

Improving the Quality of OpenAPI Specifications Using TypeScript Types and Annotations

Wolfgang Hobmaier, 20.04.2020

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

Motivation

Research Questions

Initial Results

Timeline

Documentation is important, but usually of bad quality

- APIs have become so essential to businesses that 85% consider web APIs and API-based integration fundamental to their business strategy and continued success. [1]
- Ideally, [companies] should use open, well-documented services to accelerate time to prototype. Expecting constant change and speedy execution is part of the shift to the API economy.[2]
- Clear, easy-to-access, and easy-to-grasp documentation is a prerequisite for API success. [1]

- Documentation quickly becomes stale and out-of-date. When it does, users fail to derive value from it and, worse yet, lose trust in the company. [1]
- A growing body of evidence shows that many APIs are difficult to use [3]

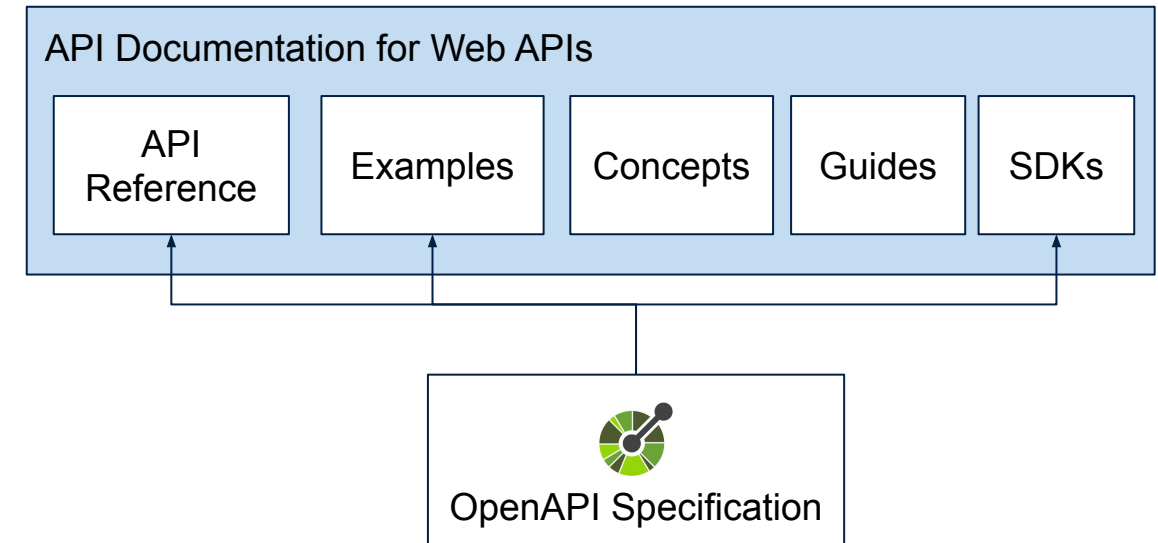
[1] Falon Fatemi "3 Keys To A Successful API Strategy"

[2] Deloitte University Press, "API economy - From systems to business services"

[3] J. Stylos, A. Faulring, Z. Yang and B. A. Myers, "Improving API documentation using API usage information," 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Corvallis, OR, 2009, pp. 119-126.

Reference documentation for Web APIs is important, but needs to be improved

- Programmers spend significant time learning new APIs [1]
- Most of the time in the Documentation is spent on Reference Documentation and Examples [1]
 - Average time spent in API Reference: 39,6%
 - Average time spent in Examples: 40,5%
- Publishing Usage Information and Examples based can be based on Tests to ensure that the information is verified [2]
- "[A]bout 49% of the endpoints have some mismatch between OASs from the documentation and the calling response." [3]
- 4 major categories [3]:
 - Undocumented keys
 - Dynamic keys
 - Unreturned keys
 - Types mismatched

