Tim Voigt

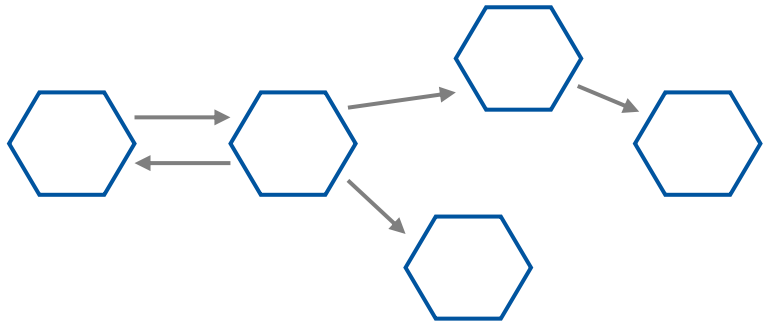
A classification approach of information needs for anti-pattern detection in microservice applications

Bachelor Thesis

Advisor: Alex Sabau

SWC Software Construction | RWTH AACHEN UNIVERSITY

Background

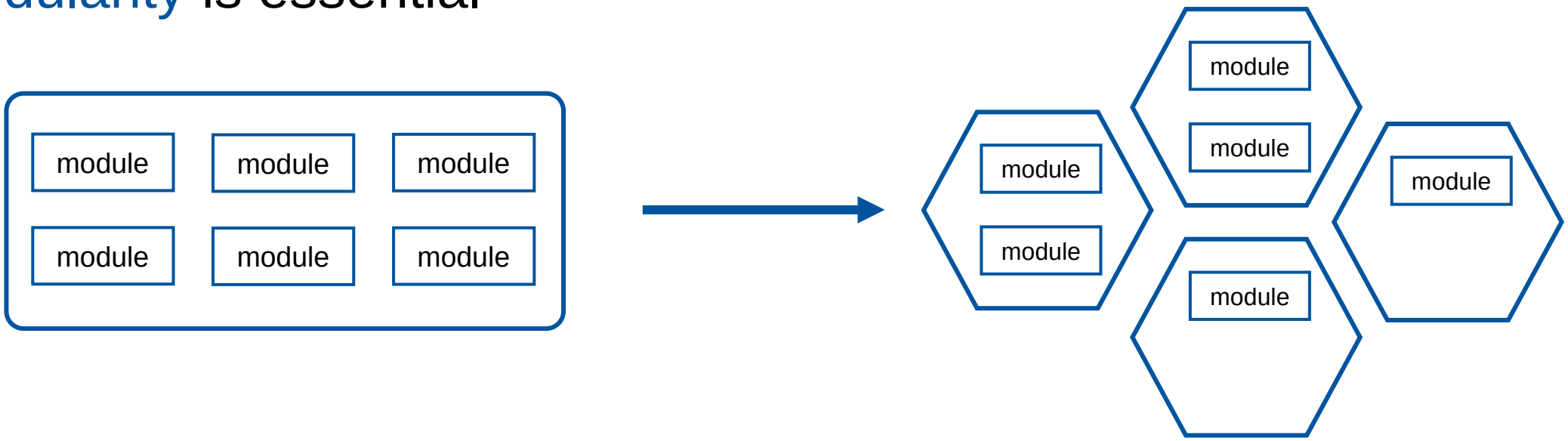


Microservice Architectures (MSAs)

Anti-patterns

Background: Microservice Architectures (I/III)

- Alternative to monolithic architectures
- Encapsulate business logic in many small components
- Modularity is essential

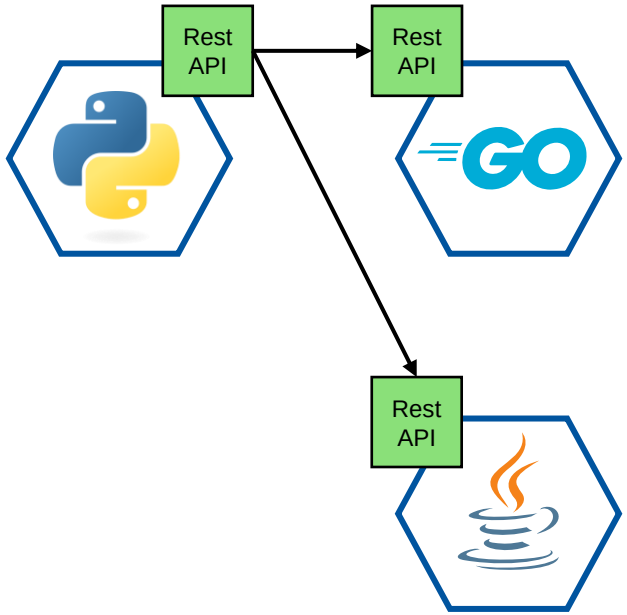


Monolithic application

Microservice architecture

Background: Microservice Architectures (II/III)

