Red Hat Research Day Europe 2022

Brno

Czech Republic

September 15th, 2022

Using static analysis for microservices

Tomas Cerny
Baylor University (USA)

Baylor University

ABOUT

- Tomas Cerny
  - Assistant professor at Baylor University
    - last year in my tenure
  - Research
    - Cloud-native systems
    - Static and dynamic analysis
- Long-term collaboration with Red Hat Research
  - Over 35 Baylor students on research projects with Red Hat

https://cs.baylor.edu/~cerny

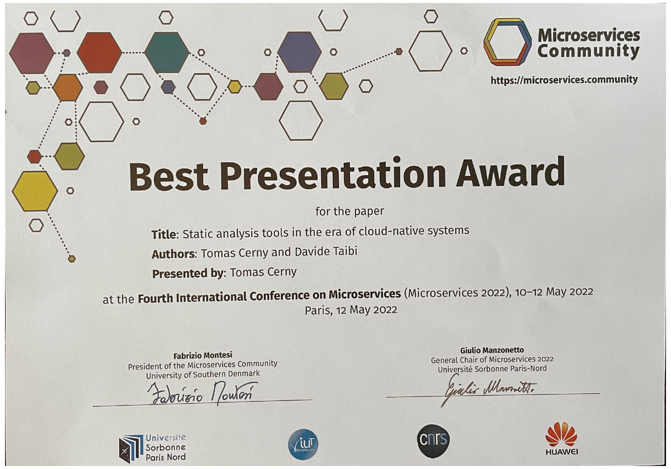
NSF award #1854049 IRES Track I: U.S.-Czech Student Research Experience on Software Test Automation and Quality Assurance.

Red Hat Research

OUTLINE

Static analysis for microservices

- Why do we want this?
- What is the challenge?
- How do we address it
- What do we get out of this?



Microservices Community

https://microservices.community

Best Presentation Award

for the paper

Title: Static analysis tools in the era of cloud-native systems

Authors: Tomas Cerny and Davide Taibi

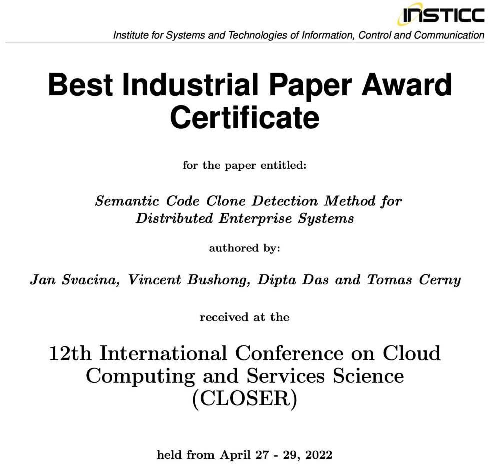
Presented by: Tomas Cerny

at the **Fourth International Conference on Microservices** (Microservices 2022), 10–12 May 2022
Paris, 12 May 2022

Fabrizio Montesi
President of the Microservices Community
University of Southern Denmark

Giulio Manzonetto
General Chair of Microservices 2022
Université Sorbonne Paris-Nord

Université Sorbonne Paris Nord · iUT Villetaneuse · cnrs · HUAWEI



INSTICC
Institute for Systems and Technologies of Information, Control and Communication

Best Industrial Paper Award Certificate

for the paper entitled:

Semantic Code Clone Detection Method for Distributed Enterprise Systems

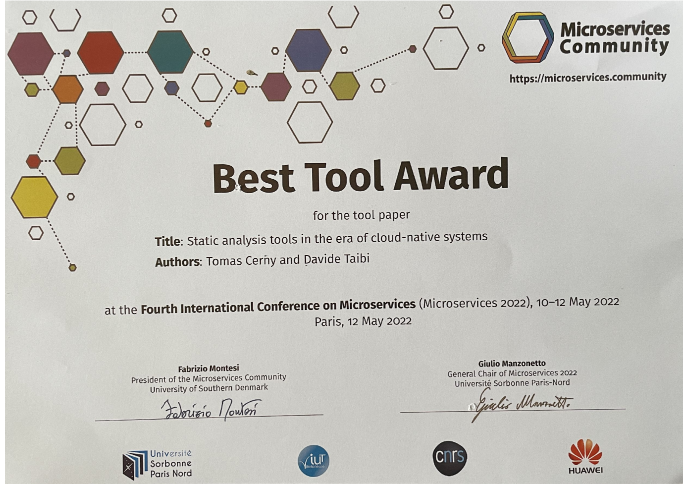
authored by:

