Microservice API Patterns

Dr.-Ing. Daniel Lübke

daniel.luebke@digital-solution-architecture.com

What is an API?

- Application Programming Interface
- Can be anything from a distributed service down to a library interface
- Interface is the master tool of an architect
  - Put something together “Interface“ (English)
  - Cut something apart “Schnittstelle“ (German)
  - Divide & Conquer
  - Separation of Concerns
- APIs are especially important in distributed systems
  - Decouple systems and their lifecycle
  - Allow for independent development and deployment
  - Microservice API

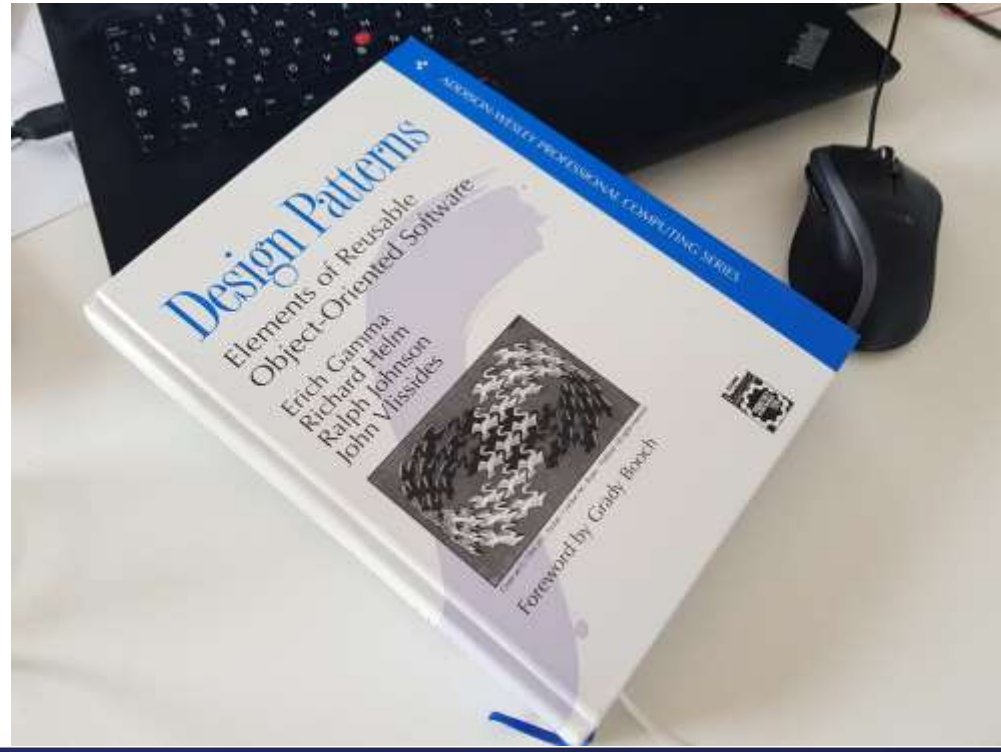
API as a Success Factor

- Often you want to offer a service to be integrated by others
  - e.g. SaaS: messaging, invoicing, …
  - e.g. internal systems that together support a business process
- Digital Transformation requires automation between systems
- Good APIs can lead to better software integration
  - Easy-to-use
  - Better to maintain
  - Cost-efficient