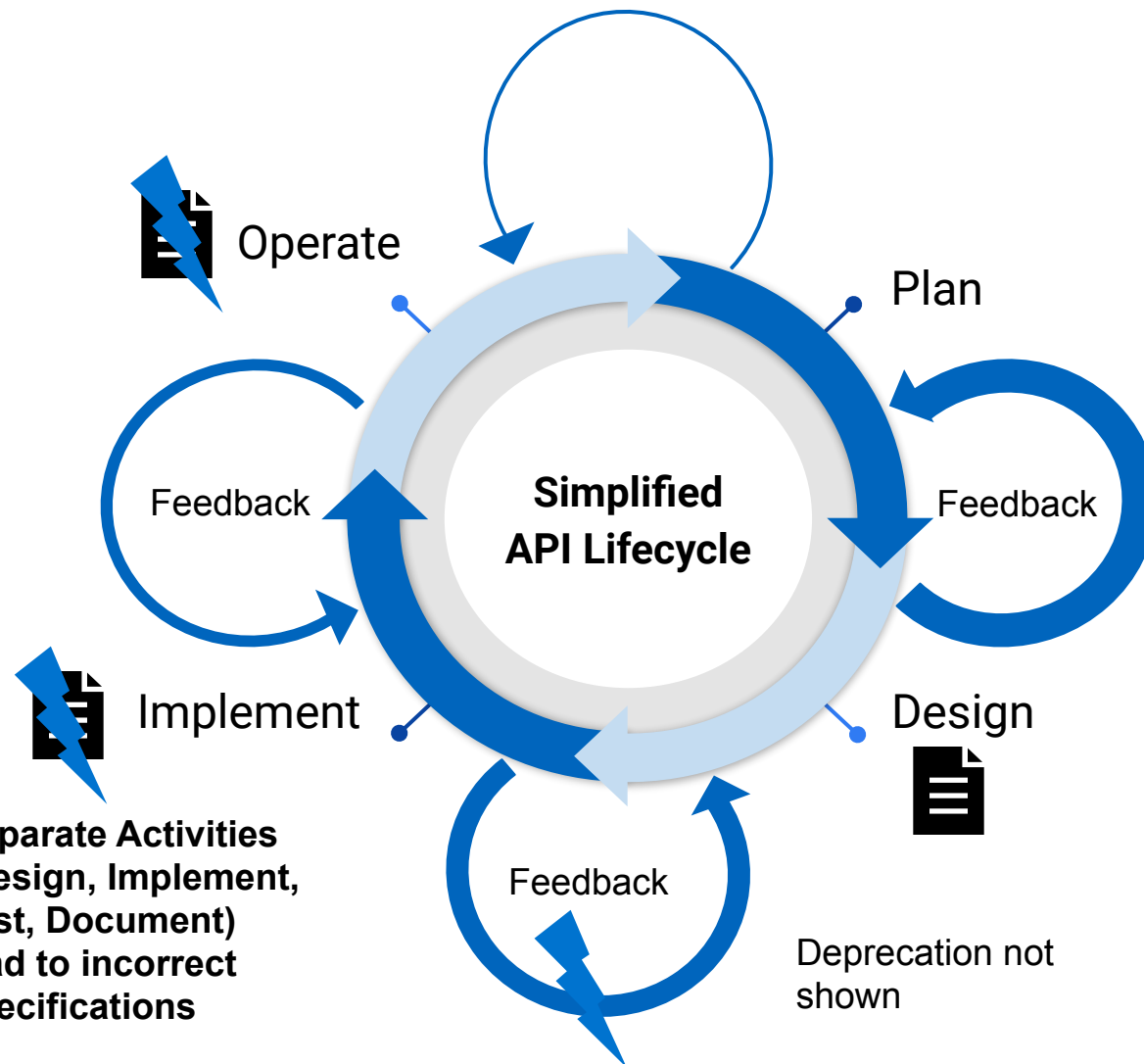


[1] Meng, M., Steinhardt, S. and Schubert, A., 2019. How developers use API documentation: an observation study. *Communication Design Quarterly Review*, 7(2), pp.40-49.

[2] Cerit, A "Improving the Developer Experience of API Consumers using Usage Scenarios and Examples"

[3] Hosono, Masaki, et al. "An Empirical Study on the Reliability of the Web API Document." *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018.