Jan Svacina, Vincent Bushong, Dipta Das and Tomas Cerny

received at the

12th International Conference on Cloud Computing and Services Science (CLOSER)

held from April 27 - 29, 2022



Microservices Community

https://microservices.community

Best Tool Award

for the tool paper

Title: Static analysis tools in the era of cloud-native systems

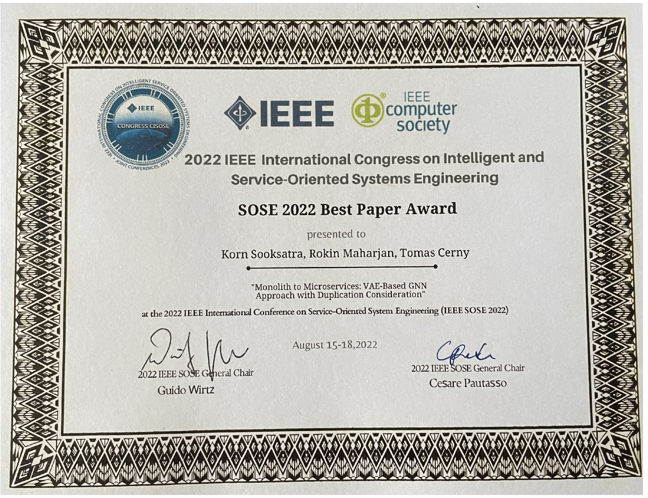
Authors: Tomas Cerny and Davide Taibi

at the **Fourth International Conference on Microservices** (Microservices 2022), 10–12 May 2022
Paris, 12 May 2022

Fabrizio Montesi
President of the Microservices Community
University of Southern Denmark

Giulio Manzonetto
General Chair of Microservices 2022
Université Sorbonne Paris-Nord

Université Sorbonne Paris Nord · iUT Villetaneuse · cnrs · HUAWEI



IEEE · IEEE computer society · Congress CISOSE

2022 IEEE International Congress on Intelligent and Service-Oriented Systems Engineering

SOSE 2022 Best Paper Award

presented to

Korn Sooksatra, Rokin Maharjan, Tomas Cerny

"Monolith to Microservices: VAE-Based GNN Approach with Duplication Consideration"

at the 2022 IEEE International Conference on Service-Oriented System Engineering (IEEE SOSE 2022)

