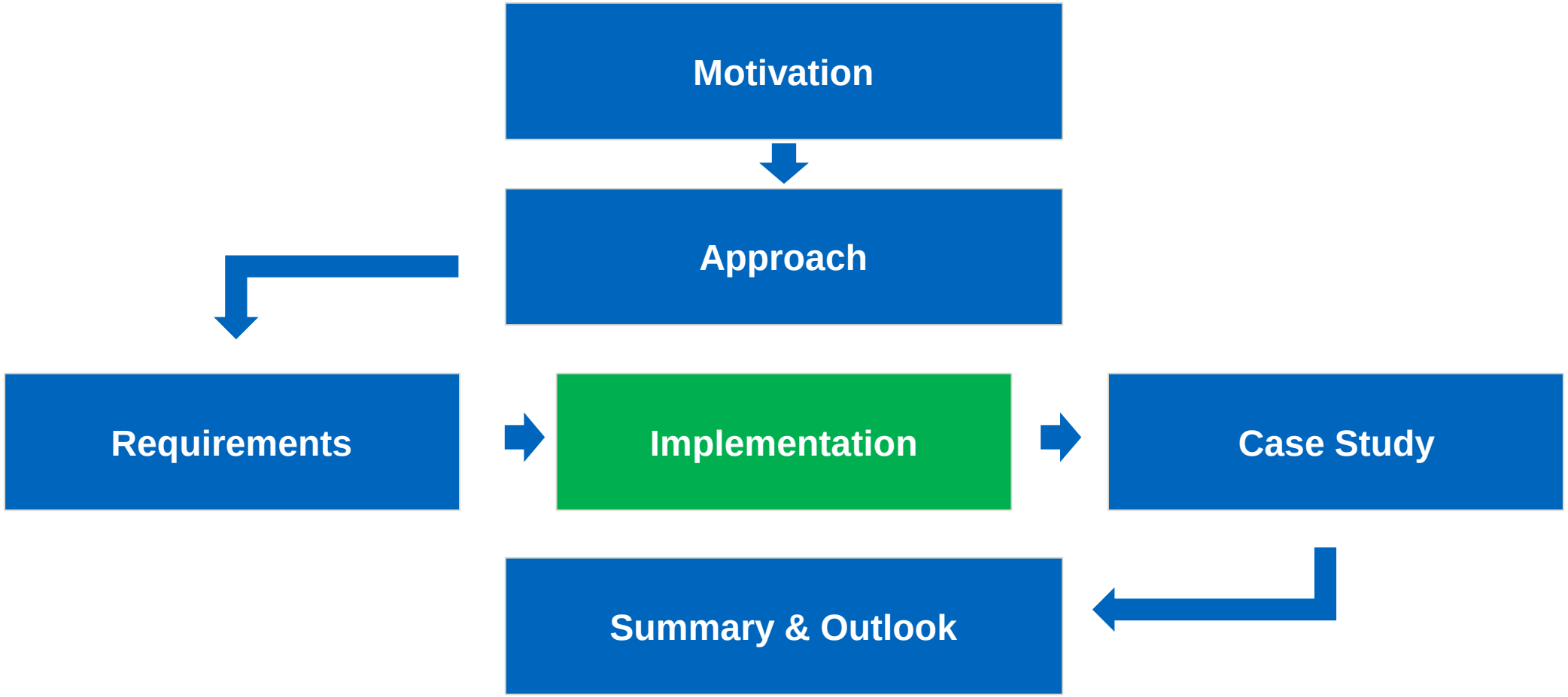
Functional	Non-functional
F1 – option to execute	**NF1** – copy-friendliness
F2 – option to adapt	**NF2** – accessibility
F3 – labelling and versioning	**NF3** – „less is more“ for descriptions
F4 – „instant“ feedback	**NF4** – cohesive „pieces of examples“
F5 – low-level documentation links	**NF5** – coverage of cases end-to-end
F6 – „main“ scenario as entry point	**NF6** – coverage of most important cases
F7 – accompanying textual description	**NF7** – increasing complexity
F8 – tool support for executability	

Requirements

Conceptual & Structural

Observation	Implication
Provider‘s Best Practices	Show and explain in examples
Intended/Unintended Use	Tutorials describe intended but also not intended use
API Capabilities	Concise list only dedicated to API capabilities
Junctions in Tutorials	Show where a path divides into alternatives
Interaction Order	Order of interaction needs explanation (if complex)
Tutorial Story	Explicit focused story to define context and outcomes
…	

Outline


Motivation
Approach
Requirements
Implementation
Case Study
Summary & Outlook

Implementation
Features

Selection criteria

- **Implemented** features:
  - Relevant
  - Potentially useful
  - Necessary

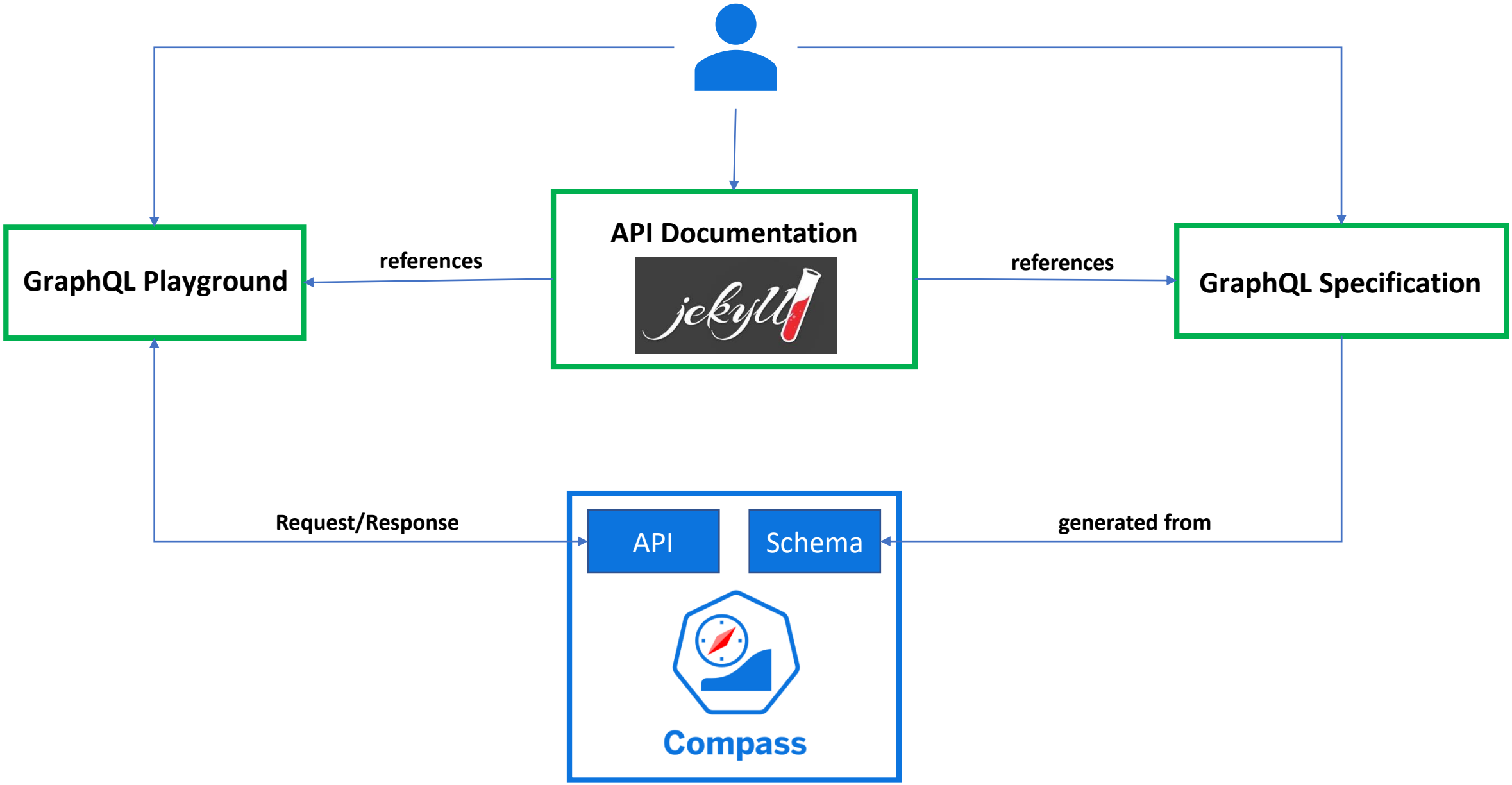
- **„Advanced“** features
  - Rare in previous research
  - Contradicting with previous research
  - Controversial