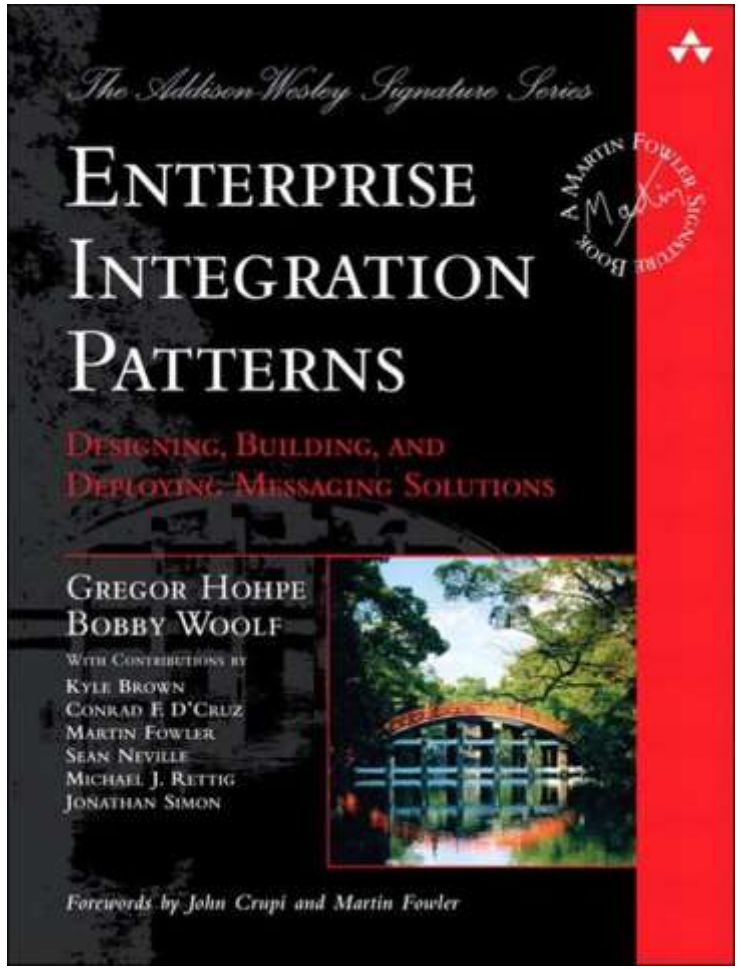
Why Patterns?

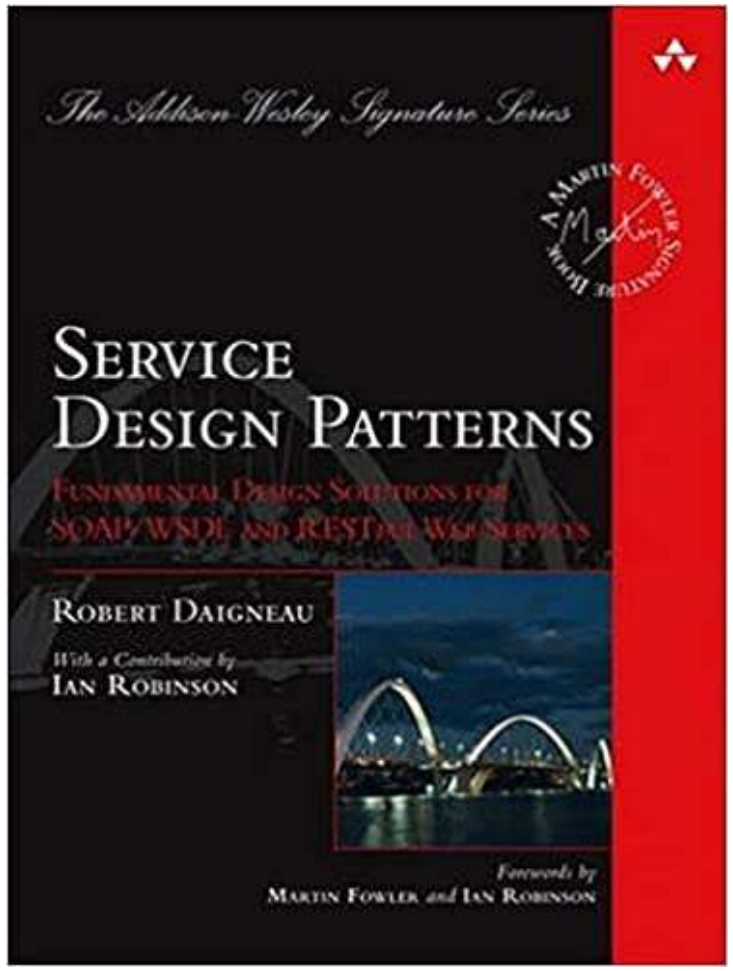
- Patterns collect and document experiences of proven solutions

- Name
- Intent
- Motivation
- Applicability
- Structure
- Consequences (Benefits & Liabilities; Forces)
- Implementation
- Sample
- Known Uses