- Set of characteristics (excerpt) [Nad+16]
  - Small in size
  - Messaging enabled
  - Bounded by contexts
  - Autonomously developed
  - Built and released with automated processes
- Encourage polyglot programming practices



Background: Microservice Architectures (III/III)

- Benefits
  - Enable autonomy of teams
  - Easy experimenting and adoption of solutions
  - Better fault isolation
  - Easily maintainable

- Drawbacks
  - Finding right set of services is challenging
  - No pre-defined way to decompose
  - Careful coordination of inter-team feature deployment

Background: Anti-patterns (I/II)

- Common solution to a recurring problem [Koe95]
  - Risk of being counterproductive
- Can impair quality of an application
- Impact on evolvability and maintainability [Sal19]
- Terms anti-pattern and smell are interchangeable

Background: Anti-patterns (II/II)

- Architectural anti-patterns
  - "Connections among source files that violate design principles and impact bug-proneness and change-proneness" [Mo+21]
- Code anti-patterns
  - Certain structures in code that indicate trouble [Fow18]
  - Can be fixed by refactoring

- Most commercial tools focus on code-related anti-patterns [Mo+21]
  - Negligence of architectural anti-patterns

Motivation (I/III)

> “If bridge building were like programming, halfway through we’d find out that the far bank was now 50 meters farther out, that it was actually mud rather than granite, and that rather than building a footbridge we were instead building a road bridge.”
>
> – Sam Newman, Building Microservices: Designing Fine-Grained Systems

- Scope may change throughout development
- Challenging to design perfect architecture
- Flaws likely end up in application

Motivation (II/III)

- Designing MSA is not easy
  - Migration even more challenging [Ric18]
- Microservices add complexity
  - Increased inter-service communication
  - Central logging infrastructure
  - Must handle partial failure of other microservices
- MSA likely grows over time

Motivation (III/III)

- Important to reduce presence of anti-patterns
  - Improve quality of MSA
- Straightforward and automated detection
  - Repeated execution in short intervals
- Automated detection can lower manual effort
- Help organizations profit from MSAs

Problem Statement

- Little research on quantifying and analyzing the quality of MSAs

- **RQ1:** How can the information need for anti-pattern detection in microservice applications be meaningfully structured and classified?

- **RQ2:** Which methods can be used to obtain the necessary information for anti-pattern detection in microservice applications?

- **RQ3:** Which anti-patterns in microservice applications can be detected automatically?

Outline

1. Introduction
2. Related Work
3. Concepts
   1. Information Sources
   2. Information Requirements
4. Detection Process Example
5. Closing
   1. Discussion
   2. Future Work
   3. Summary

Related Work

Non-Microservice Oriented

Microservice Oriented

Microservice Anti-pattern catalog

- Common microservice anti-patterns by Taibi et al. [TLP20]
  - Catalog of 27 anti-patterns

Microservices Anti-Pattern	Also proposed by	Answers		Perceived Harmfulness (0-10)
		#	%	
Hardcoded Endpoints	[6][10]	10	37	8
Wrong Cuts	[6]	15	56	8
Cyclic Dependency	[6]	5	19	7