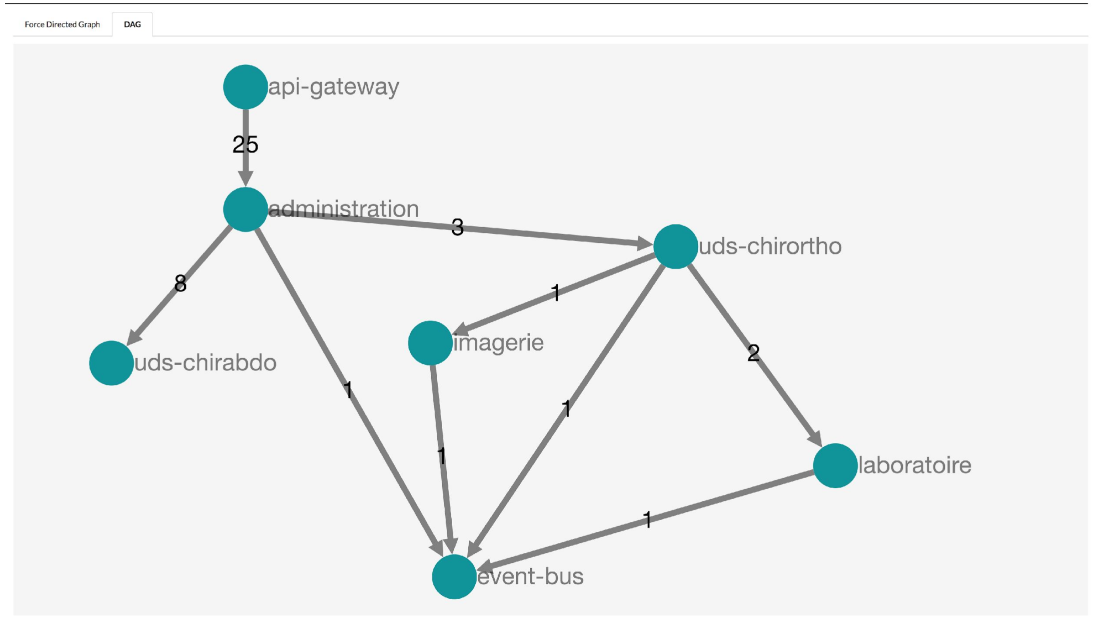
August 15-18,2022

2022 IEEE SOSE General Chair
Guido Wirtz

2022 IEEE SOSE General Chair
Cesare Pautasso

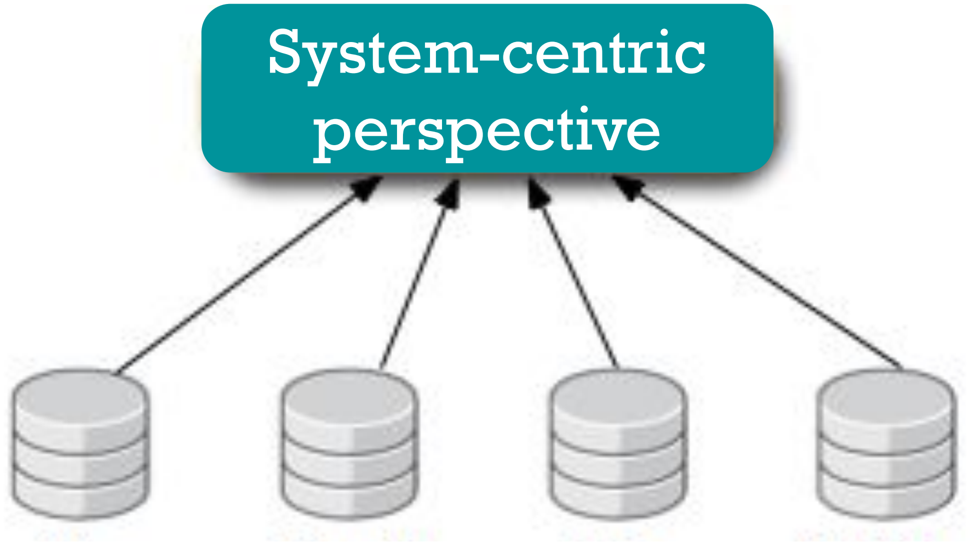
MOTIVATION

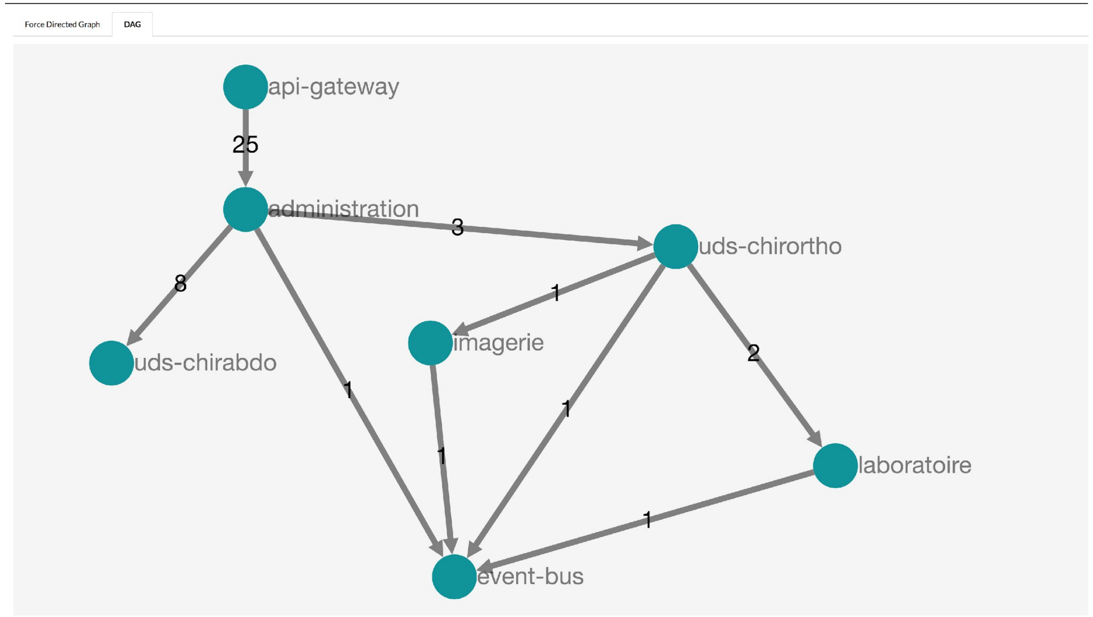
- In the context of Microservices, do you recognize what this is?
- Why do we need it, and how did we get it?