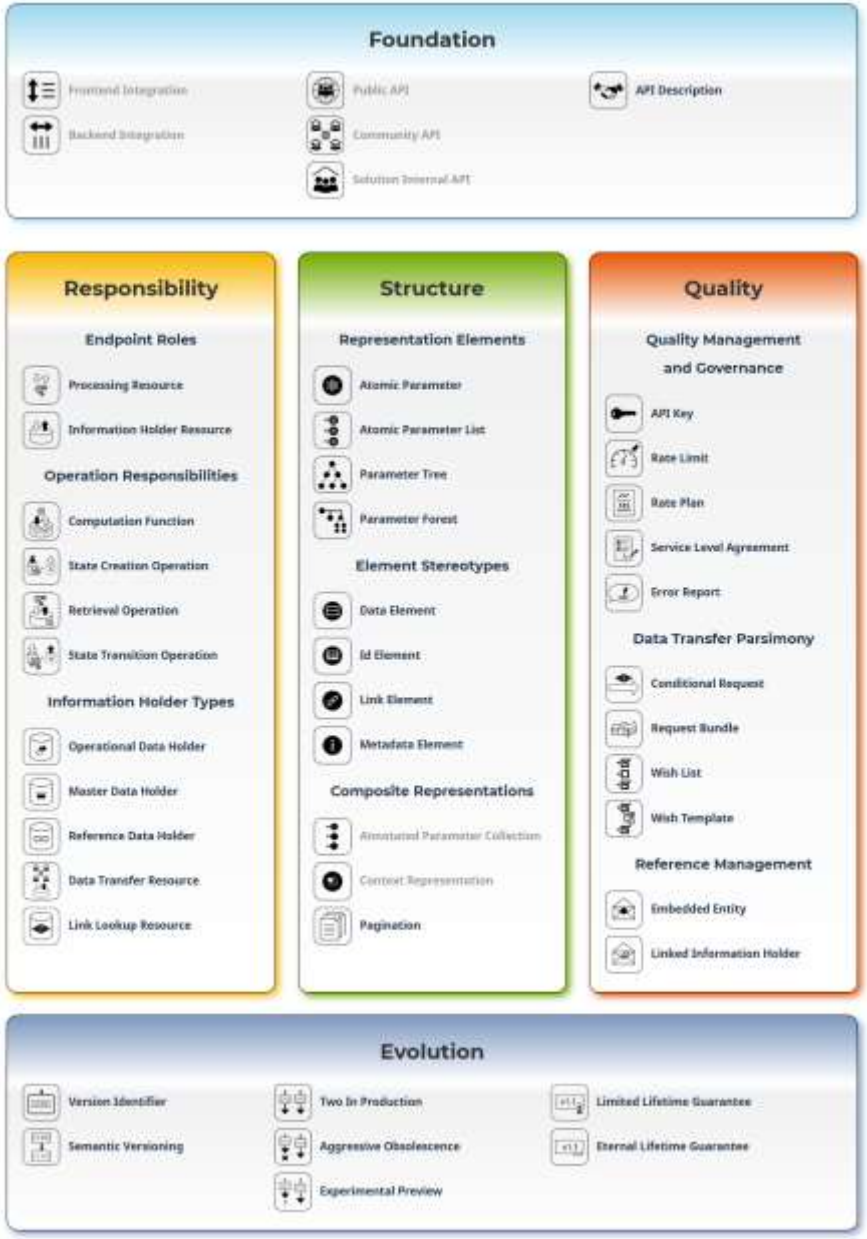
Related Patterns



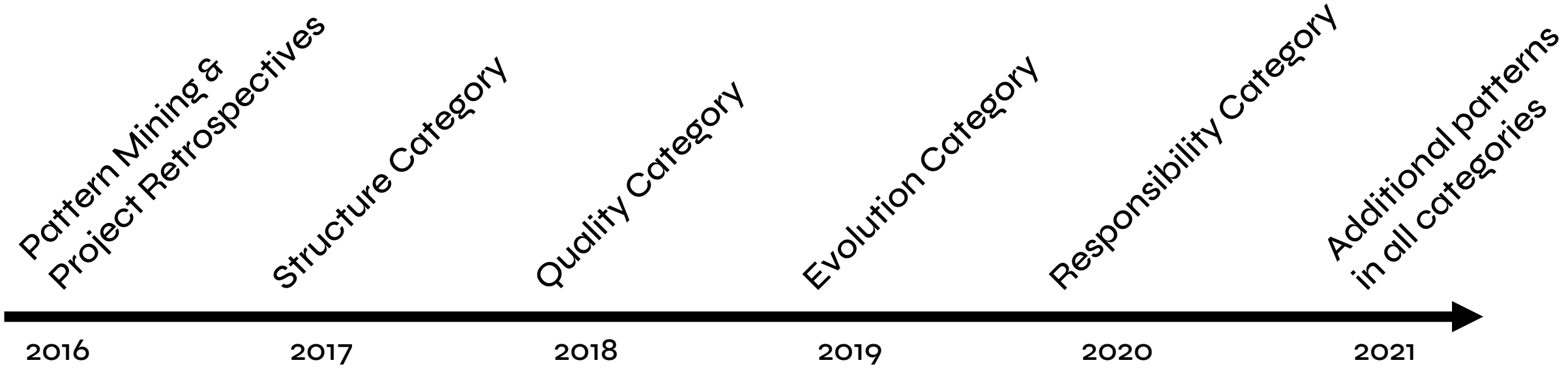


Microservice API Patterns (MAP)