Specification-driven API Lifecycle

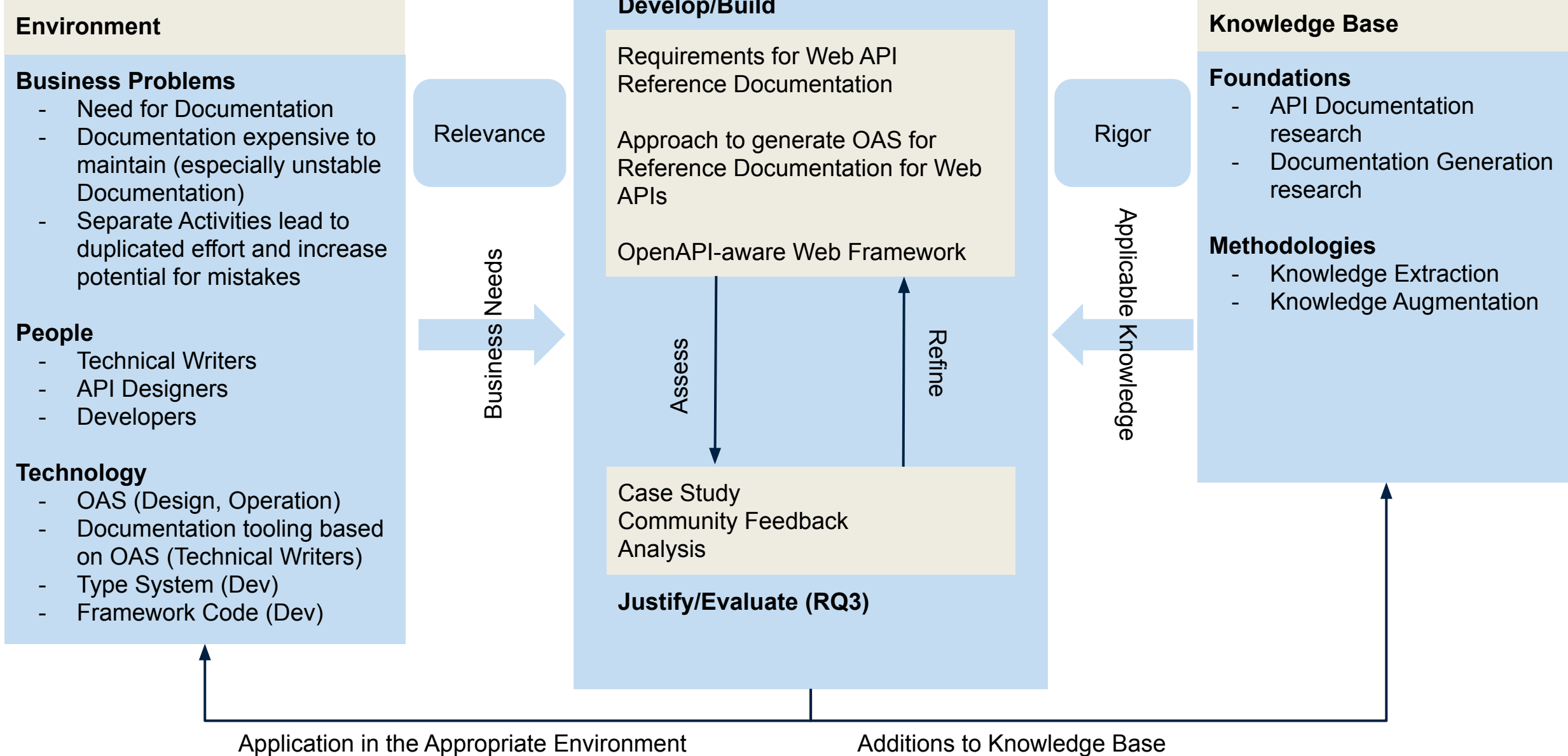


What are your biggest obstacles to providing up-to-date API documentation? *(Select all that apply)*



SmartBear – State of the API 2019

For more information visit <https://smartbear.de/resources/ebooks/the-state-of-api-2019-report/>



RQ1

What are elements of API Reference Documentation for Web APIs?