WHY DO WE GET THIS GRAPH?

- To understand how the holistic system works,
  - assess the dependencies, avoid a ripple effect, etc.
- ..get a centralized system perspective,
- .. reason about the system

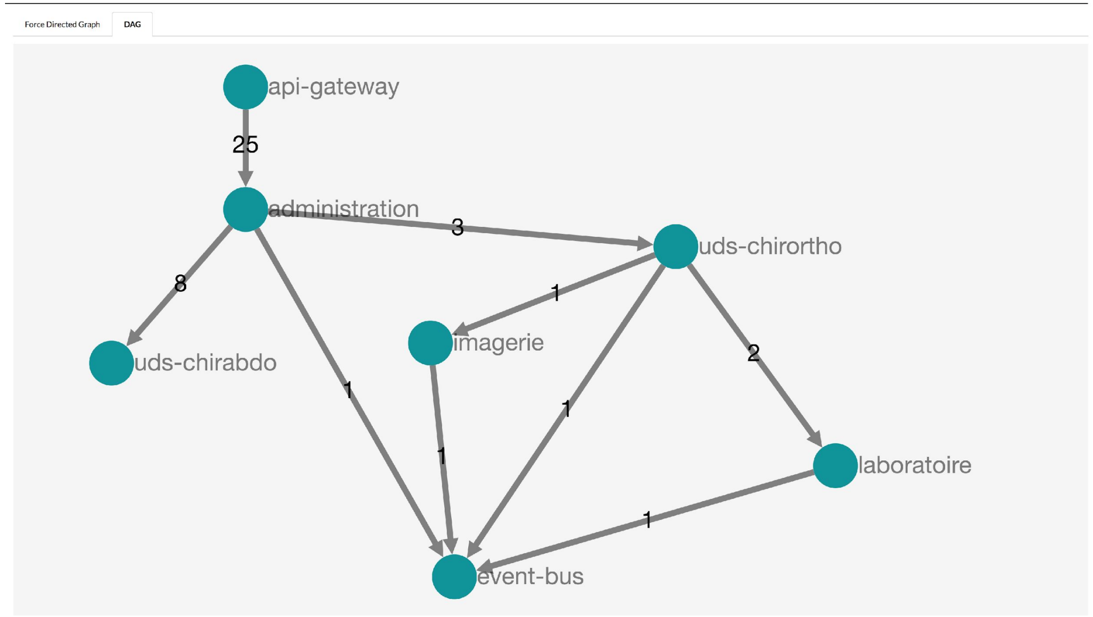




HOW DO WE GET THIS GRAPH?

- We currently obtain this graph through dynamic system analysis
  - DevOps
  - OpenTelemetry, etc.
  - Event trace with a Correlation ID
  - We extract this from a deployed system

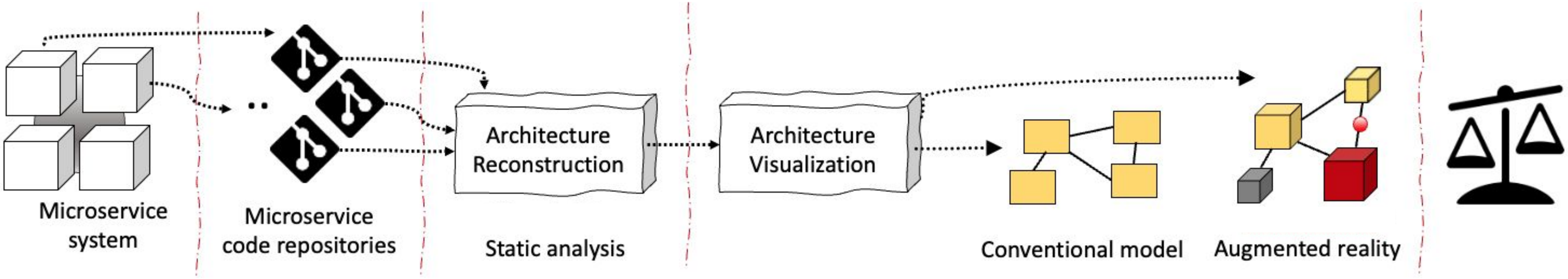
- Challenges
  - Traffic or tests needed
  - Delay: development vs. deployed system
  - No direct feedback to developers
  - Incomplete system coverage



COULD WE GET THIS GRAPH USING STATIC ANALYSIS?