- New but not new
- Distilled solution patterns from many projects
  - Not **invented** by us but **curated** by us
- API-related concepts independent of concrete technology
  - CORBA, SOAP, REST, HTTP/JSON, GraphQL, gRPC, …, MQ, Events, …
- Concerned with how messages can be structured
- Currently 46 patterns in 5 categories



History of MAP



Team

Olaf Zimmermann

Mirko Stocker

Daniel Lübke

Uwe Zdun

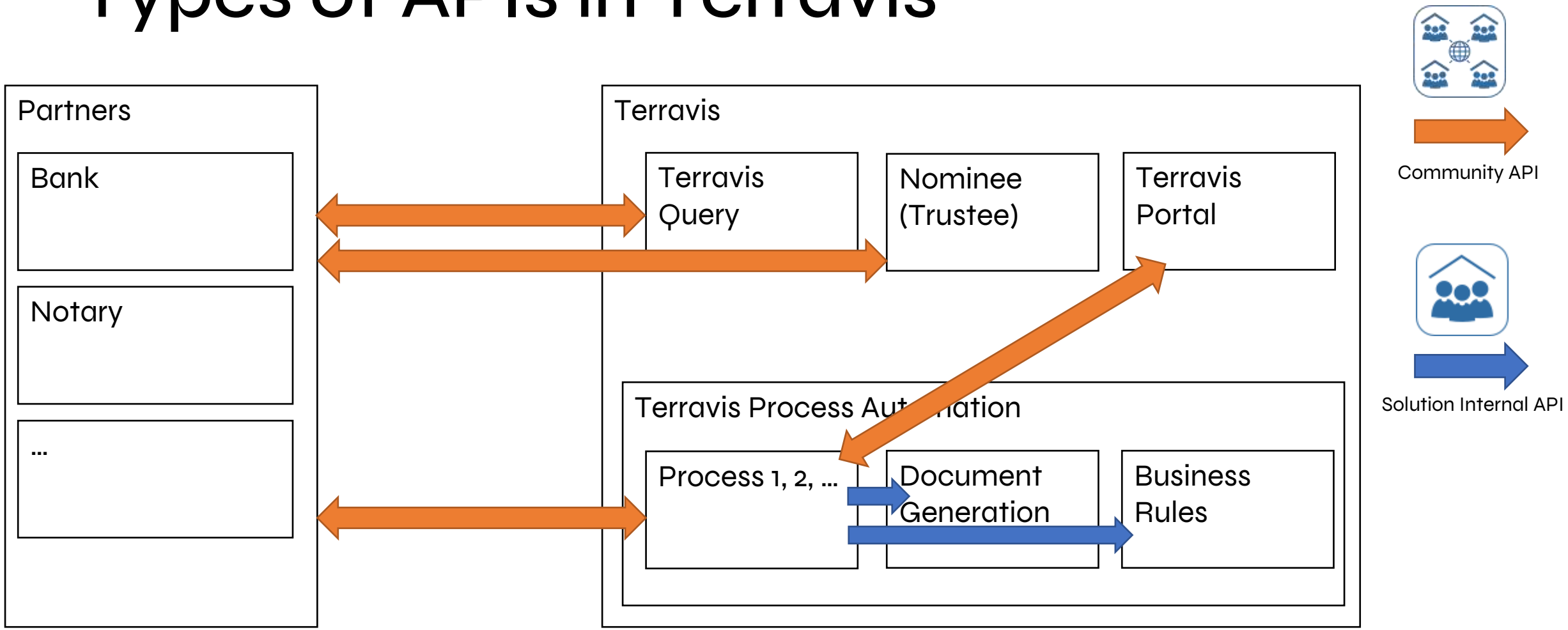
Cesare Pautasso

Example Project - Terravis

- Swiss Large-Scale Process Integration Platform eGovernment and Land Registry Context