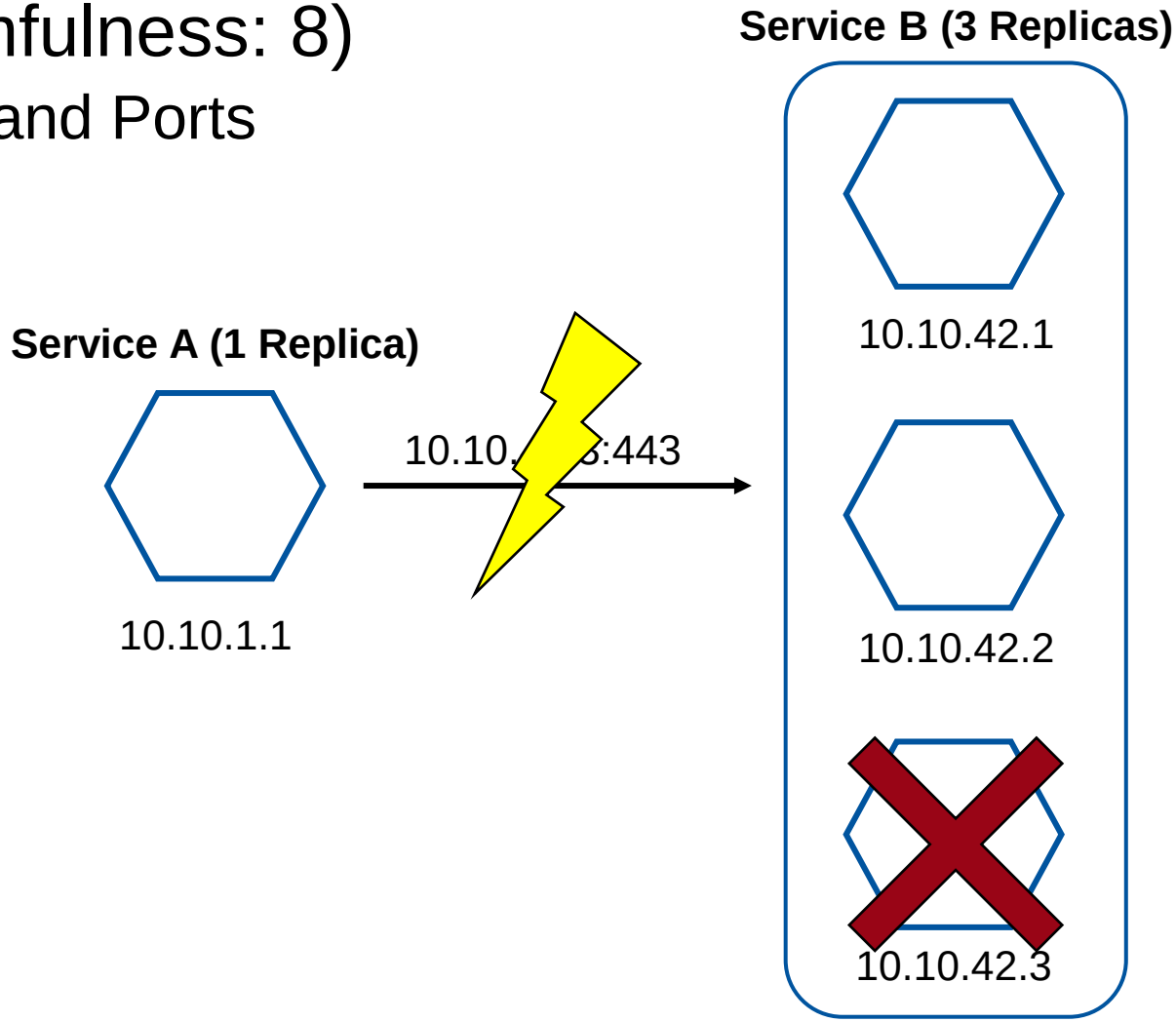
  - Taxonomy of 20 anti-patterns

Microservices Anti-Pattern	Description (Desc) / Detection (Det)	Problem it may cause (P) / Adopted Solutions (S)
Cyclic Dependency	**Desc:** A cyclic chain of calls between microservices **Det:** Existence of cycles of calls between microservices. E.g., A calls B, B calls C, and C calls back A.	**P:** Microservices involved in a cyclic dependency can be hard to maintain or reuse in isolation. **S:** Refinement of the cycles according to their shape [15] and application of an API-Gateway pattern [4].

  - Suggest harmfulness on a 10-point Likert scale

Most Harmful Anti-patterns [TLP20]

- Hardcoded Endpoints (Harmfulness: 8)
  - Also known as Hardcoded IPs and Ports
  - Defeats advantages of MSAs

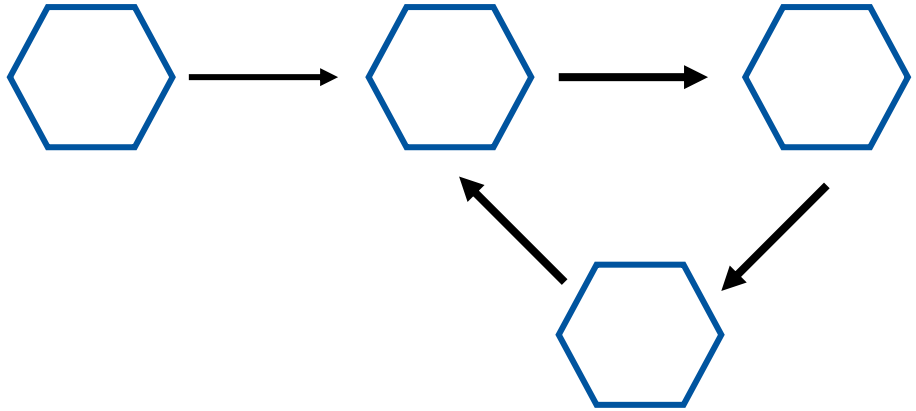


Most Harmful Anti-patterns [TLP20]

- Hardcoded Endpoints (Harmfulness: 8)
  - Also known as Hardcoded IPs and Ports
  - Defeats advantages of MSAs
- Wrong Cuts (Harmfulness: 8)
  - Incorrect decomposition of business capabilities
  - Very hard to detect automatically

Most Harmful Anti-patterns [TLP20]