- There would be no delay
  - DevOps/Developers
- No traffic needed
- More convenient for developers
- Code-change impact analysis

- **Challenges**
  - No tools for decentralized systems
  - Language heterogeneity
  - How do we connect it?


Microservice system
Microservice code repositories
Architecture Reconstruction
Static analysis
Architecture Visualization
Conventional model
Augmented reality

CHALLENGES IN MICROSERVICES

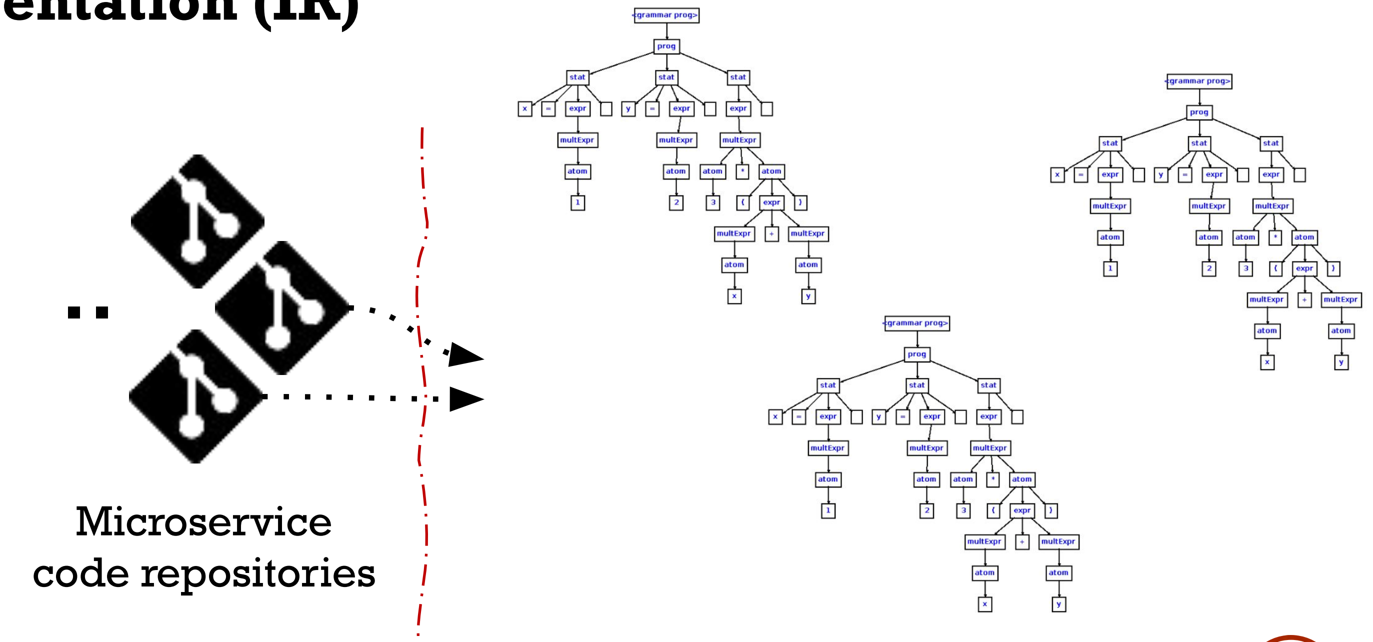
- Documentation quickly becomes outdated
- Poor dependency overview
  - No centralized view of the system
- Ripple effect
  - System-part change impacts other system parts
- Too complex systems
  - Heterogeneity | team coordination | the architecture itself
    - Descriptive vs. prescriptive architecture – are they the same?
- Separation of Duty : Architects / Dev Teams / DevOps

Industry practices and challenges for the evolvability assurance of microservices, DOI: 10.1007/s10664-021-09999-9