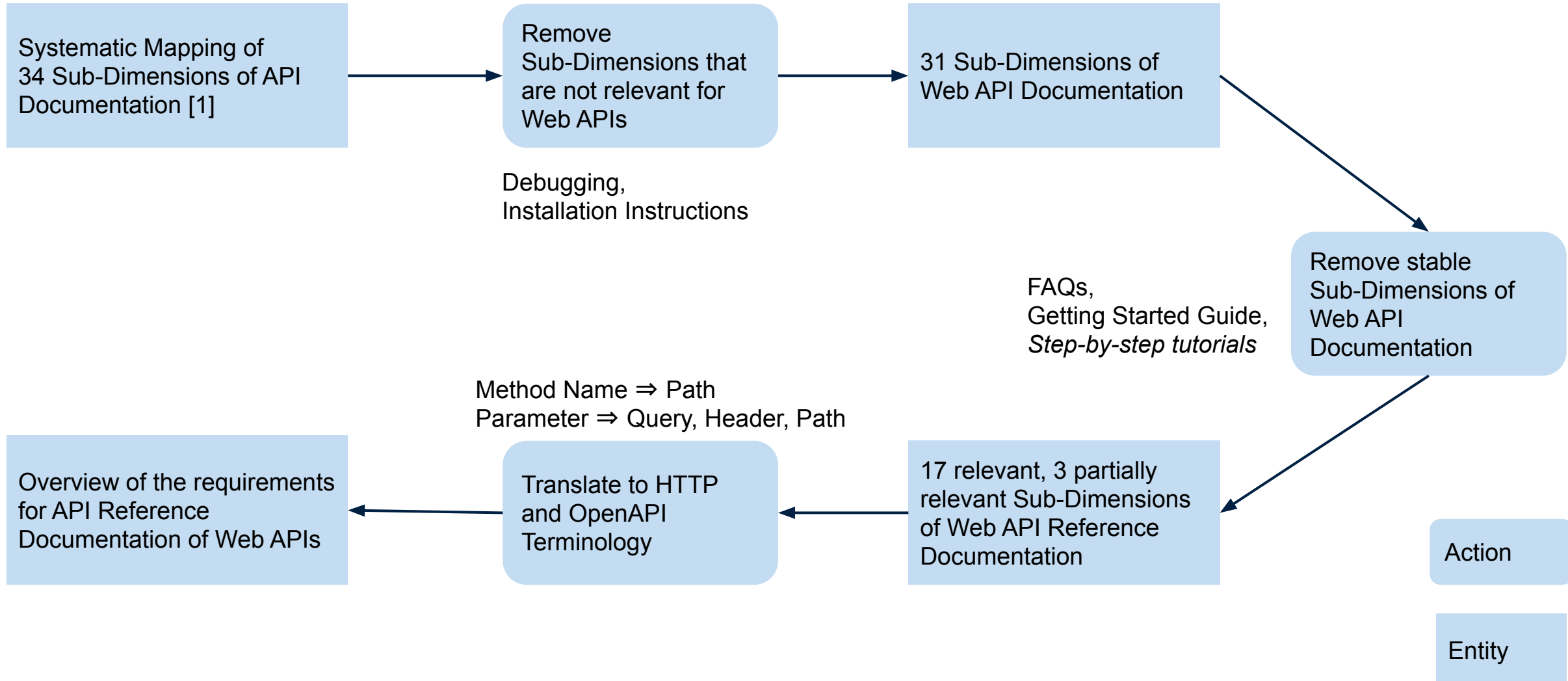
RQ2

What are possible approaches to generate accurate, correct, complete and up-to-date API Reference Documentation of Web APIs?

RQ3

Would developers use a OpenAPI driven framework?

RQ1: What are elements of API Reference Documentation for Web APIs: Approach



[1] A. Cummaudo, R. Vasa and J. Grundy, "What should I document? A preliminary systematic mapping study into API documentation knowledge," 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Recife, Brazil, 2019, pp. 1-6.

