

Improving the Quality of OpenAPI Specifications Using TypeScript Types and Annotations

Wolfgang Hobmaier, 06.07.2020

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

Motivation

Research Questions

Approach

Results

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- Documentation quickly becomes stale and out-of-date.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][4]
 - "[A]bout 49% of the endpoints have some mismatch between OASs from the documentation and the calling response." [3]
 - Lack of resources, time, and tooling support are the biggest obstacles to implementing a successful API documentation process [5]

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

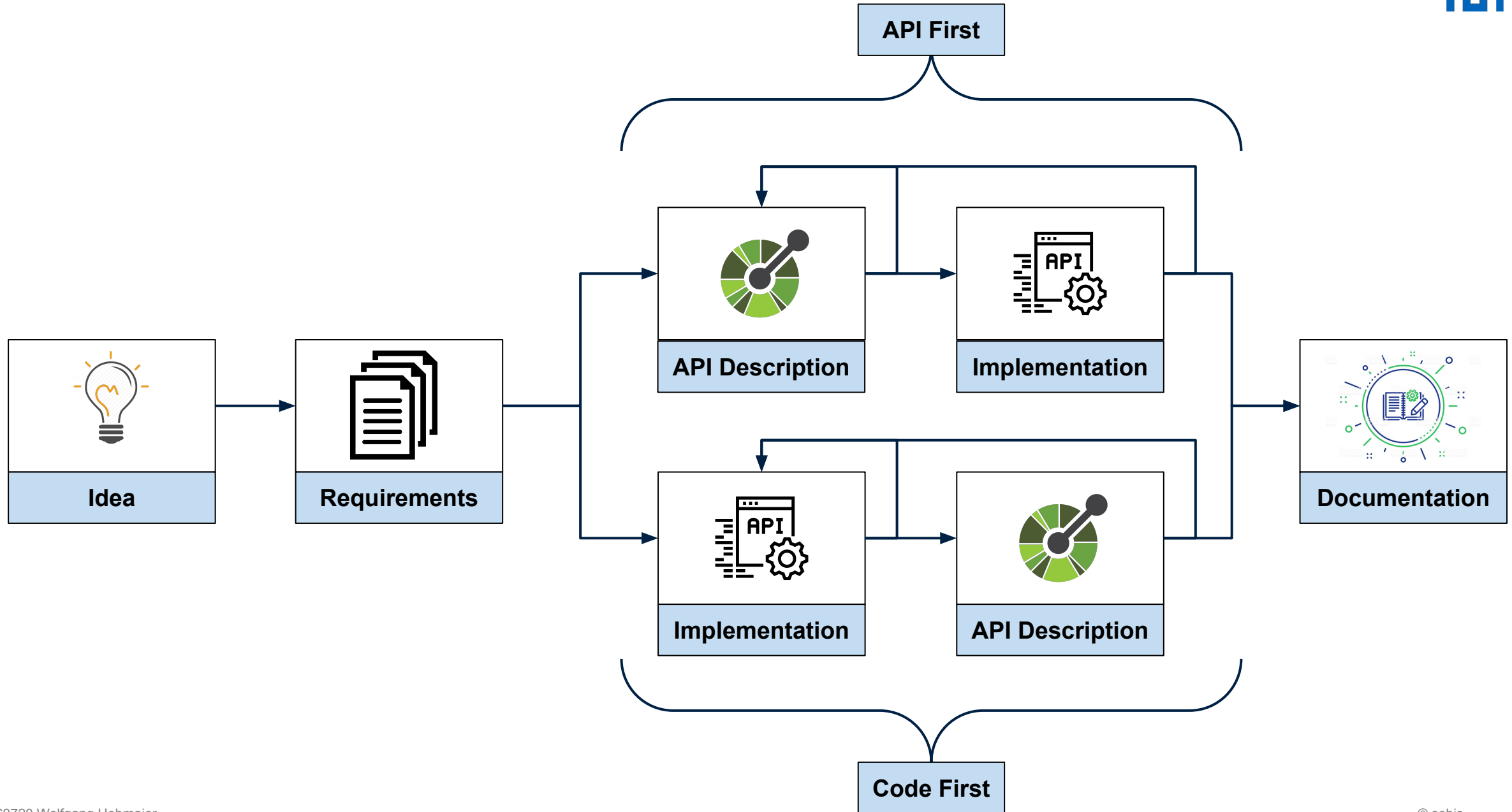
[2] Deloitte University Press, "API economy - From systems to business services"