- Covers Land Register Businesses
  - Land Registries (100s)
  - Notaries (100s)
  - Banks (100s)
  - Trustee (1)
  - Surveyor (100s)
  - Interbank Clearing (1)

Types of APIs in Terravis



Evolution (Terravis)

- Terravis uses following MAP Evolution Patterns in its general service design guidelines for its Community APIs:

Two in Production

- Two (major) versions of an API are offered in parallel to allow for easier transitions of partners to newer versions (avoid coordinated, fixed-dates deployments)

Version Identifier

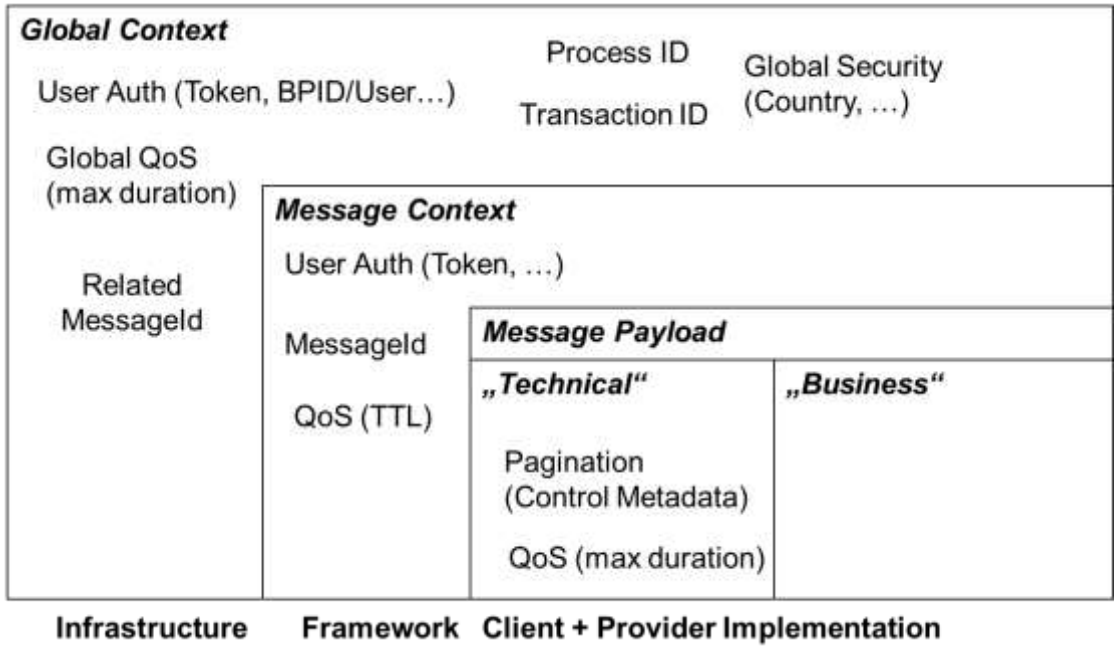
- Identify different versions of APIs and their capabilities

Semantic Versioning

- Indicate compatibility between versions

Security and Context Propagation

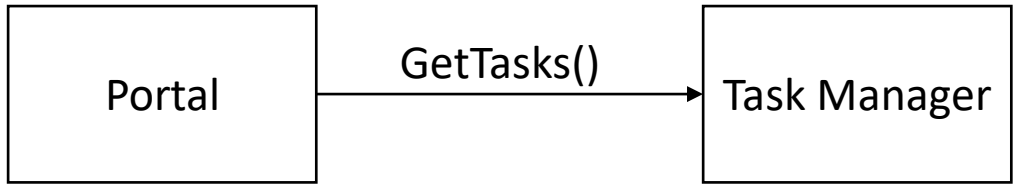
- Terravis implements the Context Representation pattern by definining a standardized header for all request messages
  - Partner
  - Partner System Location
  - User
  - Message ID
  - Business Process ID



Machine Readable Errors

- Terravis implements the Error Report pattern by defininingstandardized fault structures for all operations
  - Error Code
  - Error Objects
  - English fallback error message
  - Related Message ID
  - Business Process ID