- Hardcoded Endpoints (Harmfulness: 8)
  - Also known as Hardcoded IPs and Ports
  - Defeats advantages of MSAs
- Wrong Cuts (Harmfulness: 8)
  - Incorrect decomposition of business capabilities
  - Very hard to detect automatically
- Cyclic Dependency (Harmfulness: 7)
  - Loop of dependencies
  - Microservices are not independent
  - Increased maintenance



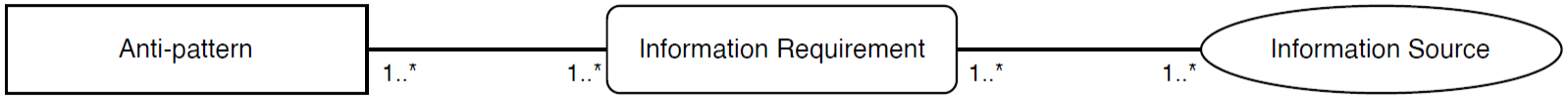
Concepts - Definition

Information Requirement

Describes the necessary information required by an algorithm or human to perform anti-pattern detection in MSAs.

Information Source

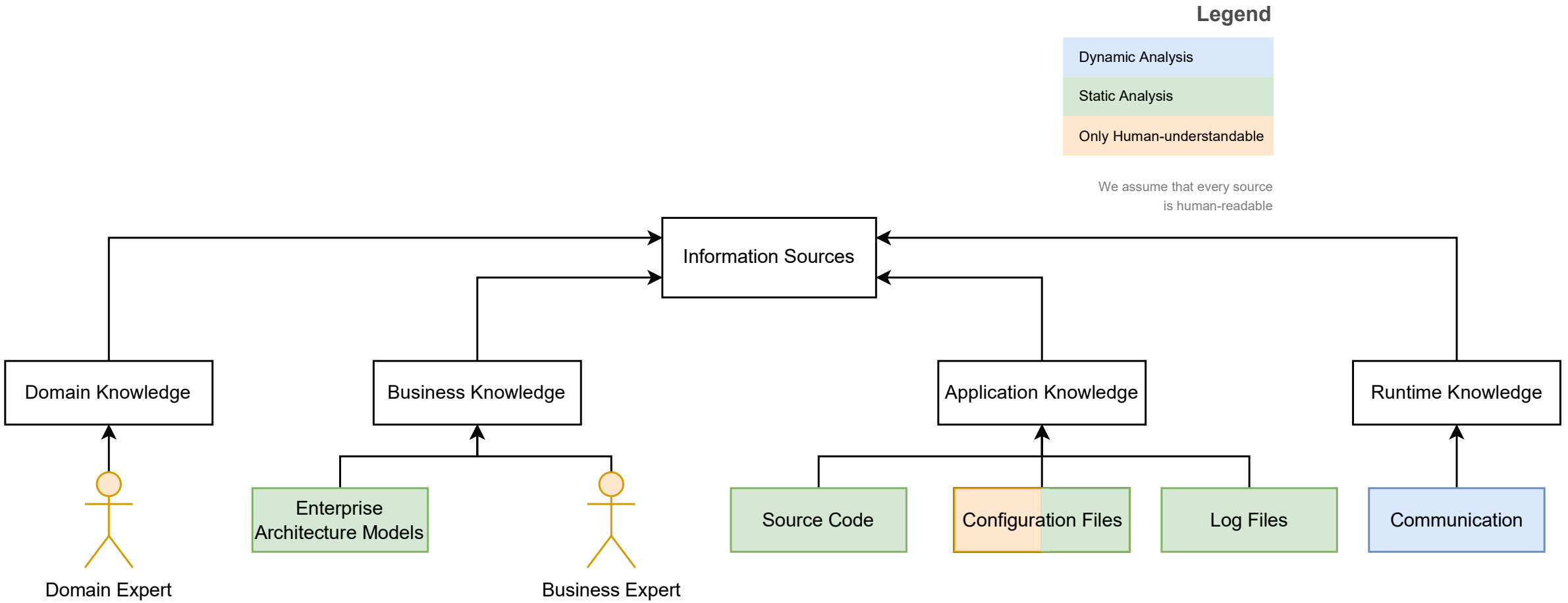
Contains information which satisfies one or more information requirements.
Can be any artifact from a microservice application which is either human- or machine-readable.
Information can be extracted without human involvement from a machine-readable information source using various methods – like static analysis.