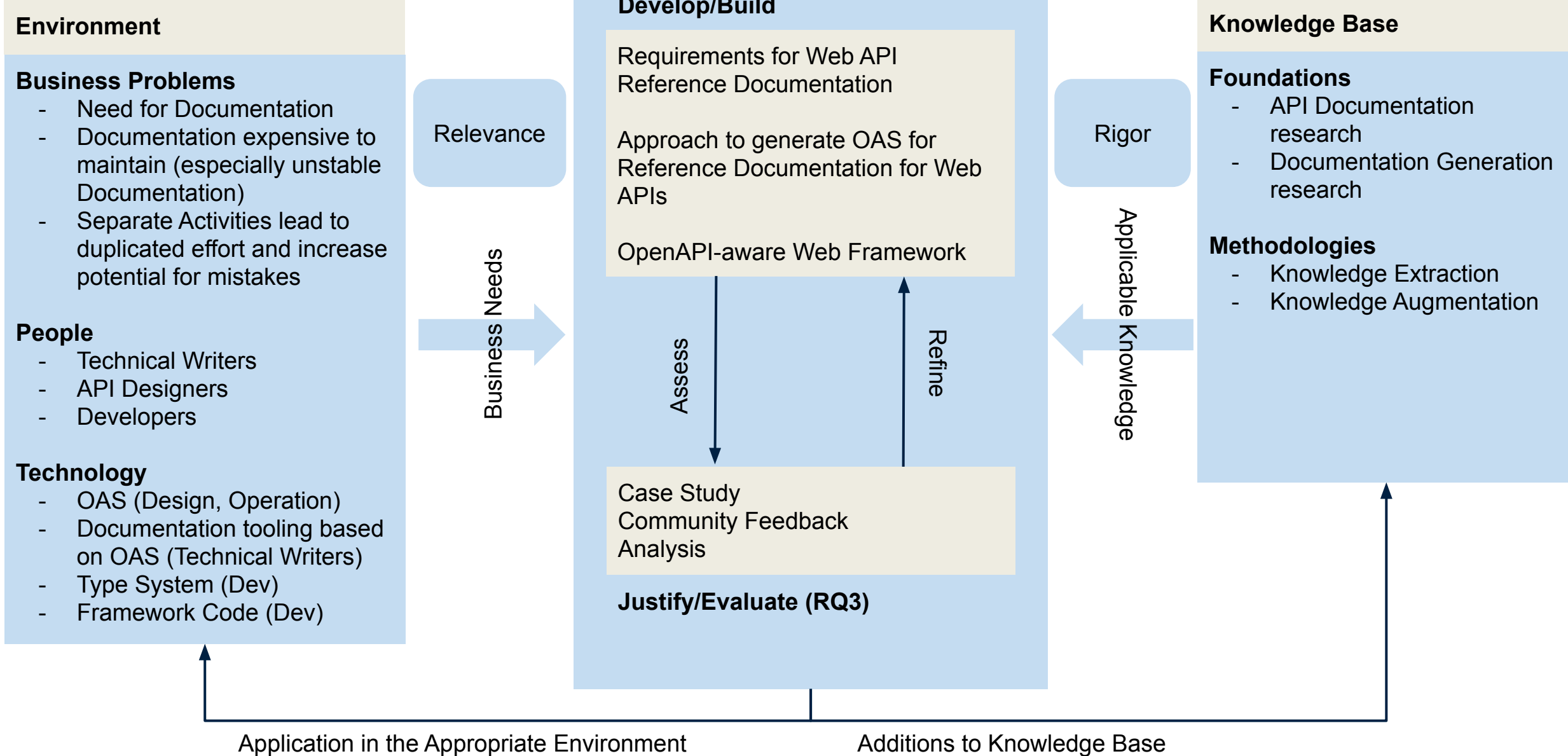
[3] Hosono, Masaki, et al. "An Empirical Study on the Reliability of the Web API Document." *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018.

[4] Adzic, G., 2011. *Specification by example. Book your training with Diaz & Hilterscheid!*, p.20.

[5] SmartBear – *State of the API 2019*

How do Web APIs provide documentation?





RQ1

What are elements of API Reference Documentation for Web APIs?