Task Query Service

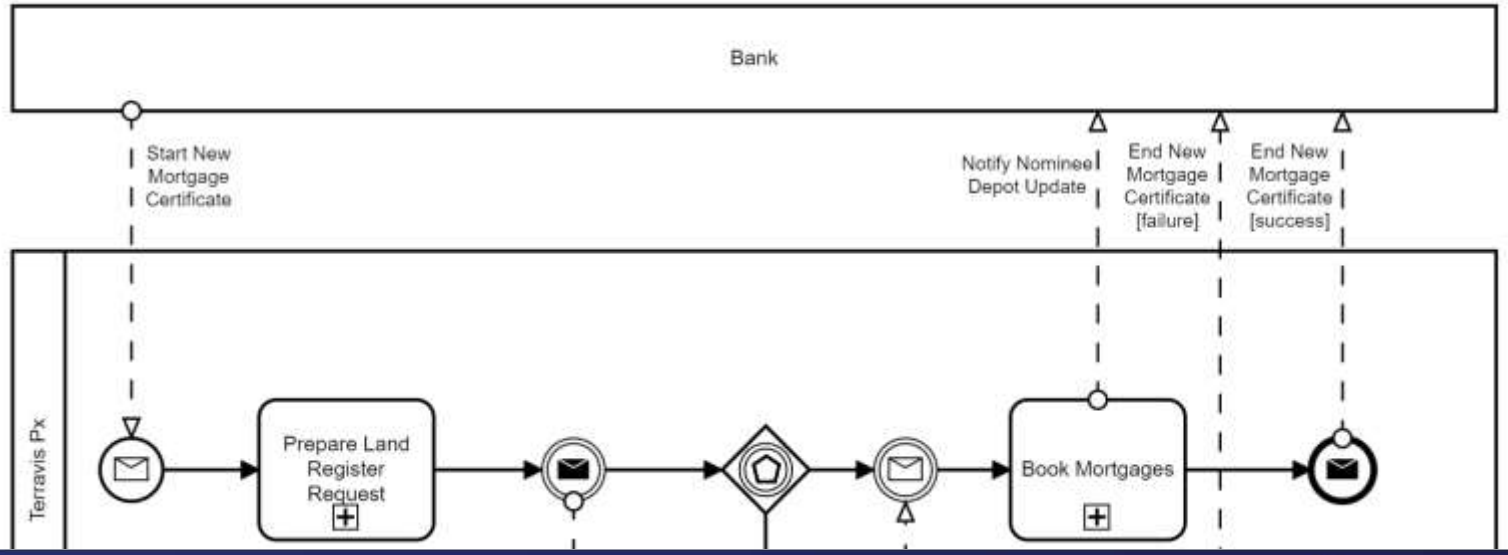



```
getTasks
- TaskName [0..1]
- ProcessType [0..1]
- …
- Pagination
    - FromIndex
    - PageSize
```

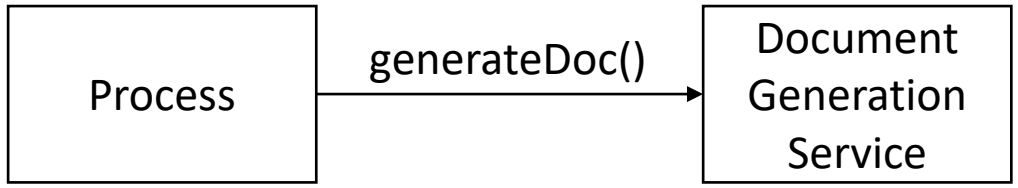
For reducing the response message size of this Solution Internal API, Terravis uses the Pagination pattern for “chunking” the response into manageable pieces

Process New Mortgage

- Processes are initiated by State Creation operations (named Start…)
- Processes are moved forward by State Transition Operations
- Distinction clear:
  notifyNomineeDepotUpdate()



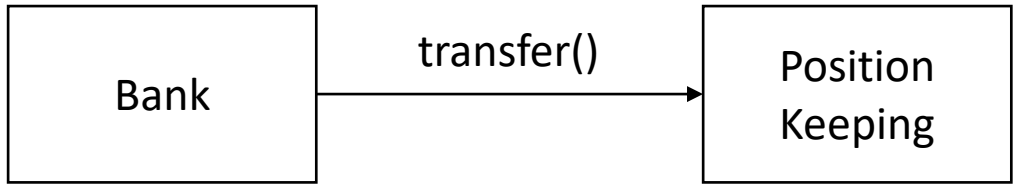
Document Generation



Document Generation is a Solution Internal API that are implemented as a Computation Function:

- No state change
- All necessary data is delivered via the API

Bulk Mortgage Transfer



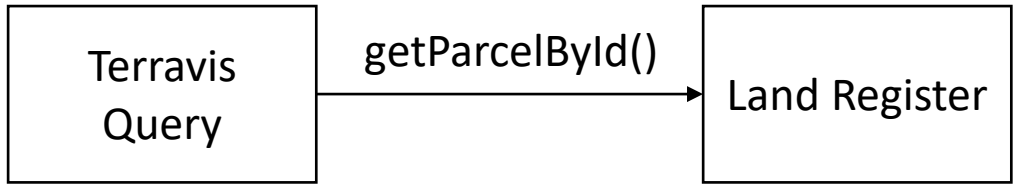

```
transfer
- Asset [1..*]
    - AssetId
    - FromDepot
    - ToDepot
```

```
transferResult
- Result [1..*]
    - AssetId
    - Status
```

Bulk Mortgage Transfer is a Community API that is implemented as a State Transition Operation utilizing the Request Bundle pattern to improve efficiency by combining multiple, independent requests into one.

Land Register Parcel Query

```
getParcelById
- EGRID
- QueryType
- IncludeHistory
```



Parcel Query is a Community API, which is a Master Data API that implements the Wishlist pattern to reduce unnecessary data in the response message

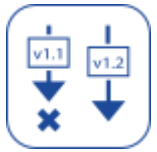
Other MAP Patterns

Conditional Request:

- Only returns a full response if data is changed

API Key

- System authentication based on a shared secret

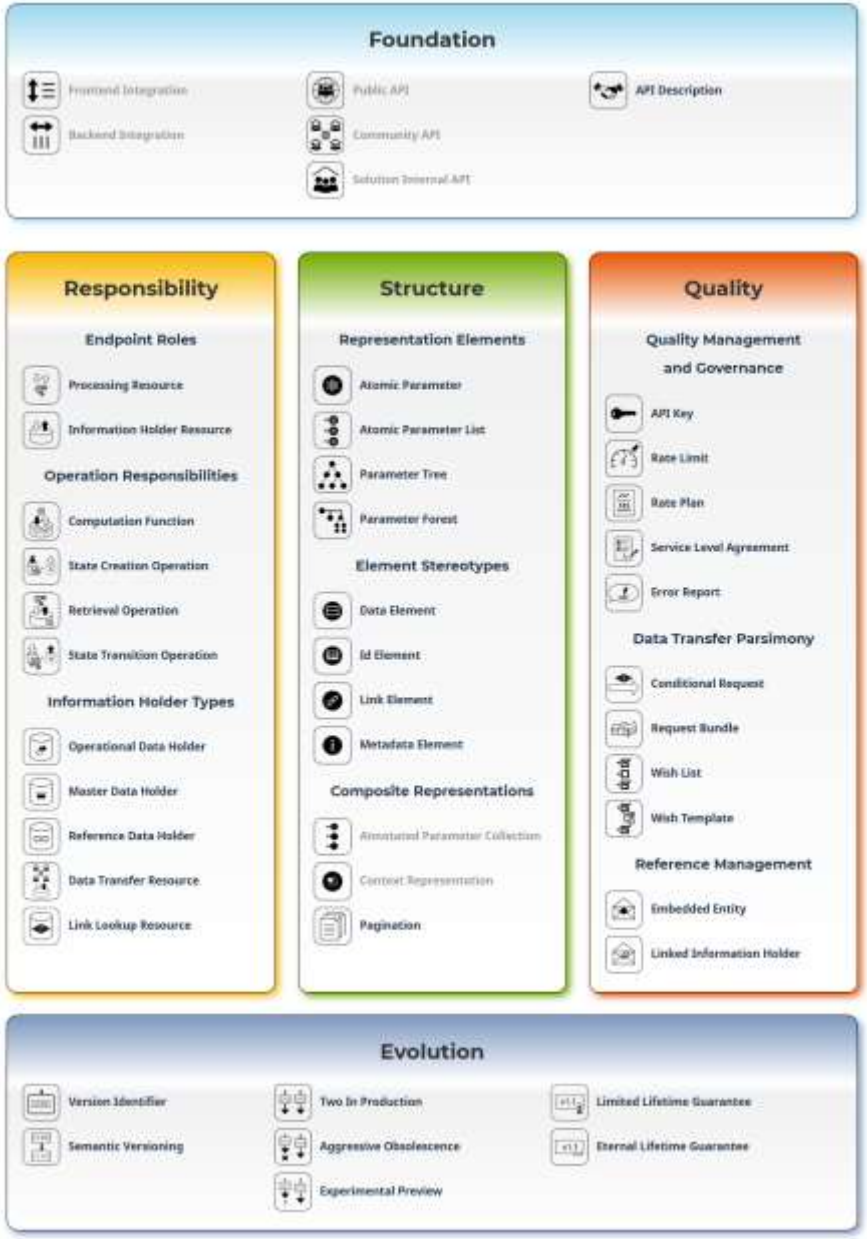


Aggressive Obsolescence

- Aggressively abandoning old API versions with much cost and effort required by the consumer