Feature	Basic	Advanced
F1 – executable	X	X
F2 – adaptable	X	X
F5 – references to low-level		X
F6 – „main scenario“		X
F7 – text descriptions	X	X
F8 – familiar tools		X
NF1 – copy-friendly	X	X
NF2 – accessibility	X	X
NF3 – „less is more“		X
NF6 – main use case coverage		X
NF7 – increasing complexity		X

Knowledge Type	Basic	Advanced
Misunderstanding Terminology	X	X
Best Practices		X
API Capabilities	X	X
Junctions in Paths		X
Stories in Tutorials		X

Implementation

Structure

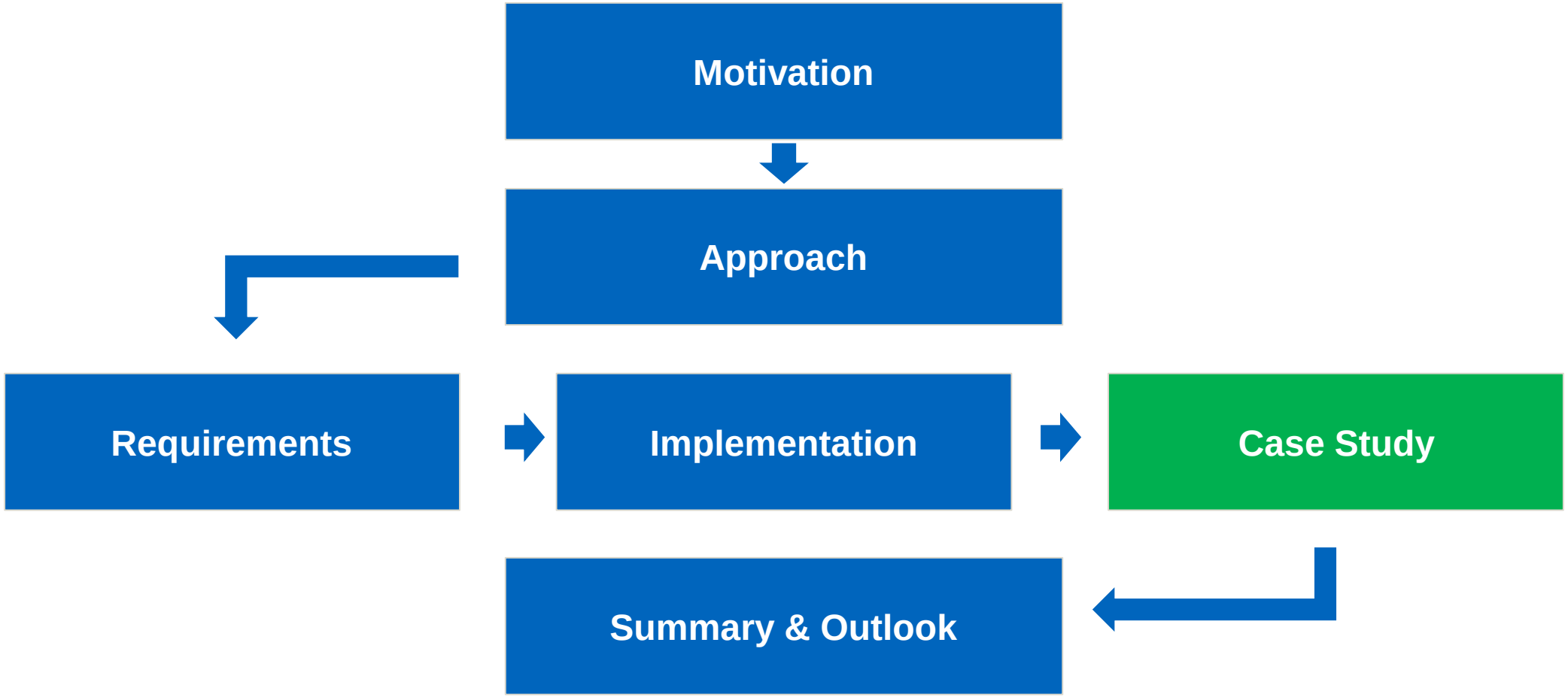


API Documentation Demo

Implementation
Structure

Section	Basic	Advanced
Overview	Concepts Components Flows	No additions
Getting Started	Playground Specification	No additions
Tutorial	First steps Simple scenarios	**+ Story** **+** Increased **complexity** **+ Tool support** **+ Concise** descriptions
Samples	Queries Basic I/O behavior	**+ Complex** requests **+ References** **+** Increased **coverage**
Best Practices		GraphQL **hints** Working with **Playground**
Glossary	All entities	No additions