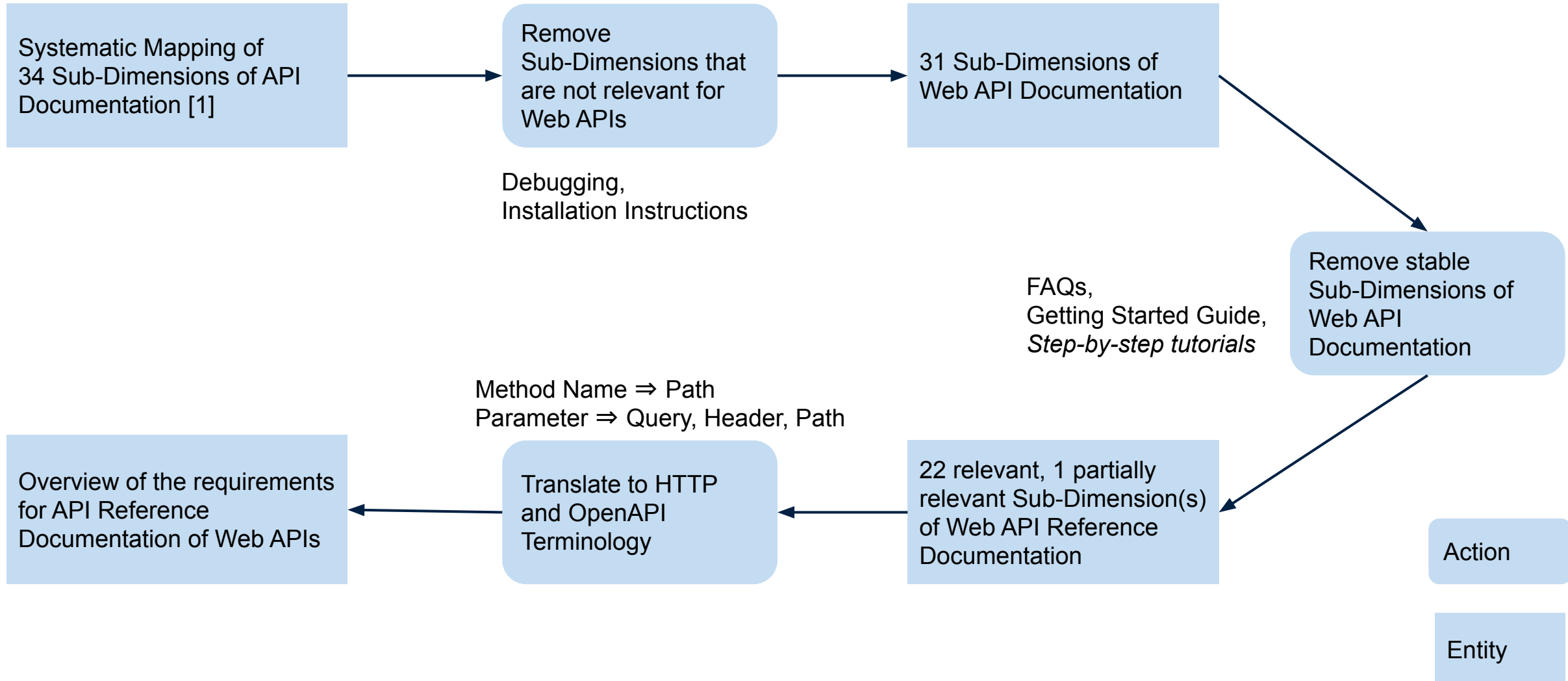
RQ2

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

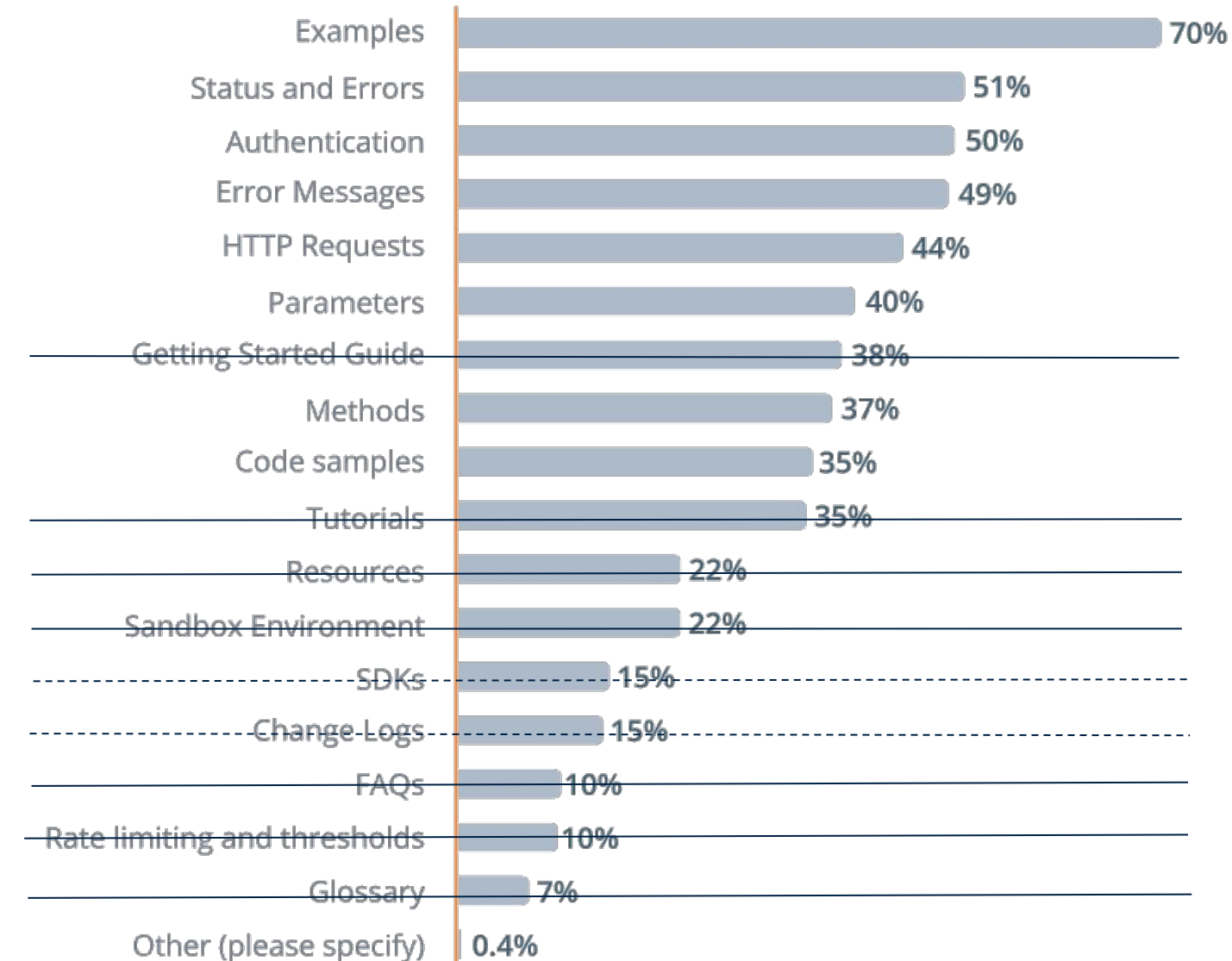
RQ3

Would developers use a OpenAPI driven framework?

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



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

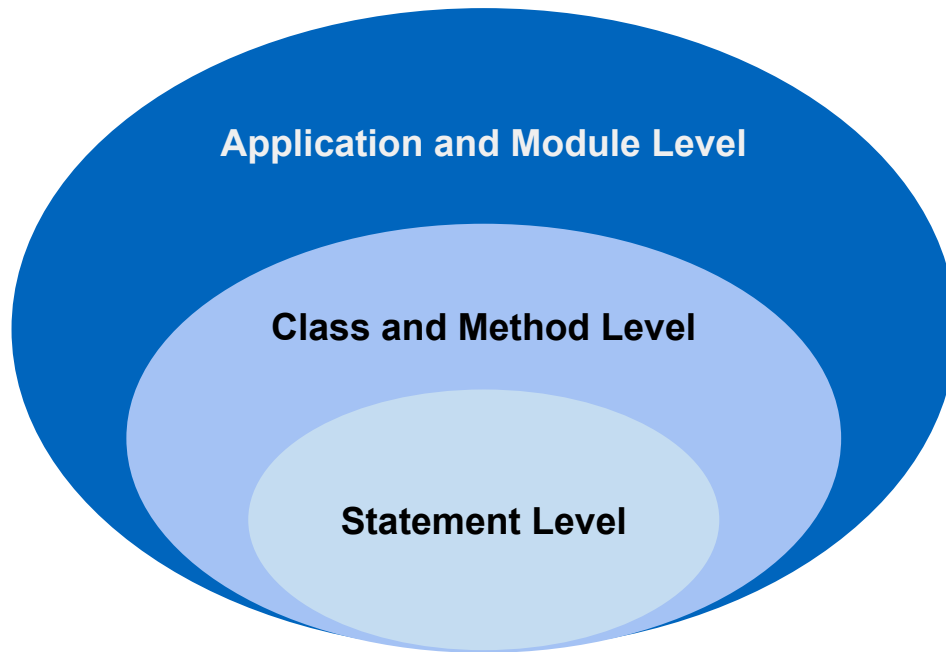


SmartBear – State of the API 2019

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

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

Sources of Documentation: Location



Application/Module Level

- Project Configuration (package.json, tsoa.json)

Class/Method Level

- Parameters: Location, Type
- Return: Status, Headers, Type
- Doc Comments: Descriptions
- Decorators / Annotations: Route, Method