BASICS OF STATIC ANALYSIS

- Parsing code / codebase / bytecode / mining software repository
- Abstract Syntax Tree / Control Flow Graph / Program Dependence Graph
  - Or other **Intermediate Representation (IR)**

- Applications
  - Pattern matching
    - Code style
    - Bad smells
    - Vulnerability checks
    - Technical debt
  - Other reasoning



Microservice code repositories

CHALLENGE: CULTURAL CLASH
STATIC ANALYSIS IN MICROSERVICES

- Static analysis
  - **Plain ‘low-level’ code** used
    - Low-level intermediate representation
  - Limited to a **single codebase**
    - Processing linearly and combining results does not work
      - Dependencies between microservices
  - **Language-specific**

- Microservices
  - Decentralized systems with **decentralized codebases** per Microservice
  - **Heterogeneous** system parts

INITIAL THOUGHTS

- How to represent a system?
  - Intermediate Representation?
  - Graph?

- Support for a single platform?
  - Yes, but…? Actually, no multiple..

- How do we merge microservices? Based on their dependencies?
  - If so, can we generalize it across platforms?

- We assessed common testbenches and frameworks for microservices
  - **Observation**: Common best practices are cross-platform applicable

WHAT DOES MICROSERVICE CODE LOOK LIKE?

Observation: code uses high-level structures -> components

- endpoints/controllers,
- services,
- repositories,
- **remote calls**,
- messaging,
- entities,
- data-transfer objects
- ..

What static analysis deals with?

- Low-level code, no components
- **Can we analyze code for components?**

WHAT ARE THE BEST PRACTICES?

- i.e., 12-Factor app tells us how to design, build and operate **cloud-native systems**.

- What else can we conclude?
  - Many patterns for design, communication, robustness, resilience, routing, discovery, authorization/authentication, ..

- How do we use best practices in the source code?
  - **Components and high-level structures**!
    - Nice separation of concerns

WHAT ARE THE BEST PRACTICES?

- i.e., 12-Factor app tells us how to design, build and operate **cloud-native systems**.

- What else can we conclude?
  - Many patterns for design, communication, robustness, resilience, routing, discovery, authorization/authentication, ..

- How do we use best practices in the source code?
  - **Components and high-level structures**!
    - Nice separation of concerns

Do static analysis tools operate with such components?

No!

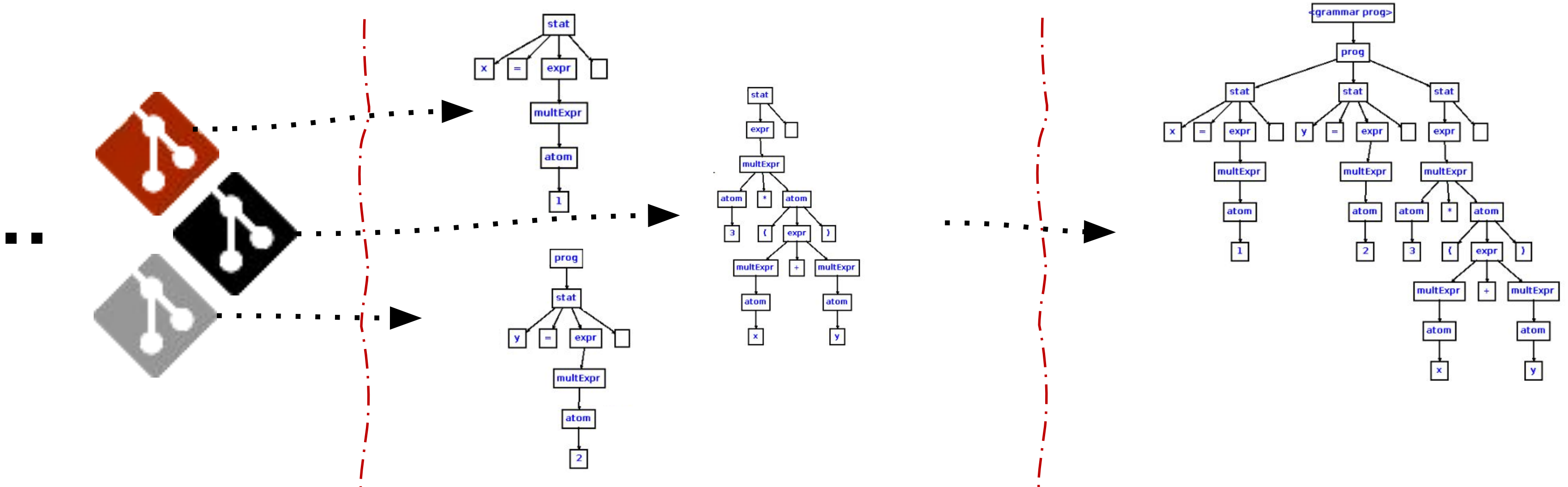
GOAL: IMPROVE STATIC ANALYSIS CAPABILITIES