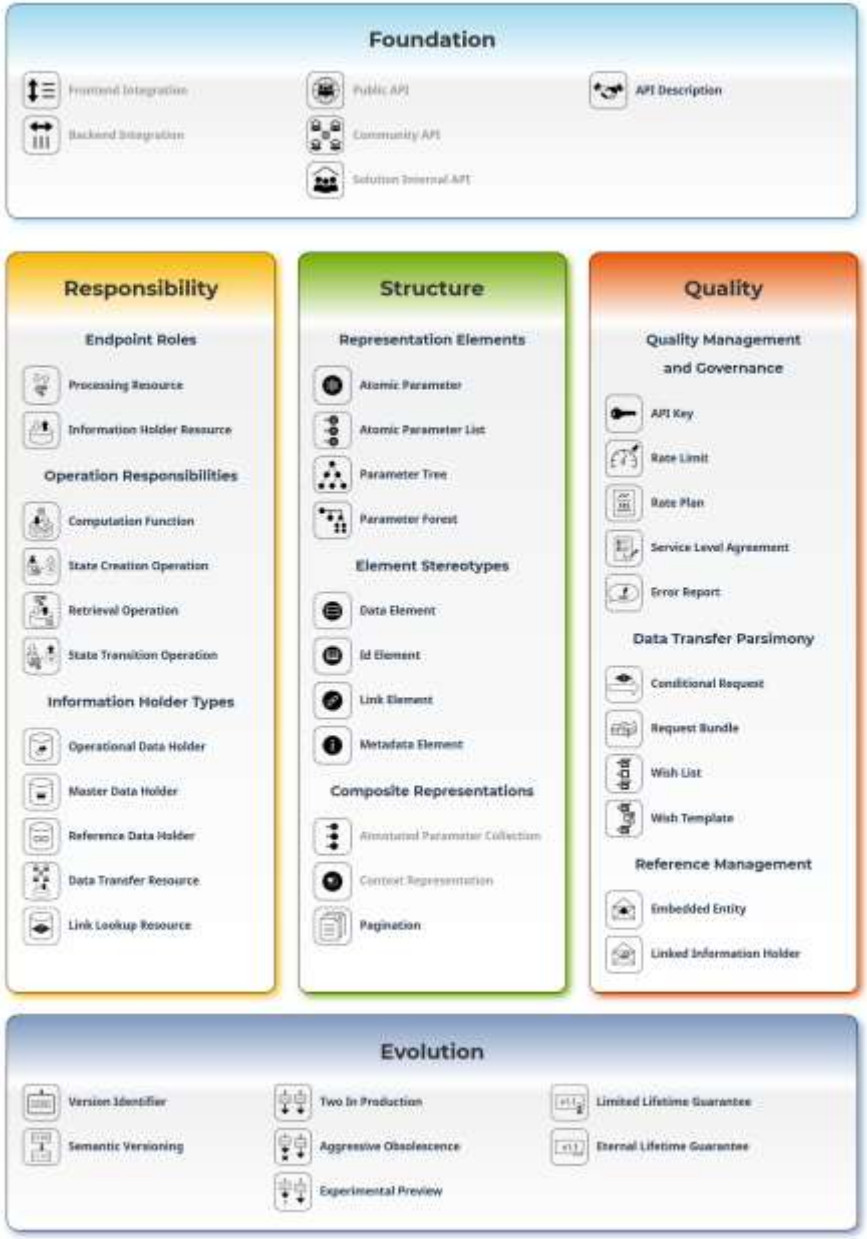
Eternal Lifetime Guarantee

- Endless backwards-compatibility with much cost and effort required by the provider



Where to look?

- Microservice API Patterns are
  - the foundation to a shared language concerning API design
  - a resource for necessary and possible design decisions
  - a library for implementation hints
- Visit https://www.microservice-api-patterns.org
- Happily use our content!
- Let us know your known uses or comments
- **Let's us establish a common language in API design!**



Thank you for your attention! Any Questions?
daniel.luebke@digital-solution-architecture.com