Outline


Motivation
Approach
Requirements
Implementation
Case Study
Summary & Outlook

Case Study
Design

Participants

„Advanced“ documentation

„Basic“ documentation

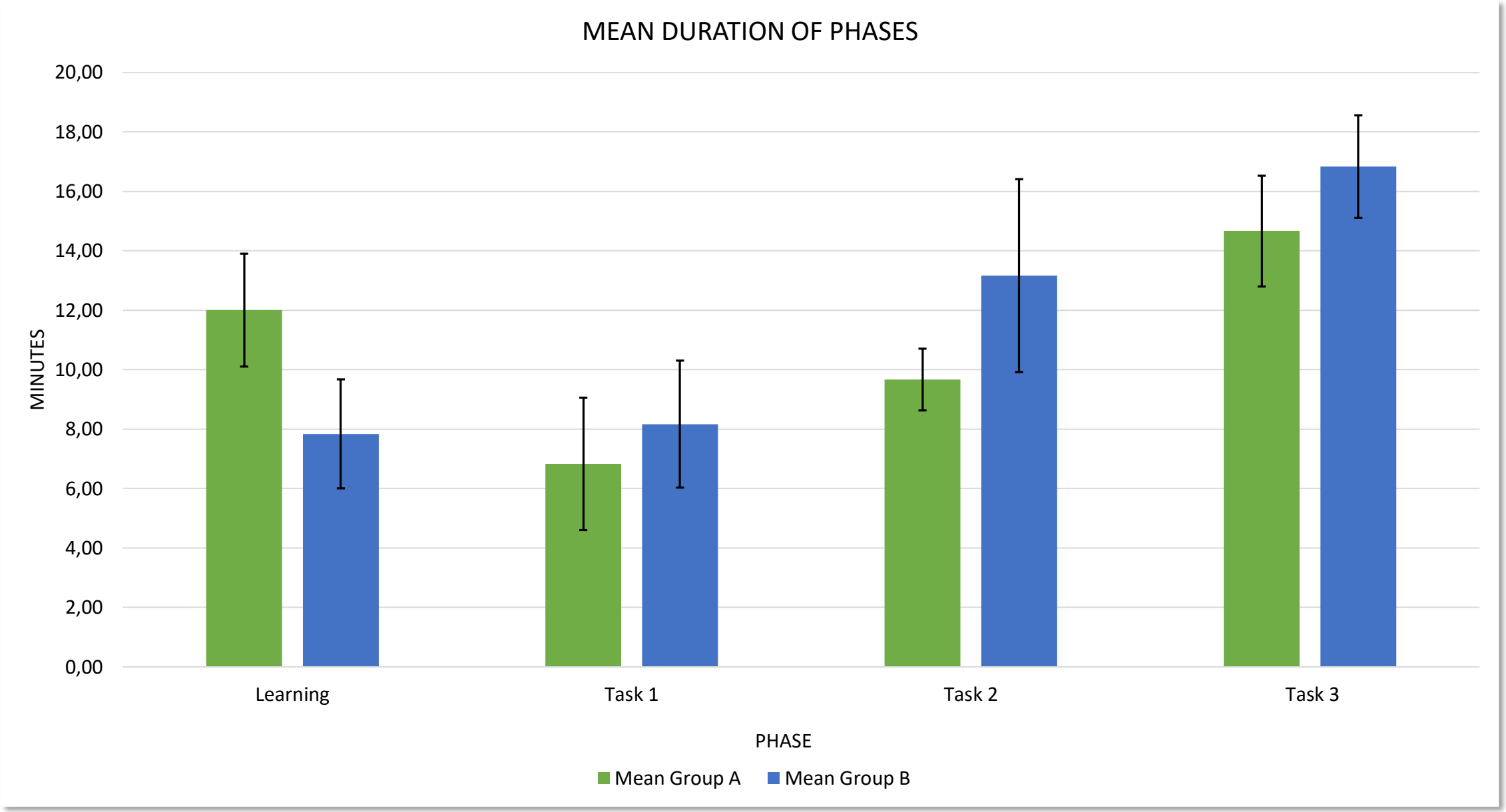
ID	API Experience	Total Experience
A1	10	14
A2	3	4
A3	10	15
A4	4	4
A5	3	9
A6	9	10
B1	7	9
B2	7	7
B3	10	15
B4	2	4
B5	6	10
B6	3	4
Mean	**6,17**	**8,75**
Mean Group A	**6,50**	**9,33**
Mean Group B	**5,83**	**8,17**

Procedure


Preparation
Organizational Questions
Learning Phase (max. 10 min)
Task Phase (max. 35 min)
SUS
Interviews