Step 3: Translate to HTTP / OpenAPI terminology

Usage Description

- **Low-level reference manual: Method, Path, Query, Body, Status, Header**
- Overview of the Component Schemas of the API
- Endpoint descriptions
- **Executable Example Requests (with comments)**
- **Links between Endpoints**
- Example Responses
- An exhaustive list of all major components
- Error Response definitions

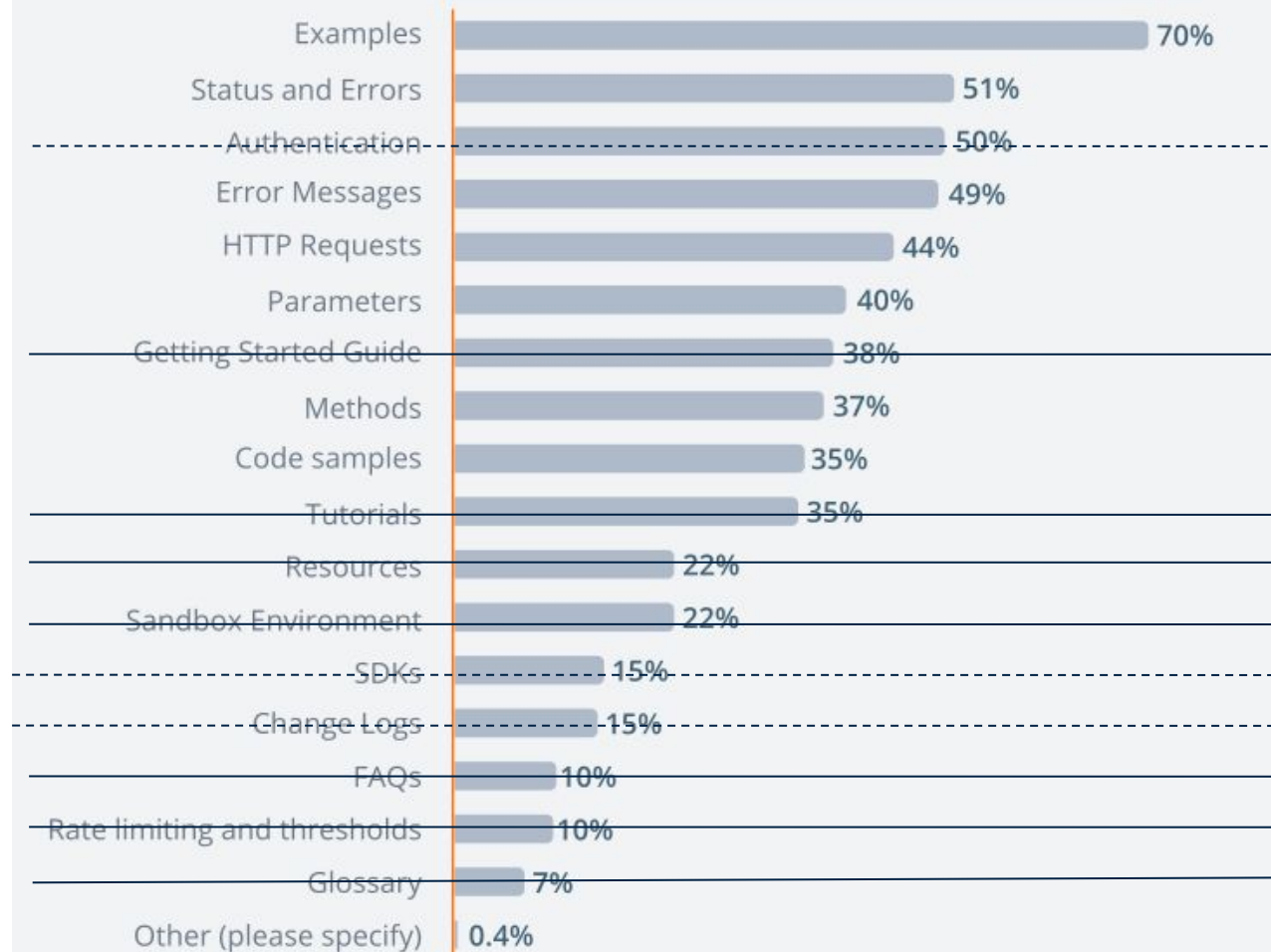
Domain Concepts

- Component descriptions and References
- Definitions of domain-terminology and concepts, with synonyms if applicable

Documentation Presentation

- Searchable knowledge base
- Description Links
- Structured navigational style (e.g., breadcrumbs).
- Visualised map of navigational paths to certain API components
- **Consistent look and feel of documentation.**

Please select the top 5 most important things you look for in API documentation.