Information Sources (Knowledge Classes)

- Domain knowledge
  - Domain: Subject area to which a program is applied [Eva04]
  - Domain experts are crucial for extraction
- Business knowledge
  - Total set of business concepts, their organizing connections, and the business rules upon which the existence of the business depends [Ros]
- Application knowledge
  - Information about microservice applications and its artifacts
- Runtime knowledge
  - Information exclusively available from running systems

Information Sources (Knowledge Classes)

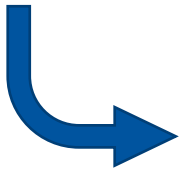


Information Requirements

- Thesis proposes 20 information requirements
  - Investigated anti-pattern detection methods
- Only satisfied information requirements can be used
- How to satisfy an information requirement?
  - Underlying requirement must be fulfilled
  - Irrelevant whether human- or machine-readable
- Information necessary to satisfy is specified in thesis

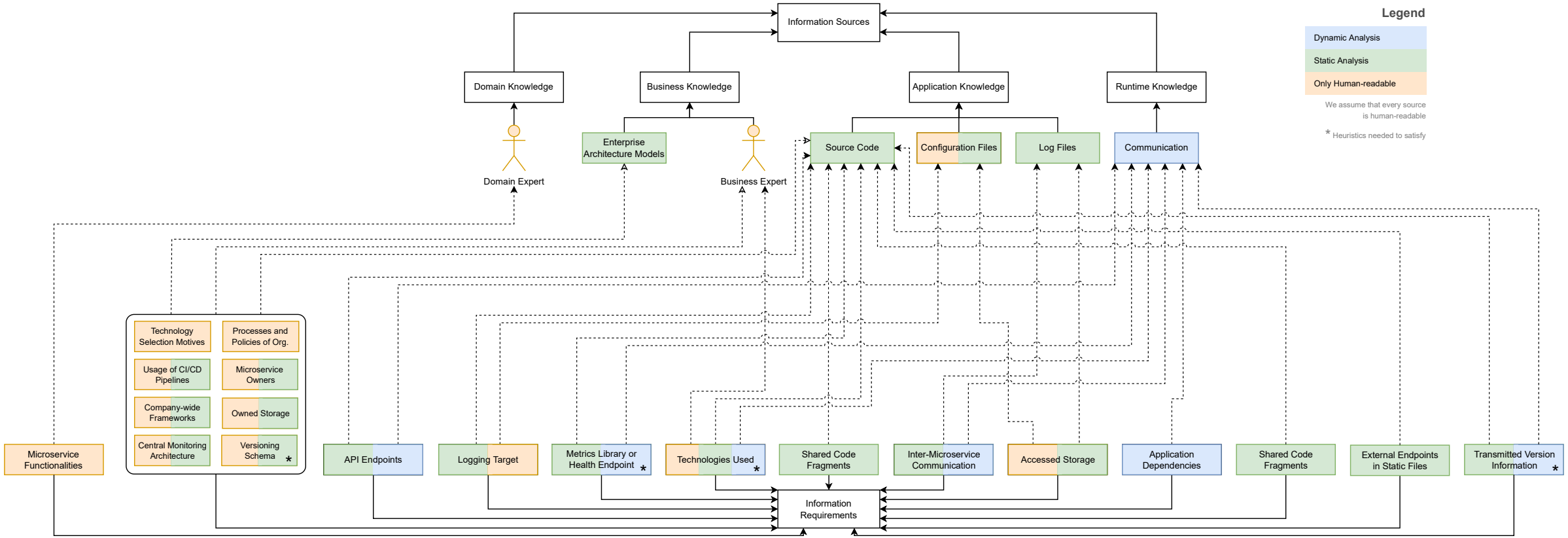
Information Requirements

- 10 out of 20 focus on technical information
- Eight anti-patterns have more than one information requirement
  - True for opposite direction as well
- Anti-patterns with two or more information requirements likely to have at least one business-related information requirement