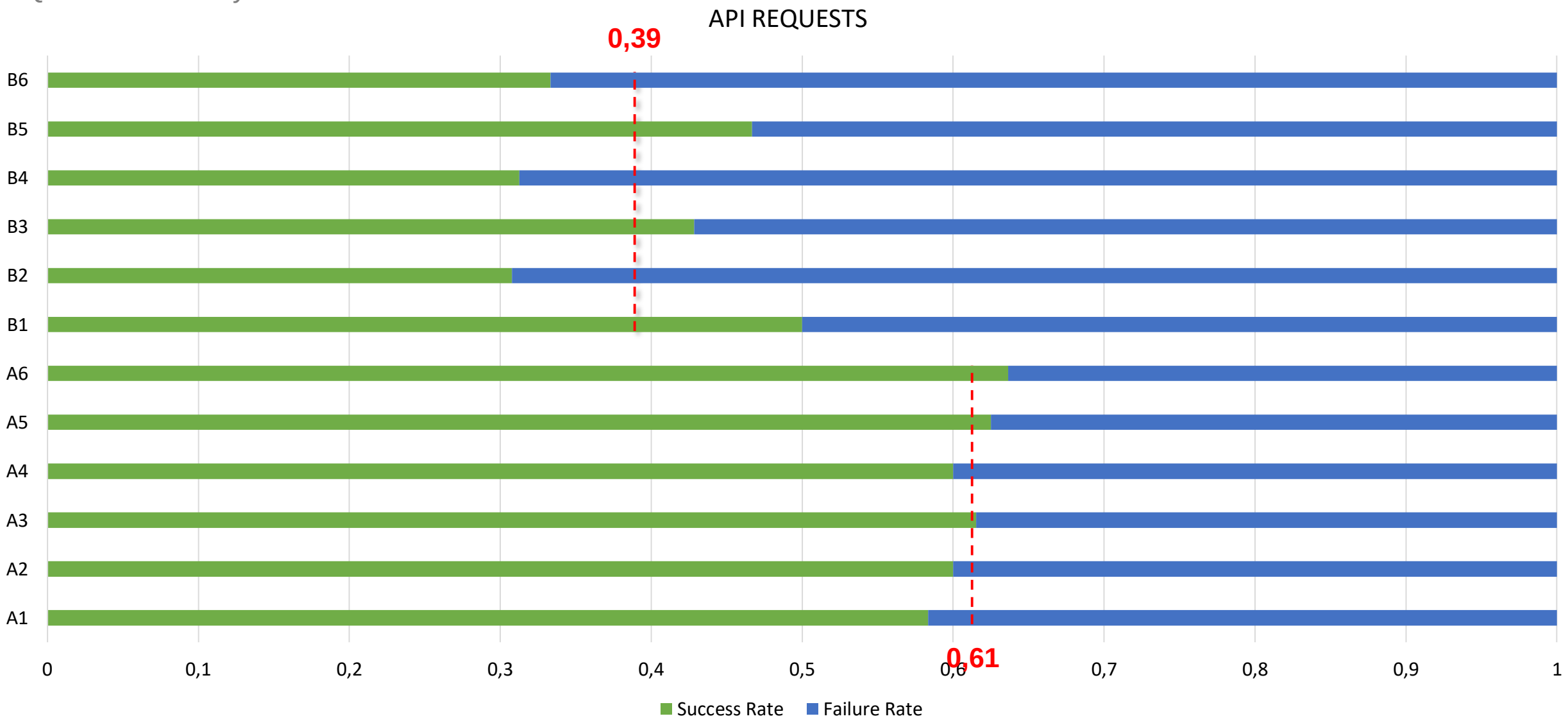
Case Study

Quantitative Analysis



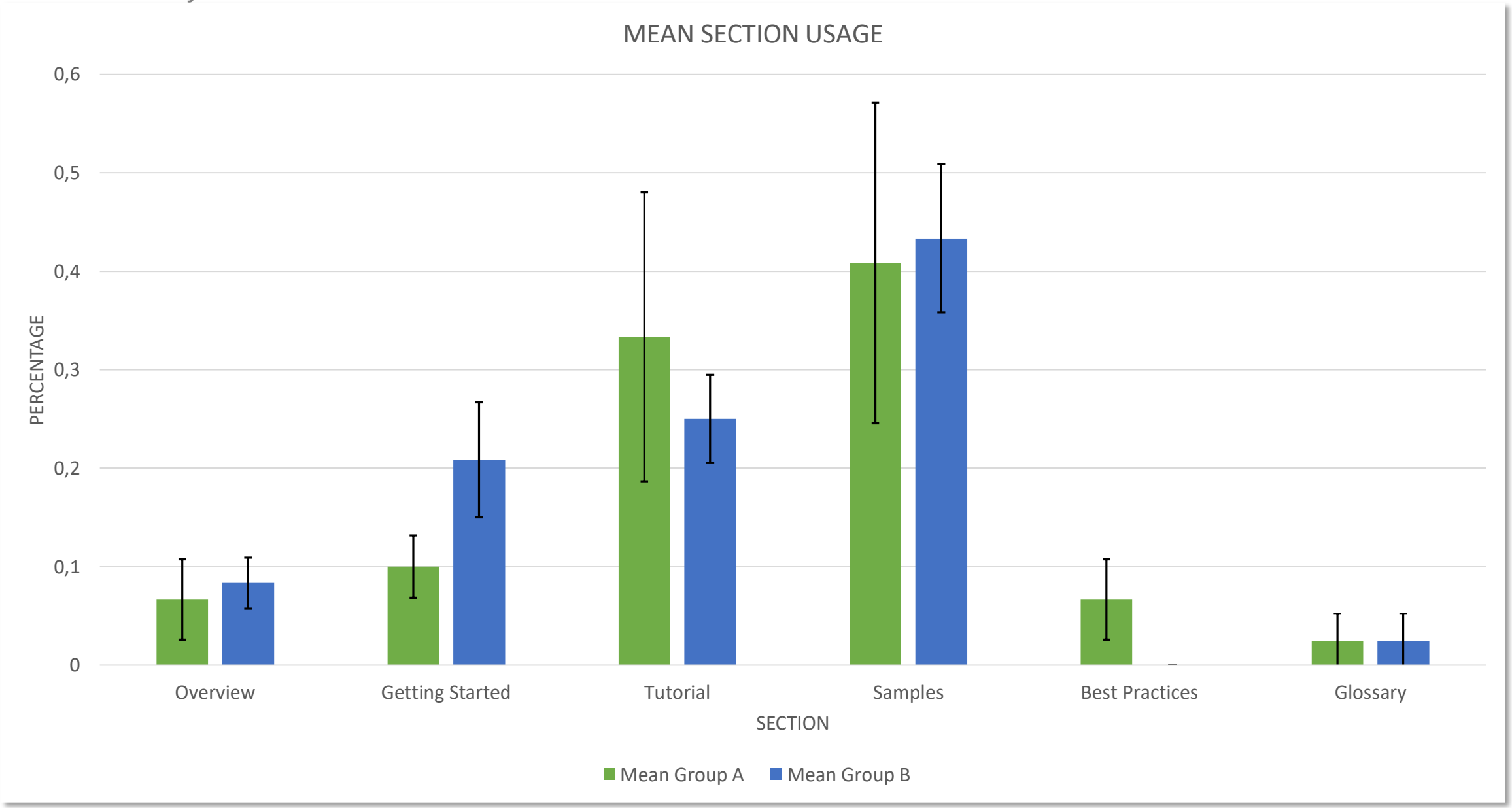
Case Study

Quantitative Analysis



Case Study

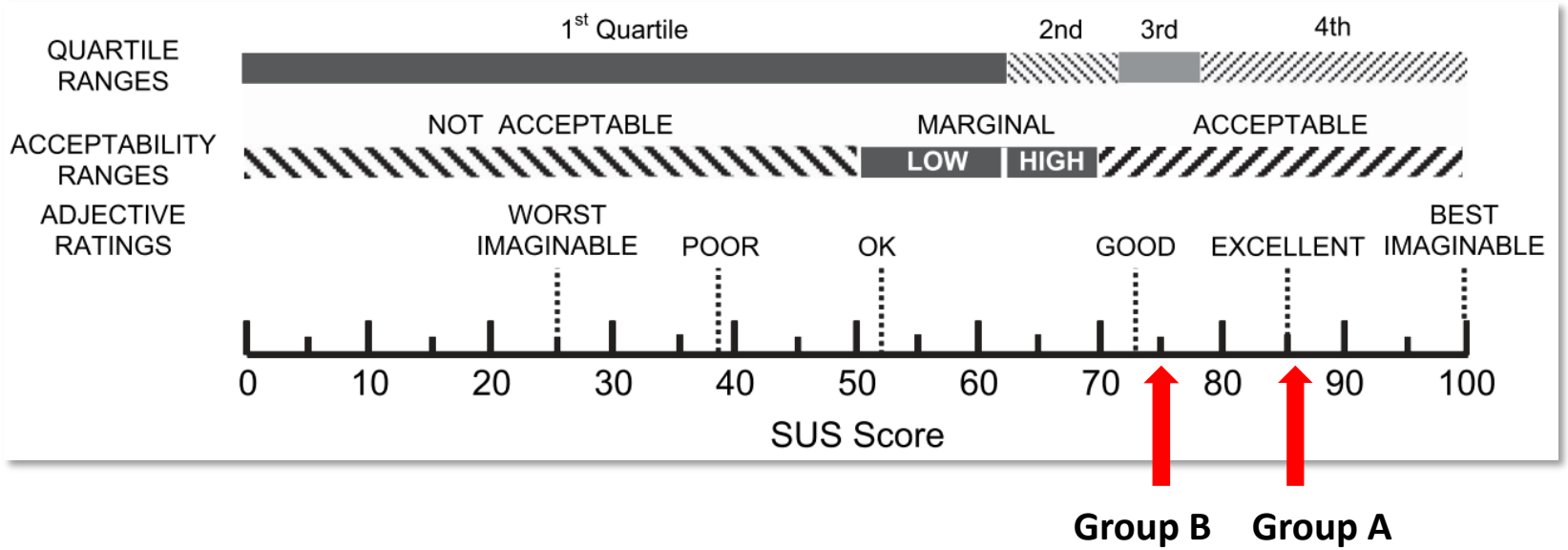
Quantitative Analysis



Case Study

Quantitative Analysis

ID	SUS Score
A1	75
A2	90
A3	87,5
A4	92,5
A5	77,5
A6	92,5
B1	85
B2	75
B3	80
B4	77,5
B5	80
B6	52,5
Mean Group A	**85,8**
Mean Group B	**75**



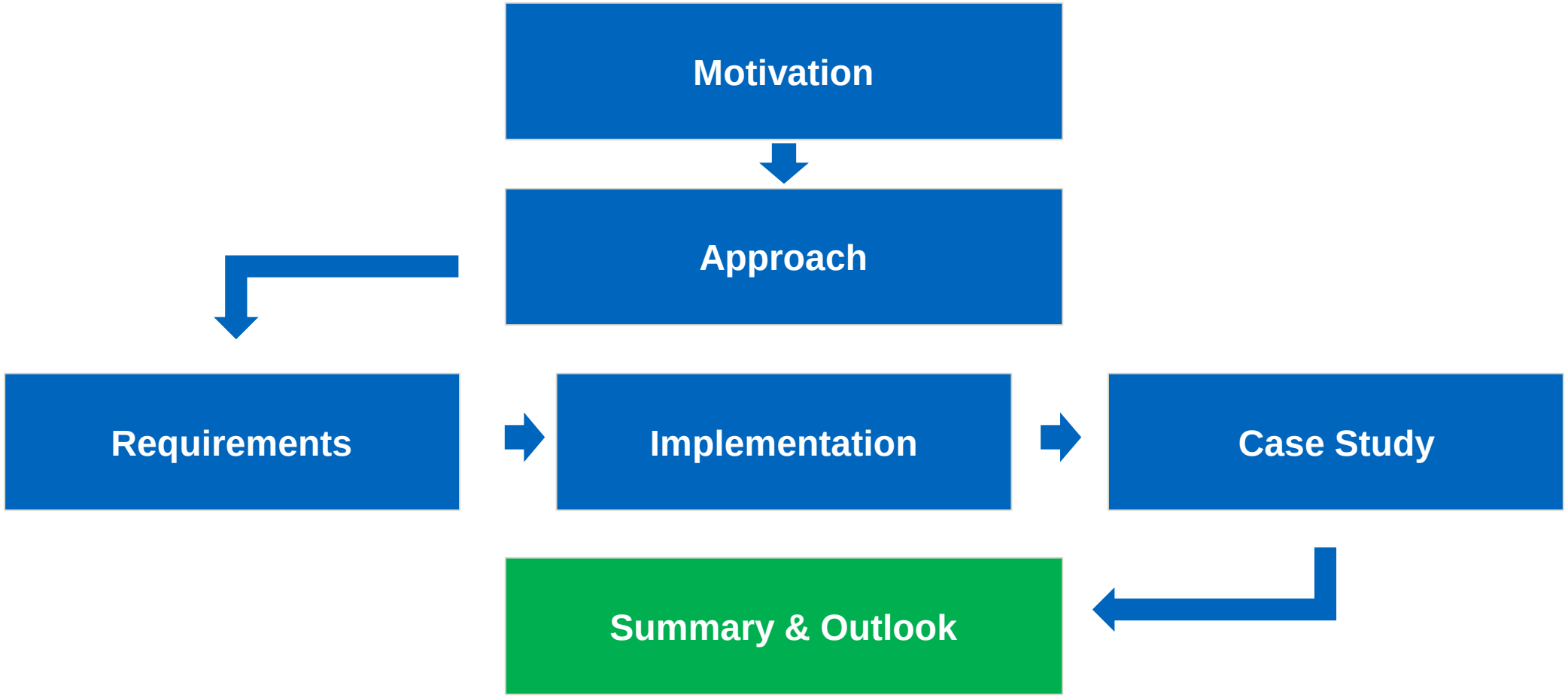
Case Study
Qualitative Analysis

Group	Usefulness?	Favorite sections?	Disliked sections?	Missing features/info?
A & B	- Specificaton - Playground	- Specification - Playground	- Glossary - Overview	- "Helper buttons"
Only A	- Tutorial - Tool support	- Tutorial - References - "Good" coverage - Best practices (?)		- Prerequisites
Only B	- Samples	- Samples	- Descriptions	- More examples - Higher complexity - Links between resources - "Better" descriptions

Case Study
Recommendations

Topic	Recommendation
Terminology	A glossary should be part of the documentation for **high-complexity APIs** but **not for simple APIs**. For simple APIs, the glossary is perceived as unnecessary.
References	The high-level documentation should **reference the low-level documentation** to facilitate lookups and searches.
Coverage	Increasing the **coverage of the most important cases** with API examples might improve developer performance and perceived usability.
Complexity	The complexity of examples and usage scenarios presented in the documentation should **get increasingly higher**.
…	…

Outline


Motivation
Approach
Requirements
Implementation
Case Study
Summary & Outlook

Summary & Outlook

Summary

Usage scenarios & examples are powerful levers with significant influence on devX

Improving the devX requires a combination of technical & conceptual elements

Effect & suitability of features heavily depend on the API's characteristics

Outlook

Longitudinal studies to measure lasting effects beyond devX

How to incorporate beneficial features into automatic workflows & generators?