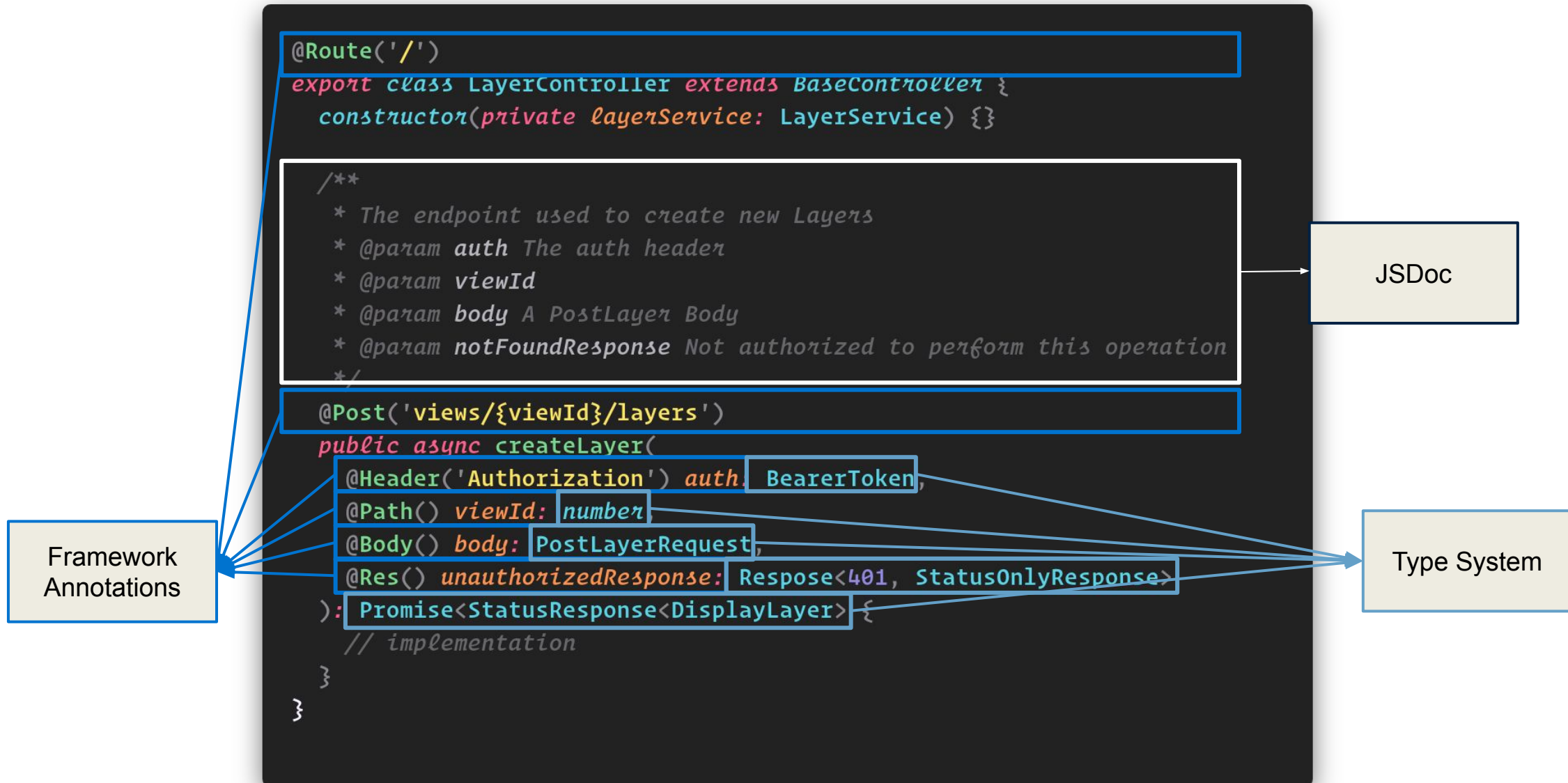
Statement Level

- Not considered

```
@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
  }
}
```

Sources of Documentation: Form



Approaches

Reflection

- Inspect the application programmatically at runtime
- Decorators

AST Analysis

- Inspect the application programmatically at build time
- Macros

Integration Tooling

- Language Server Protocol
- Type Checker

RQ2: How can we extract knowledge from Source Code

	Type System	Annotations	JSDoc-Comments	Configuration
JSON Schema	✓	—	✗	✗
Operation Schema	—	✓	✗	✗
Usability information	✗	—	✓	✓
Metadata	✗	—	✗	✓

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

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

RQ3: Approach

- Evaluate with developers
- Create/Improve tooling

RQ3: Would developers use a OpenAPI aware Framework?

- Incentives for developers
- Effort
- Limitations

Selected Contributions: Improving an existing tool that leverages AST Parsing



JSDoc

- Support for multiple response examples

Web framework/Annotations

- TypeChecked Responses

Type System

- Type Aliases
- Null vs. undefined
- Conditional & Mapped types

Configuration

- Extract Author Information (Name, Email, Web)
- Tag descriptions

Type checked (alternative) Responses

Issue:

- Usually, frameworks throw errors and define a global handler to catch these Errors and transform them into JSON responses
- Documenting these Error responses requires manual annotation (`@Response<Type>(Status, Description)`)
⇒ Potential for issues
- TypeScript does not do type checking on throw/catch
- For backwards compatibility and ease of adoption, the return type of the controller method only reflects the default (2xx) response

Approach:

- Add the ability to inject a type checked responder function that can be invoked and returned

Benefits:

- Type checked Status, Response type and headers
- Framework ensures proper documentation

Example

```
@Route("/orders")
@Security("jwt")
@Tags("Orders")
@Response<UnauthorizedErrorMessage>(401, "Unauthorized")
export class OrdersController extends Controller {
  /**
   * @param paymentRequired Insufficient funds available
   * @param requestBody The Create Order payload
   * @example notFound {"message": "Not Found", "details": "Ship with this id not exist"}
   */
  @Post()
  public async createOrder(
    @Res() badRequest: TsoaResponse<400, ErrorMessage>,
    @Res() paymentRequired: TsoaResponse<402, ErrorMessage>,
    @Res() notFound: TsoaResponse<404, ErrorMessage>,
    @Body()
    requestBody: CreateOrderBody
  ): Promise<Order> {
    notFound(404, { message: "Ship with this id not exist" }); // type checked
    // Implementation
  }
}
```