SmartBear – State of the API 2019

For more information visit <https://smartbear.de/resources/ebooks/the-state-of-api-2019-report/>

Step 3: Translate to HTTP / OpenAPI terminology

- HTTP Message specification
- JSON Data Format descriptions (JSON Schema)
- Reusable Components and Relationships
- Descriptions, Links, Examples

RQ2: What are possible approaches to generate accurate, correct, complete and up-to-date API Reference Documentation of Web APIs?

- Knowledge Extraction [1]
- Knowledge Augmentation* [1]

RQ2: Approach

- Use Knowledge Extraction whenever possible as the source of truth for the specification
- Ensure Augmented knowledge is correct

[1] Martraire C., "Living Documentation: Continuous Knowledge Sharing by Design", Addison-Wesley Professional, 2019

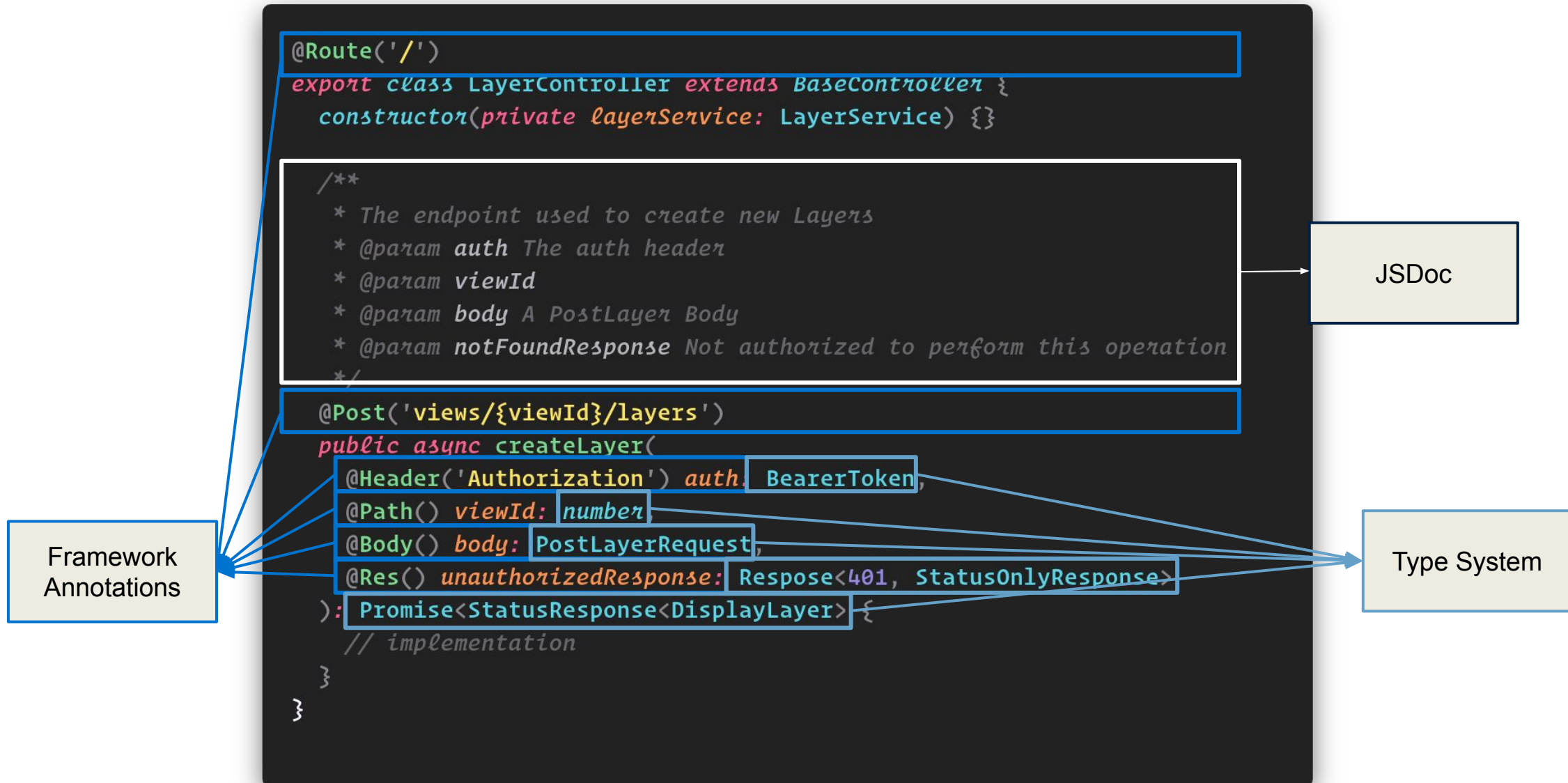
[2] Nybom, K., Ashraf, A. and Porres, I., 2018, August. A systematic mapping study on API documentation generation approaches. In 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 462-469). IEEE.