More fine-grained studies to find individual contributions of e.g. playground, spec, …

Arif Cerit
arif.cerit@tum.de

Technische Universität München
Faculty of Informatics
Chair of Software Engineering for Business Information Systems

Boltzmannstraße 3
85748 Garching bei München

wwwmatthes.in.tum.de

Backup

References

Cruzes & Dyba (2011) ”Research synthesis in software engineering: A tertiary study”

Fagerholm & Münch (2012) ” Developer experience: Concept and definition”

Hevner et al. (2004) ” Positioning and Presenting Design Science Research for Maximum Impact”

Lethbridge et al. (2005) ” How software engineers use documentation: the state of the practice”

Maalej & Robillard (2013) ”Patterns of Knowledge in API Reference Documentation”

Meng et al. (2019) ”How Developers Use API Documentation: An Observation Study”

Peffers et al. (2007) ”A Design Science Research Methodology for Information Systems Research”

Robillard (2009) ”What Makes APIs Hard to Learn? Answers from Developers”

Runeson & Höst (2007) ”Guidelines for conducting and reporting case study research in software engineering”

Sohan et al. (2017) ”A study of the effectiveness of usage examples in REST API documentation”

Uddin & Robillard (2015) ”How API Documentation Fails”

Zhang et al. (2019) ”Enriching API Documentation with Code Samples and Usage Scenarios from Crowd Knowledge”

Requirements

Existing concepts

	Concepts				
Articles	**Mapping Scenarios to API**	**Tests as Examples**	**Integrate Concepts & Code**	**Conceptual Knowledge**	**Complexity**
Hoffman & Strooper (2000)		X			
Hoffman & Strooper (2003)		X			
Ko et al. (2004)	X				
Ko et al. (2007)				X	
Robillard & DeLine (2010)	X		X	X	X
Myers et al. (2010)			X		
Nasehi & Maurer (2010)		X	X		X
Ko & Riche (2011)			X	X	
Kuhn & DeLine (2012)	X				
Nasehi et al. (2012)			X		X
Watson et al. (2013)				X	
Glassman et al. (2018)				X	
Meng et al. (2018)	X		X	X	X
Meng et al. (2019)	X			X	

Knowledge in API Documentation by Maalej & Robillard (2013)

Functional	Non-functional	Conceptual & Structural
• Features • Behavior	• Quality Attributes	• Concepts • Directives • Purpose & Rationale • Control-Flow • Structure • References

Knowledge in API Documentation by Maalej & Robillard (2013)

Knowledge Type	Description (Excerpt)
Functionality and Behavior	Describes what the API does (or does not do) in terms of functionality or features. Describes what happens when the API is used (a field value is set, or a method is called).
Concepts	Explains the meaning of terms used to name or describe an API element, or describes design or domain concepts used or implemented by the API.
Directives	Specifies what users are allowed / not allowed to do with the API element. Directives are clear contracts.
Purpose and Rationale	Explains the purpose of providing an element or the rationale of a certain design decision. Typically, this is information that answers a "why" question: Why is this element provided by the API? Why is this designed this way? Why would we want to use this?
Quality Attributes and Internal Aspects	Describes quality attributes of the API, also known as non-functional requirements, for example, the performance implications. Also applies to information about the API's internal implementation that is only indirectly related to its observable behavior.
Control-Flow	Describes how the API (or the framework) manages the flow of control, for example by stating what events cause a certain callback to be triggered, or by listing the order in which API methods will be automatically called by the framework itself.
Structure	Describes the internal organization of a compound element (e.g. important classes, fields, or methods), information about type hierarchies, or how elements are related to each other.
Patterns	Describes how to accomplish specific outcomes with the API, for example, how to implement a certain scenario, how the behavior of an element can be customized, etc.
Code **Examples**	Provides code examples of how to use and combine elements to implement certain functionality or design outcomes.
Environment	Describes aspects related to the environment in which the API is used, but not the API directly, e.g., compatibility issues, differences between versions, or licensing information.
References	Includes any pointer to external documents, either in the form of hyperlinks, tagged "see also" reference, or mentions of other documents (such as standards or manuals).
Non-information	A section of documentation containing any complete sentence or self-contained fragment of text that provides only uninformative boilerplate text.

Requirements

Functional & Non-functional

Feature	% of Participants	% of all Codings
Functional: Executable Examples	64	2,12
Functional: Adaptable Examples	36	1,35
Functional: Correct Examples	64	3,47
Functional: Feedback	29	0,96
Functional: Entry Points & Barriers	57	5,01
Functional: Textual Descriptions	64	3,28
Functional: Tooling	86	5,01
Non-Functional: Implementation Details	71	2,31
Non-Functional: Copy & Paste	36	1,93
Non-Functional: Readability	29	1,35
Non-Functional: Consumability	86	5,01
Non-Functional: Coverage	50	1,93
Non-Functional: Complexity	57	3,08
Non-Functional: Visualization	29	1,35

Requirements

Conceptual & Structural

Knowledge Type	% of Participants	% of all Codings
Concepts	79	6,17
Directives	79	5,59
Purpose & Rationale	57	2,89
Control-Flow	86	6,94
Structure	79	3,66
References	86	4,24

Challenges

- Problem Relevance

Highly relevant	Moderately relevant	Irrelevant
11	3	0

- Problem Frequency

Frequent	Sometimes	Seldom	Never
13	1	0	0

Challenges

(A | D) = interviewee group mentioning this problem

Providers	Documentation/Specification	Consumers
No knowledge of **consumer needs** (A)	**Disconnection** API and documentation (A, D)	Underestimation of **complexity** (A)
No knowledge of how their API is used (A)	No guidance for **error cases** (A, D)	No understanding of **usage context** (A, D)
Documentation has lower priority (A, D)	Does not contain the **„How“** of the API (A)	Struggle with error handling (A) → **need insight** (D)
Unable to incorporate **continuous feedback** (A)	Does not explicitly contain **solutions to common problems** and scenarios (A, D)	Lack of trust in documentation → trial & error approach (D)
	Missing the **„big picture“** (A)	

General
Feedback over **multiple channels** (A, D)
Inter dependent teams (A)
Semantic information are scattered across locations (D)