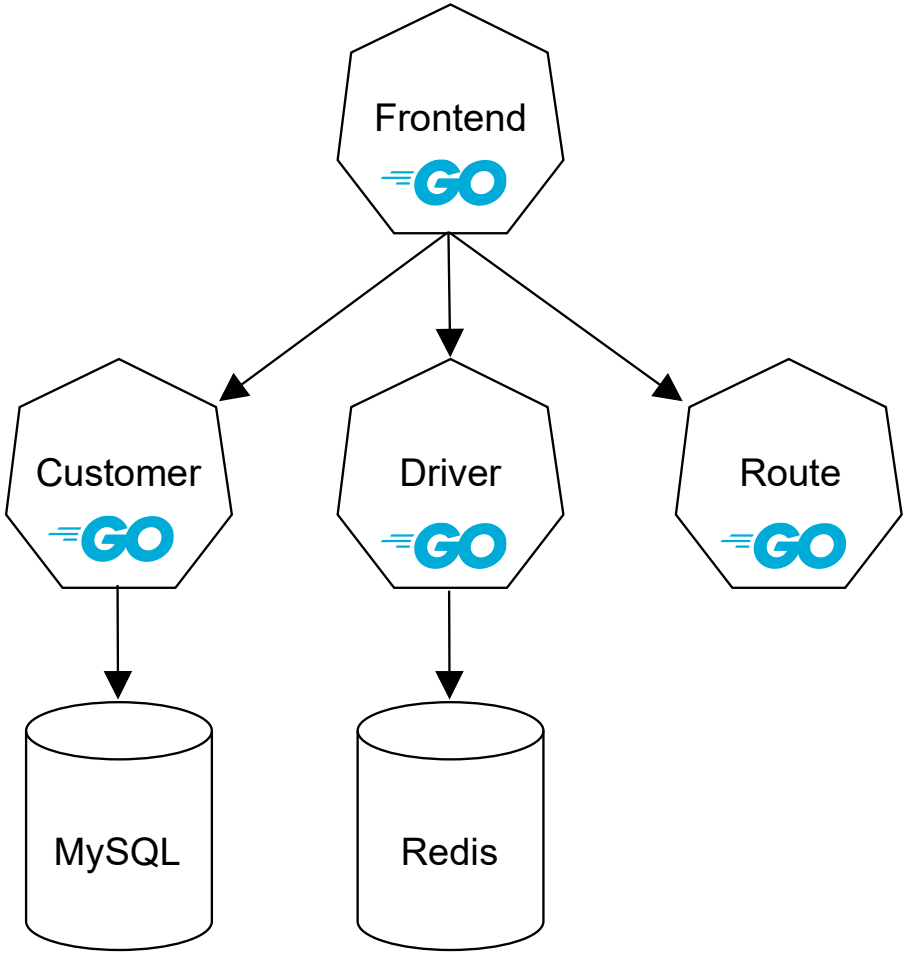


Some anti-patterns require more information than others

Information Source and Requirement Relationship



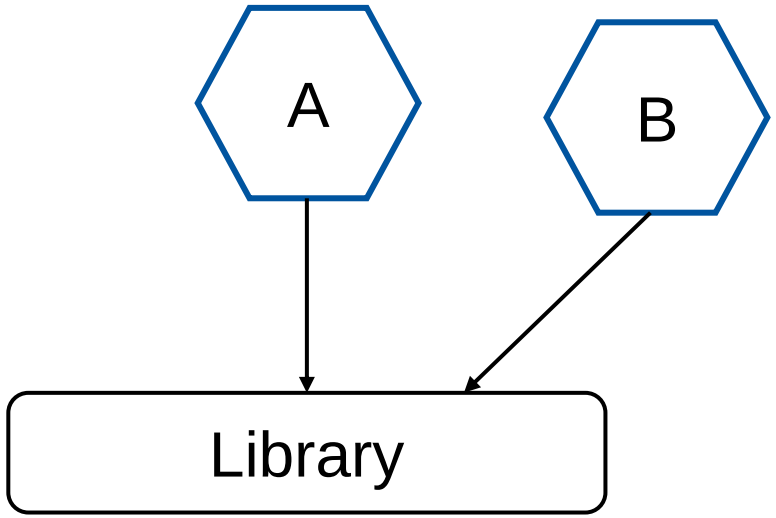
Detection Process Example



Detection Process Example

Detection Process Example

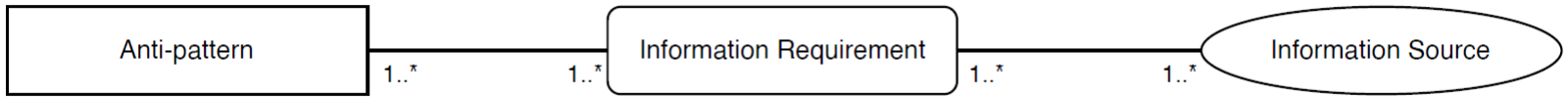
- Shared libraries anti-pattern



- Tightly coupled microservices
- Loss of independence
- Coordination efforts increase