Type Aliases

Issue:

- Type Aliases are a very common usage of the Type System to Name and reference type constructs
- No support

Approach:

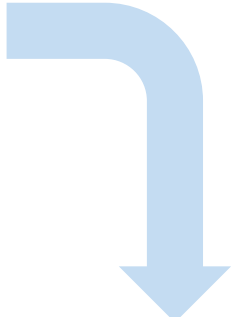
- Add a new naming algorithm
- Add a context (scope) to properly handle Generics

Benefits:

- Reuse types, descriptions, example and format annotations
- Enables advanced type constructs that rely on partial application via type aliases

Example

```
/**
 * Stringified UUIDv4.
 * See [RFC 4112](https://tools.ietf.org/html/rfc4122)
 * @pattern [0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-4[0-9A-Fa-f]{3}-[89ABab][0-9A-Fa-f]{3}-[0-9A-Fa-f]{12}
 * @example "52907745-7672-470e-a803-a2f8feb52944"
 */
export type UUID = string;
```



```
{
  "UUID": {
    "type": "string",
    "example": "52907745-7672-470e-a803-a2f8feb52944",
    "description": "Stringified UUIDv4.\nSee [RFC 4112](https://tools.ietf.org/html/rfc4122)",
    "pattern": "[0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-4[0-9A-Fa-f]{3}-[89ABab][0-9A-Fa-f]{3}-[0-9A-Fa-f]{12}"
  },
}
```

```
UUID string
example: 52907745-7672-470e-a803-a2f8feb52944
pattern: [0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-4[0-9A-Fa-f]{3}-[89ABab][0-9A-Fa-f]{3}-[0-9A-Fa-f]{12}
Stringified UUIDv4. See RFC 4112
```



Conditional & Mapped Types

Issue:

- Modern type systems allow complex constructs to allow more expressive types
- Previously not supported

Approach:

- Instead of complex AST parsing, defer work to the Type Checker

Benefits:

- More flexible approach, allow documentation of advanced types

Theoretical

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

Popularity

- GitHub stars
- Feedback from the community

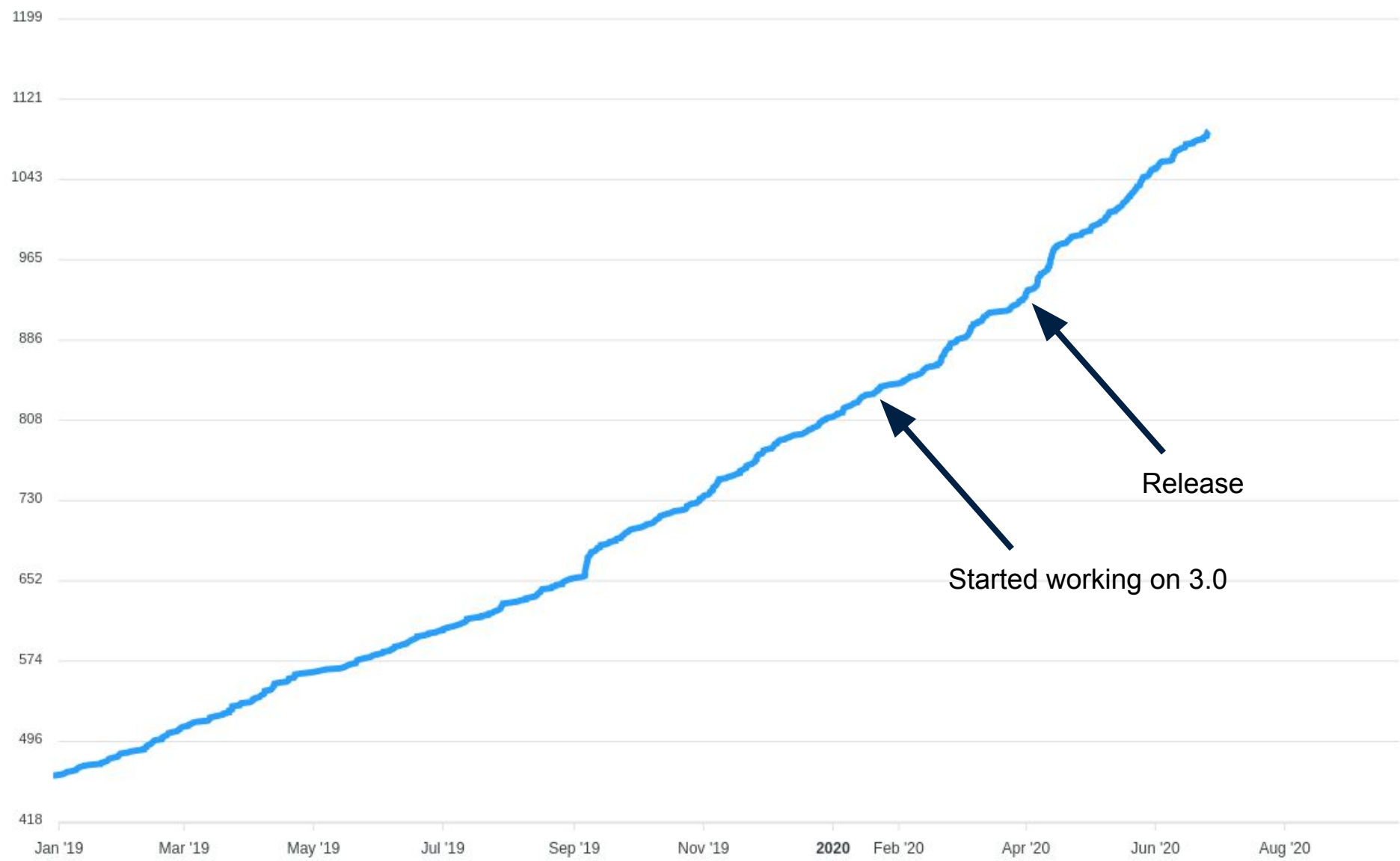
Coding Assignment

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

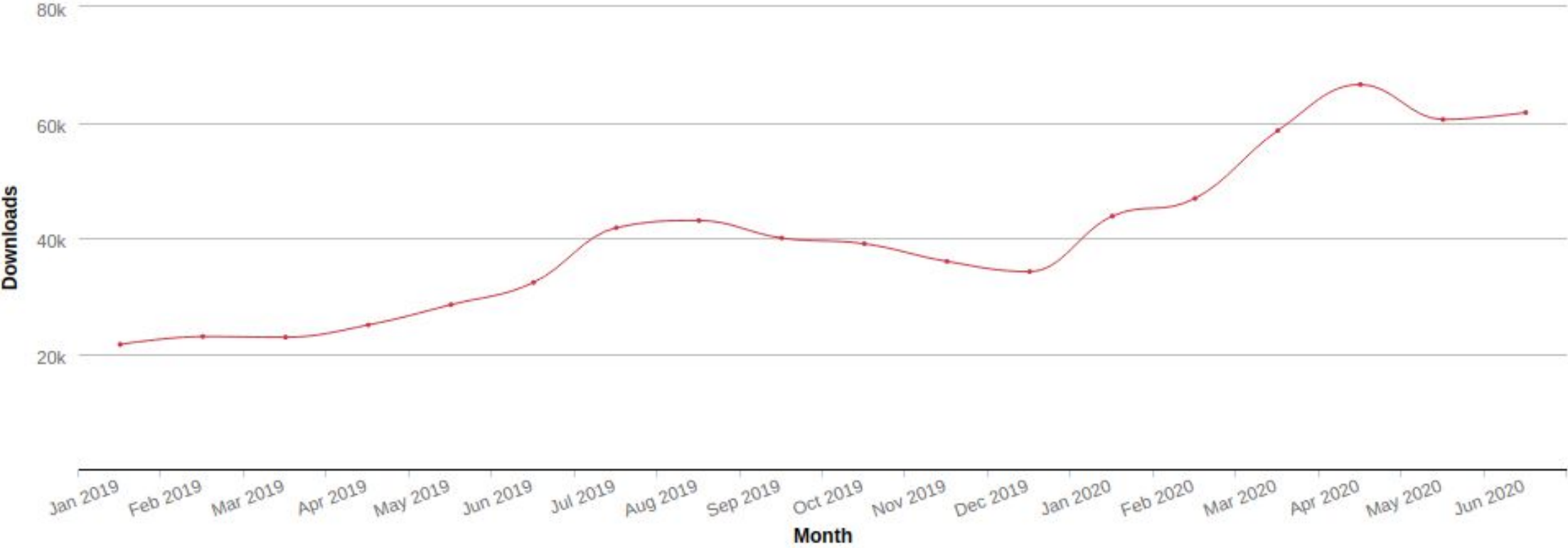
There's No Reason to Write OpenAPI By Hand - Phil Sturgeon, Author (Build APIs You Won't Hate, Surviving Other People's APIs)

- Other frameworks have first-party or third-party support for annotations, which are purely descriptive repetitions of the actual code they sit above at best. **At worst they're just lies.**
- There is a new category of API description integration popping up in some web frameworks which is somewhat like Annotations or DSLs, **but instead of being purely descriptive it's actually powering logic and reducing code, giving you one source of truth.**
- **This is a brand new approach.** Instead of descriptive annotations or comments shoved in as an afterthought, the API framework has been designed around the use of annotations.
- this new approach for making annotations useful is very much closing the gap. If you're going to use a code-first approach, you should absolutely try and find a framework like TSOA to power your API and reduce the chance of mismatches.