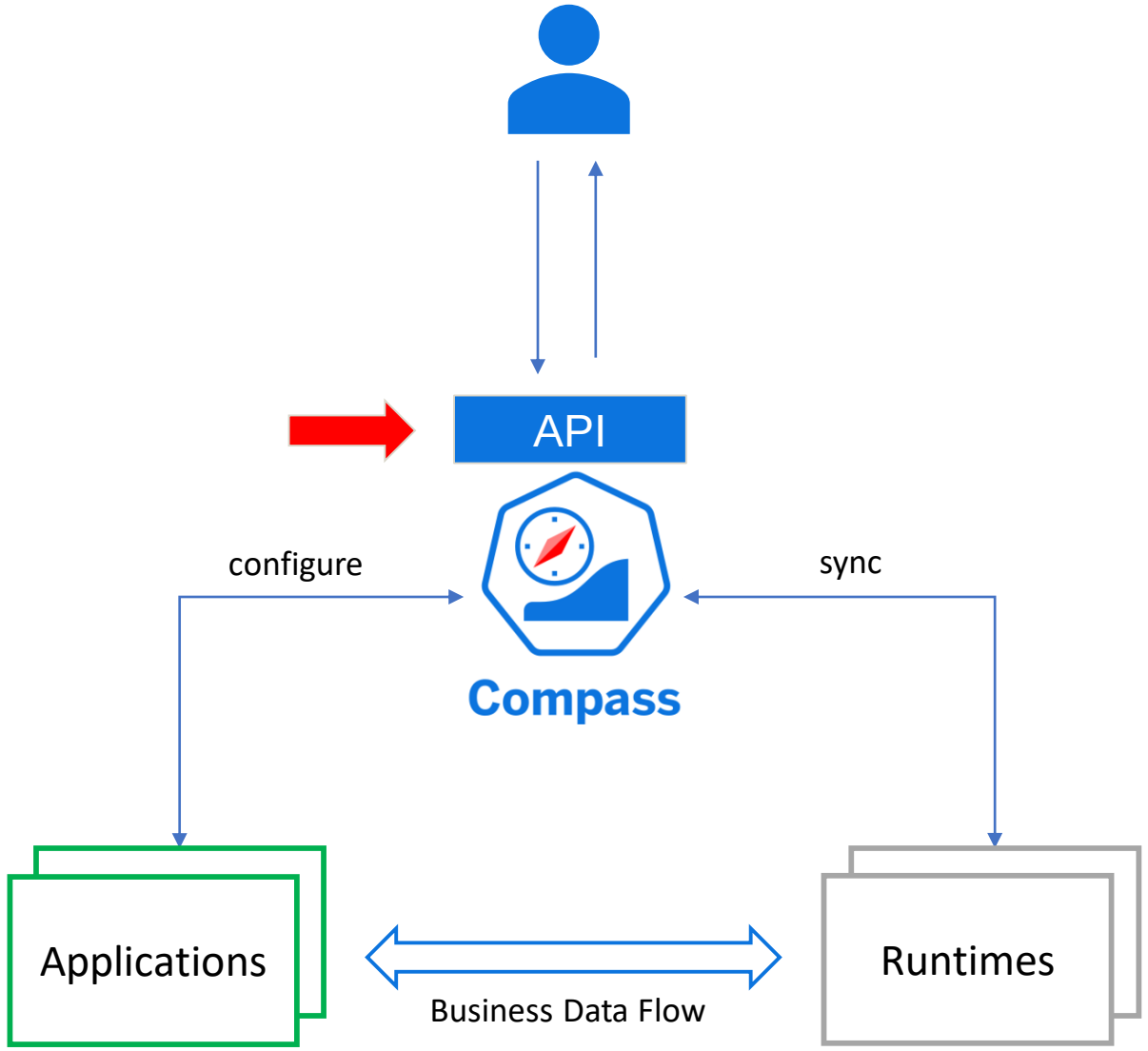
Challenges

- A3: „The **Why** and **What** of the API are often disconnected resulting in a communication hurdle.“
- A4: „The information asymmetry is **mutual**. Providers don‘t know what problems their consumers have. And consumers don‘t know how the flows through the API work.“
- A5: „A significant problem is **dealing with errors** on the consumer side.“
- A7: „Often the providers can‘t address the common API problems because they don‘t know what they are. This knowledge must be acquired and incorporated into the API.“
- D3: „All problems with the documentation and specification are caused by providers who **don‘t know their consumers**.“

Selection criteria

- **Organizational**
  - Team & document availability
  - Easy collaboration

- **Technical**
  - Moderate/high complexity
  - Multiple scenarios
  - No/less mature documentation