Detection Process Example

Dependencies of each microservice



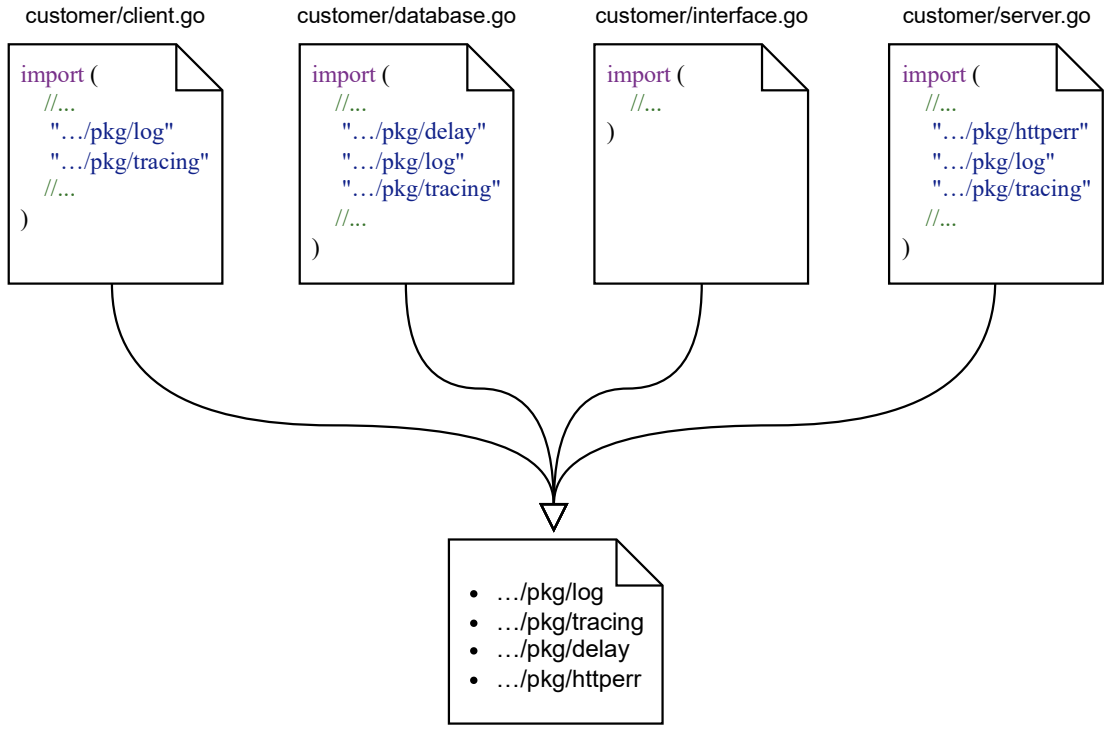
Detection Process Example


```
1  // Copyright (c) 2019 The Jaeger Authors.
2  // Copyright (c) 2017 Uber Technologies , Inc.
3
4  package customer
5
6  import (
7   "context"
8   "fmt"
9   "net/http"
10
11  "github.com/opentracing -contrib/go -stdlib/nethttp"
12  "github.com/opentracing/opentracing -go"
13  "go.uber.org/zap"
14
15   "github.com/jaegertracing/jaeger/examples/hotrod/pkg/log"
16   "github.com/jaegertracing/jaeger/examples/hotrod/pkg/tracing"
17  )
18
19  //...
```