RQ2: How can we extract knowledge from Source Code

```
@Route('/')
export class LayerController extends BaseController {
  constructor(private layerService: LayerService) {}

  /**
   * The endpoint used to create new Layers
   * @param auth The auth header
   * @param viewId
   * @param body A PostLayer Body
   * @param notFoundResponse Not authorized to perform this operation
   */
  @Post('views/{viewId}/layers')
  public async createLayer(
    @Header('Authorization') auth: BearerToken,
    @Path() viewId: number,
    @Body() body: PostLayerRequest,
    @Res() unauthorizedResponse: Response<401, StatusOnlyResponse>
  ): Promise<StatusResponse<DisplayLayer> {
    // implementation
  }
}
```

RQ2: How can we extract knowledge from Source Code



RQ2: How can we extract knowledge from Source Code

Sources

- Type System
- Framework Annotations
- JSDoc

How can we extract that knowledge

- AST parsing
- Working with the Type Checker
- Reflection

Goal

- Comparison Table
- Initial results

RQ2: How can we extract knowledge from Source Code

	Type System	Annotations	JSDoc-Comment
JSON Schema	✓	—	✗
HTTP Schema	—	✓	✗
Relationships, Components	✓	—	—
Descriptions, Links, Examples	✗	—	✓

	Type System	Annotations	JSDoc-Comment
AST Parsing	—	✓	✓
Type Checker	✓	✗	✗
Reflection	—	✓	✗

⇒ Only a combination of more than one approach can lead to the best results

RQ3: Would developers use a OpenAPI aware Framework?

- Incentives for developers
- Effort
- Limitations

RQ3: Approach

- Create/Improve tooling
- Evaluate with developers

- Improve an existing tool that leverages AST Parsing (<https://github.com/lukeautry/tsoa>)
- Add working with the Type Checker for Advanced Types
- Reusable Definitions (Type Aliases)
- Higher Types (Partial, Required, Mapped/Conditional Types, Generic Type Aliases)
- Typechecked alternative Responses (@Res() decorator)
- @link support for Links