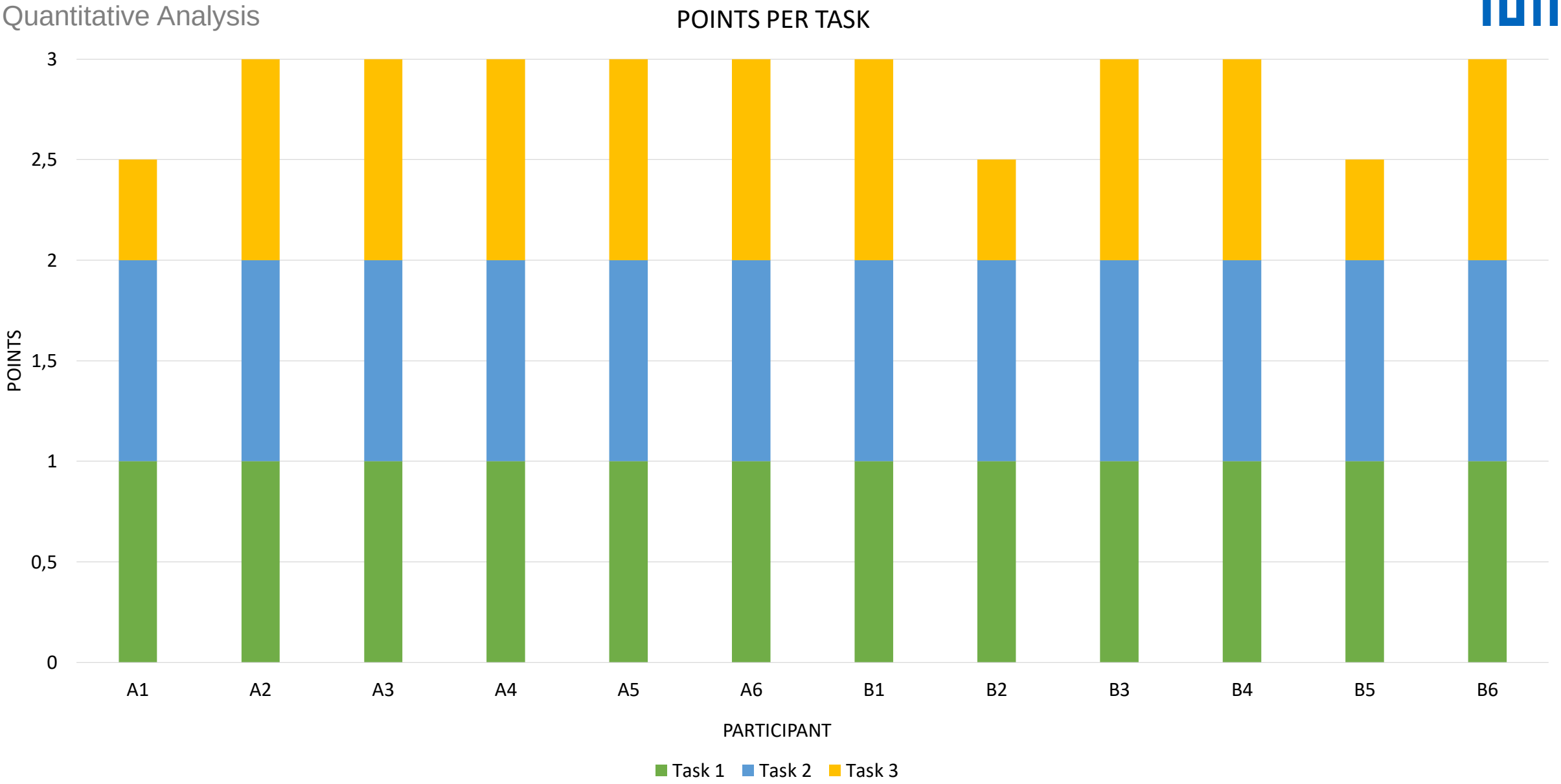


Tasks

ID	1	2	3
Task	Runtime Status	Application with Documents	Adding API to Application
Topic	Runtimes	Applications, Documents	Application, APIs, Specifications
Minimum required steps	2	2	3
Difficulty	Easy	Moderate	Hard

Case Study

Quantitative Analysis



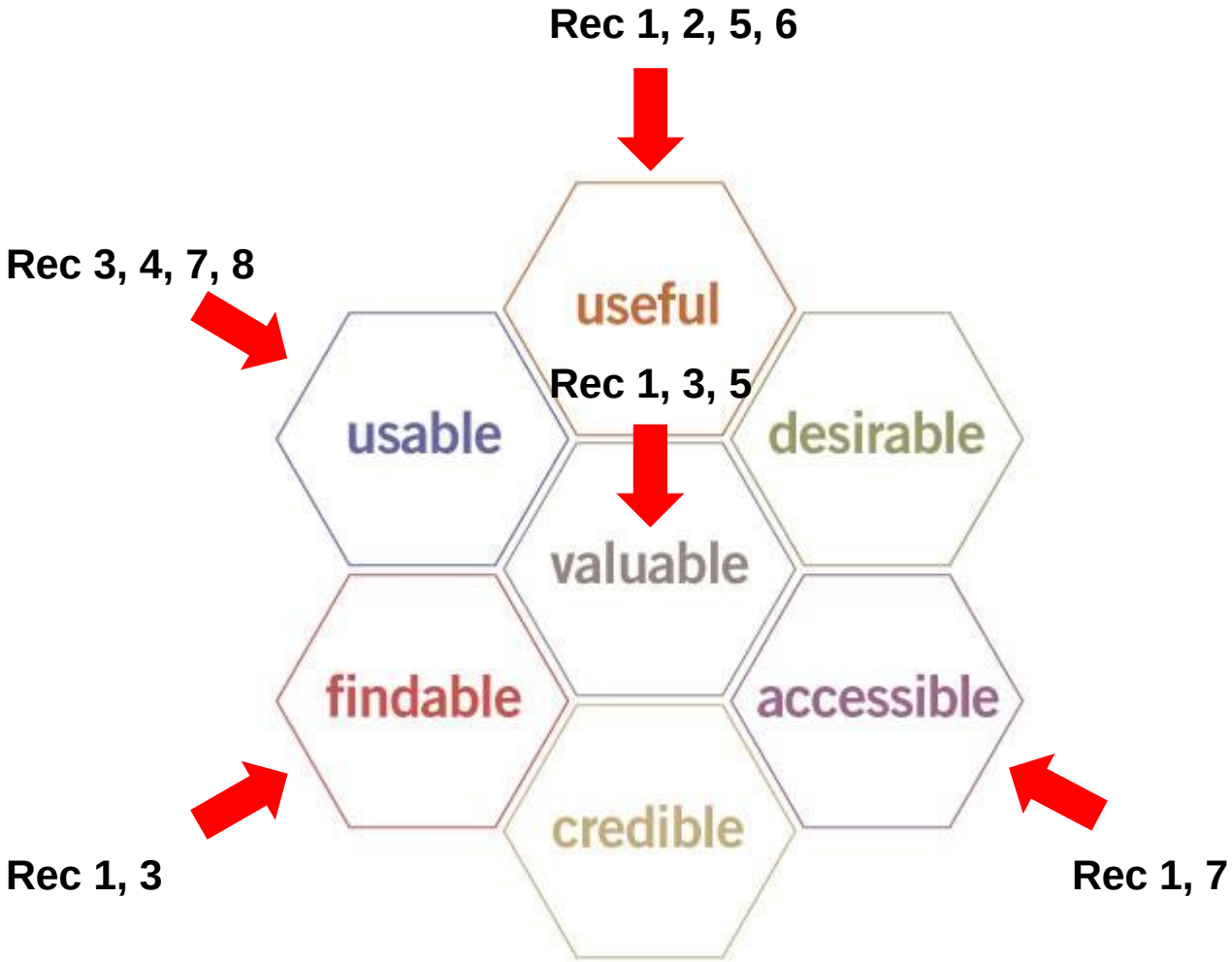
Recommendations

Common Features

ID	Topic	Recommendation
1	Examples	API providers should make an API playground available as an easy way to execute and modify the examples in the documentation.
2	Terminology	A glossary should be part of the documentation for high-complexity APIs but not for simple API. For simple APIs, the glossary is perceived as unnecessary
3	References	The high-level documentation (usage scenarios, examples) should reference the low-level documentation (specification) to facilitate lookups and searches.
4	Descriptions	The textual description accompanying examples and usage scenarios should highlight crucial information, be concise, and focused.
5	Coverage	Increasing the coverage of the most important cases with API examples might improve the developer experience with regards to developer performance and perceived usability.
6	Complexity	The complexity of examples and usage scenarios presented in the documentation should get increasingly higher.
7	Tool support	The examples and usage scenarios should be supported by the right tools. Ideally, the addressed API consumers should be power users of the supported tools.
8	Helper Buttons	The examples and usage scenarios should be supported "helper buttons". These help API consumers to work efficiently with the resources by automating repetitive tasks.

Case Study

Impact on Developer Experience



Morville & Sullenger (2010)

Summary & Outlook

RQ1	RQ2	RQ3
Concept matrix with existing concepts observed across 42 research papers	8 functional and 7 non-functional **requirements** 13 implications for **conceptual & structural knowledge**	8 recommended **features and characteristics** based on observed evidence