Import declarations

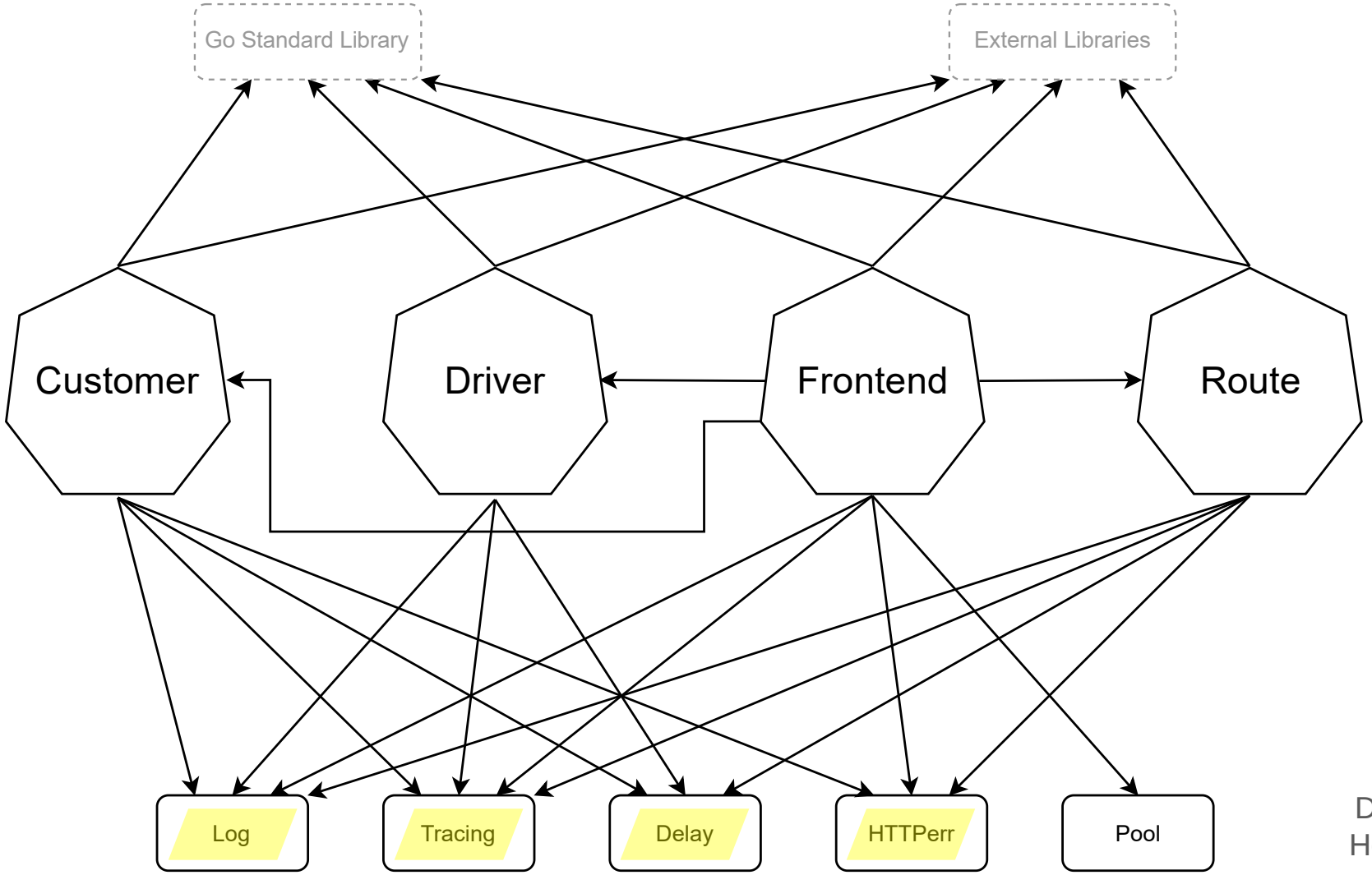
Detection Process Example

- Import declarations from source files
- Multiple processing steps
  - Analysis of import declarations for every source file
  - Merge results on a per-microservice basis
- Static analysis methods feasible



In-house dependencies of *customer* microservice

Detection Process Example



Dependency Graph of Hot R.O.D. application

Summary of exemplary detection process

- Defined information requirements
- Explored suitable information sources
- Proposed extraction methods
- Performed anti-pattern detection

Shared library anti-pattern is present!

Discussion

- RQ1: Proposed concepts of information requirements and sources
  - Four categories of information
  - One or more information requirements for every anti-pattern
- RQ2: Proposed extraction methods
  - Based on information sources
  - Not implemented or validated yet
- RQ3: Information sources linked to information requirements
  - Static- and dynamic analysis can be performed automatically
  - Linked to anti-patterns

Future Work

- Evaluation of results
  - Interviewing a set of experts on chosen information requirements
  - Case study
- Extending information requirements and extraction methods
- Automated processing of human-readable information
  - Business- and domain-knowledge
  - Could increase automatically detectable anti-patterns
- Automatically resolving anti-patterns
  - Complex problem
  - Holistic view of microservice application necessary

Summary

Problem Statement

- Little research on quantifying and analyzing the quality of MSAs
- **RQ1:** How can the information need for anti-pattern detection in microservice applications be meaningfully structured and classified?
- **RQ2:** Which methods can be used to obtain the necessary information for anti-pattern detection in microservice applications?
- **RQ3:** Which anti-patterns in microservice applications can be detected automatically?

Concepts - Definition

Information Requirement

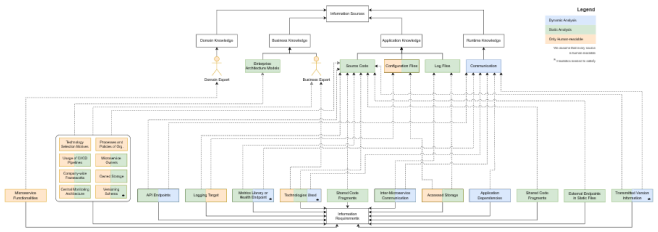
Describes the necessary information required by an algorithm or human to perform anti-pattern detection in MSAs.

Information Source

Contains information which satisfies one or more information requirements.
Can be any artifact from a microservice application which is either human- or machine-readable.
Information can be extracted without human involvement from a machine-readable information source using various methods – like static analysis.

Detection Process Example

Information Source and Requirement Relationship



Discussion

- RQ1: Proposed concepts of information requirements and sources
  - Four categories of information
  - One or more information requirements for every anti-pattern
- RQ2: Proposed extraction methods
  - Based on information sources
  - Not implemented or validated yet
- RQ3: Information sources linked to information requirements
  - Static- and dynamic analysis can be performed automatically
  - Linked to anti-patterns

Future Work

- Evaluation of results
  - Interviewing a set of experts on chosen information requirements
  - Case study
- Extending information requirements and extraction methods
- Automated processing of human-readable information
  - Business- and domain-knowledge
  - Could increase automatically detectable anti-patterns
- Automatically resolving anti-patterns
  - Complex problem
  - Holistic view of microservice application necessary

Software Construction | RWTH AACHEN UNIVERSITY