Theoretical

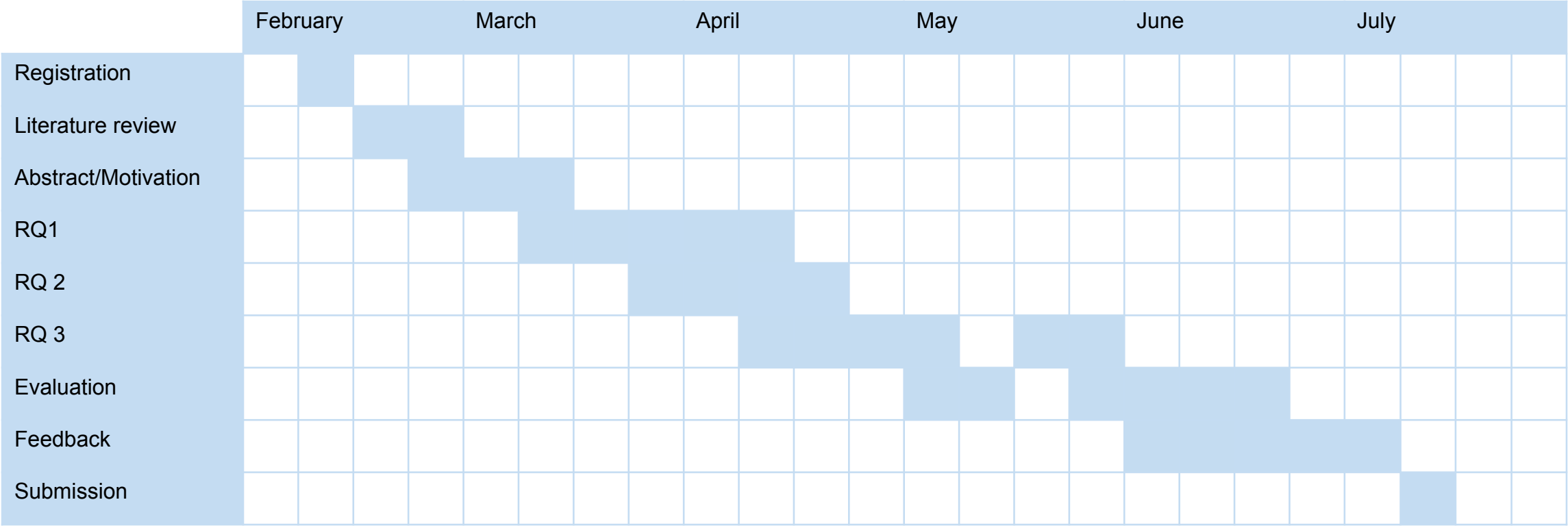
- Reason how tsoa can enforce attributes of good documentation
- Lay out future work

Popularity

- GitHub stars
- Feedback from the community

Coding Assignment

- Implement a To-Do Application with 3 Tools: tsoa, reflection based tool (nestjs/swagger), swagger-jsdoc
- Compare correctness, time, developer experience



Thank you! Questions?



Wolfgang Hobmaier

wolfgang.hobmaier@in.tum.de



- Falon Fatemi "3 Keys To A Successful API Strategy", <https://www.forbes.com/sites/falonfatemi/2019/04/30/3-keys-to-a-successful-api-strategy/#2c8256c378ee>
- Deloitte University Press, "API economy - From systems to business services", <https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/technology/deloitte-uk-api-economy.pdf>
- J. Stylos, A. Faulring, Z. Yang and B. A. Myers, "Improving API documentation using API usage information," 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Corvallis, OR, 2009, pp. 119-126.
- Meng, M., Steinhardt, S. and Schubert, A., 2019. How developers use API documentation: an observation study. Communication Design Quarterly Review, 7(2), pp.40-49.
- Cerit, A "Improving the Developer Experience of API Consumers using Usage Scenarios and Examples", <https://wwwmatthes.in.tum.de/pages/w7n5od3ggdav/Master-s-Thesis-Arif-Cerit>
- Hosono, Masaki, et al. "An Empirical Study on the Reliability of the Web API Document." 2018 25th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2018.
- SmartBear – State of the API 2019, https://static1.smartbear.co/smartbearbrand/media/pdf/smartbear_state_of_api_2019.pdf
- Gregor, Shirley & Hevner, Alan. (2013). Positioning and Presenting Design Science Research for Maximum Impact. MIS Quarterly. 37. 337-356.
- A. Cummaudo, R. Vasa and J. Grundy, "What should I document? A preliminary systematic mapping study into API documentation knowledge," 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Recife, Brazil, 2019, pp. 1-6.
- Martraire C., "Living Documentation: Continuous Knowledge Sharing by Design", Addison-Wesley Professional, 2019
- Adrian Hernandez-Mendez, Niklas Scholz, and Florian Matthes. "A Model-driven Approach for Generating RESTful Web Services in Single-Page Applications". 2018. In Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018). SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT, 480–487.
- Bondel, G., Bui, D. H., Faber, A., Seidel, D., Hauder, M.: Towards a Process and Tool Support for Collaborative API Proposal Management, The 25th Americas Conference on Information Systems (AMCIS), Cancun, Mexiko, 2019.