- Recognize **high-level structures/components** in code
- **Combine** results across analyzed codebases
- Operate on **heterogeneous** platforms

- Choosing the proper system **Intermediate Representation** (IR)
- If it was **component-based**, it likely fits many platforms
  - We use **a component dependency graph** as IR

CHALLENGES (NEXT FEW SLIDES)

1. How to operate on heterogeneous platforms
2. How can we recognize high-level structures?
3. How to combine multiple codebases



1. OPERATE ON HETEROGENOUS PLATFORMS

- Low-Level Virtual Machine (LLVM)*
  - Designed around a language-independent Intermediate Representation (IR)
  - The IR serves as a portable, high-level assembly language that can be optimized with a variety of transformations
  - Meant for compilers – **removes the high-level language features**
    - Suits heterogeneous platforms
    - **Unsuitable for component detection**
- No good alternative exists that would make it simple to detect components.

*LLVM https://llvm.org/

1. OPERATE ON HETEROGENOUS PLATFORMS

- Our response

Our own solution **Language-Agnostic Abstract Syntax Tree** (LAAST)

- Recently published at IEEE Access’22[1]
- Using RUST language-based core that can parse multiple languages
  - **The benefit of LAAST** – all languages parsed into the same IR

[1] Advancing Static Code Analysis With Language-Agnostic Component Identification, DOI: 10.1109/ACCESS.2022.3160485

2. RECOGNIZING HIGH-LEVEL STRUCTURES

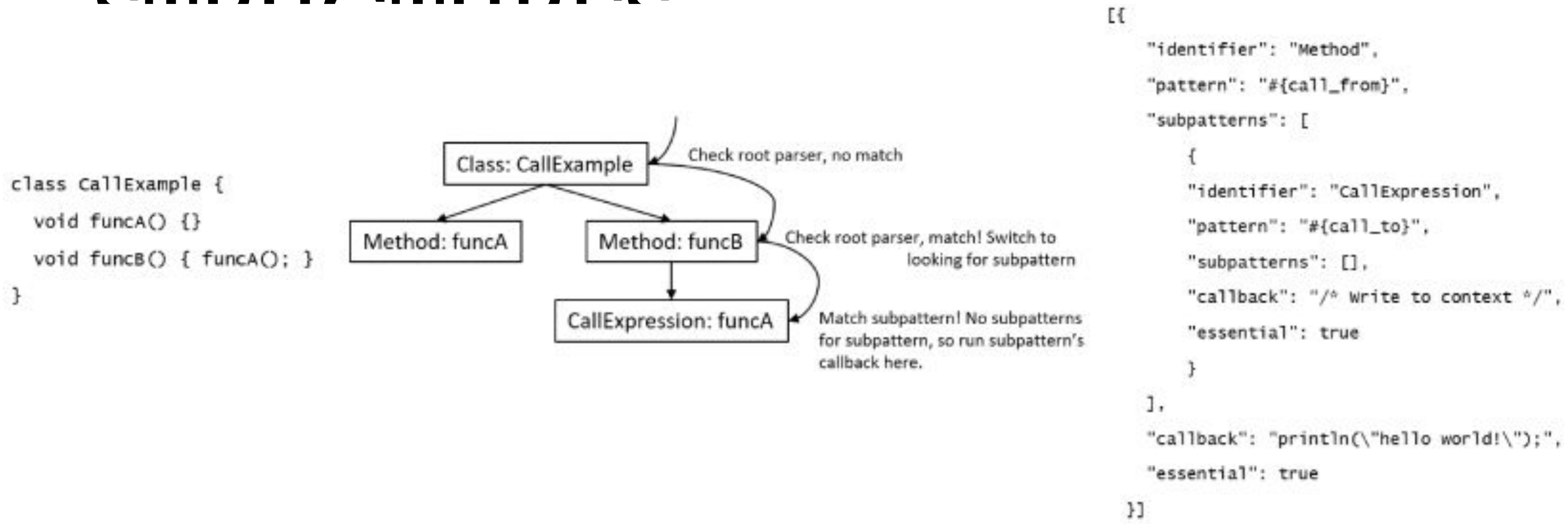
Typically, components or coding conventions.