RQ3: Evaluation: Results



RQ3: Evaluation: Initial Results



Coding Assignment

- Compare Annotations, Reflection and Macro (AST parsing + type checker) based approach
- 4 participants
- Different familiarity with the frameworks, all have TypeScript experience
- 3 Tasks: Initial Implementation based on requirements, evolution with new requirements, modeling (implementation filled in afterwards)

RQ3: Evaluation

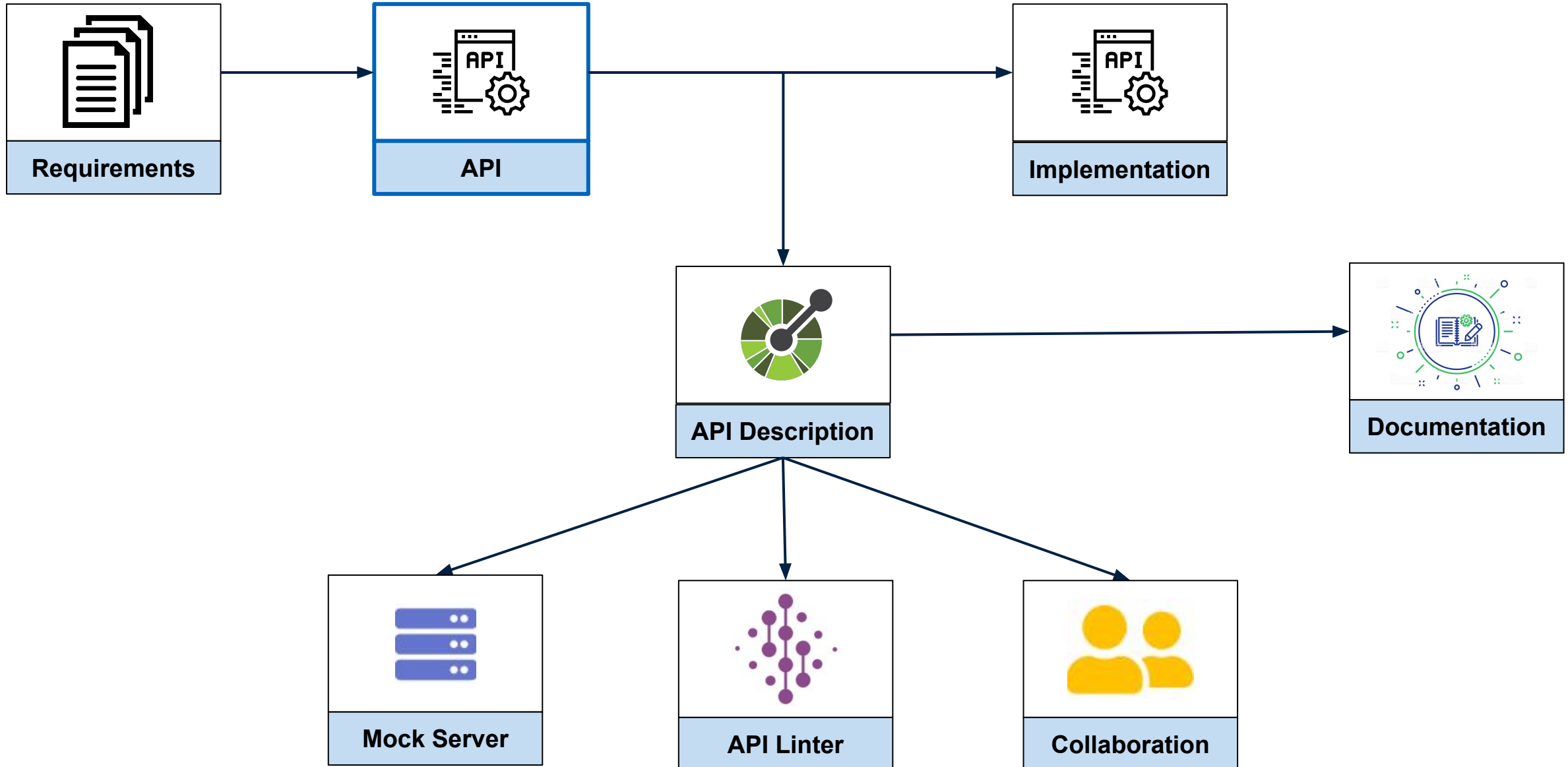
ID	Role	Programming (years)	nest	tsoa	OpenAPI
1	Author	9	4	5	4
2	Director Development	30	1	2	3
3	Senior Developer	8	1	3	1

RQ3: Evaluation: Time

ID	Task	nest	tsoa	OpenAPI
1	1	1:30	1:20	-
1	2	0:15	0:15	-
2	1	1:40	1:25	-
2	2	0:10	0:10	-
3	1	-	~1:35*	-
3	2	-	-	-

RQ3: Evaluation: Correctness

ID	nest	tsoa	OpenAPI
1	14	18,5	-
2	9	17,5	-
3	-	12/15	-



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.