- **Generalized mechanism: component detectors**
  - Platform-specific patterns to detect components in platform unspecific approach
- **Language-Agnostic Abstract Syntax Tree**[1] is an easy-to-traverse
  - Detect high-level structures through a set of generic parsers - **“detectors,”**
  - **Detectors:** Recursively visit tree nodes to check if expected properties exist on a given subtree to **“match”** a component.

- Some can be detected by annotation; others are more complex structures
  - i.e., inheritance, dependencies, specific properties or menthods
  - Java vs. C++ vs. Python vs. Go

[1] Advancing Static Code Analysis With Language-Agnostic Component Identification, DOI: 10.1109/ACCESS.2022.3160485

2. RECOGNIZING HIGH-LEVEL STRUCTURES



```
class CallExample {
  void funcA() {}
  void funcB() { funcA(); }
}
```

Check root parser, no match

Check root parser, match! Switch to looking for subpattern

Match subpattern! No subpatterns for subpattern, so run subpattern's callback here.

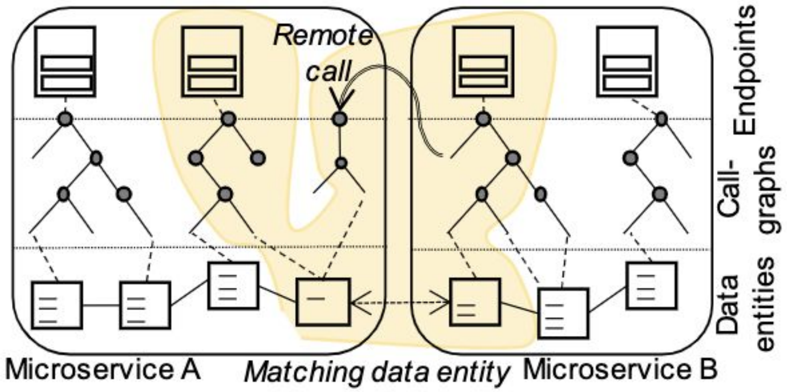
```
[{
   "identifier": "Method",
   "pattern": "#{call_from}",
   "subpatterns": [
      {
      "identifier": "CallExpression",
      "pattern": "#{call_to}",
      "subpatterns": [],
      "callback": "/* write to context */",
      "essential": true
      }
   ],
   "callback": "println(\"hello world!\");",
   "essential": true
}]
```

Sample: Class | LAAST and pattern matching | Resulting structure

- Evaluated on **TrainTicket** (ISCE | Java) and **DeathStarBench** (APLOS | C++) system testbeds
- Component detection : Precision 96-100%, Recall 86-100%

[1] Advancing Static Code Analysis With Language-Agnostic Component Identification, DOI: 10.1109/ACCESS.2022.3160485

3. CONNECTING COI



Having each codebase IR in the form of a component-dependency graph, we can **combine them** into a **holistic system IR**

Three ingredients

1. **Inter-service calls** detection – can be very precise
2. Parsing **deployment descriptors** (i.e., docker files)
3. Microservice overlaps (bounded context/domain driven dev.)
   - Detecting overlaps in **data entities**

Reconstructing the Holistic Architecture of Microservice Systems using Static Analysis, DOI: 10.5220/0011032100003200

SUMMARY OF THE PROCESS

1. Analyze each system part codebase
   - produce LAAST (auxiliary IR to face heterogeneity)
2. Detect components and extract component graph
   - system part IR – i.e., single microservice
3. Combine component graphs based on various strategies
   - holistic system IR – as if the system was a virtual monolith

SO WHAT.. ?

SUMMARY OF THE PROCESS

1. **Analyze** each system-part heterogenous codebase -> LAAST
2. **Detect components** to extract a system-part intermediate representation
3. **Determine** holistic system IR based on various ingredients

Outcome: System intermediate representation

Based on a component dependency graph

SO WHAT..(?)..CAN WE DO WITH THIS?

Static analysis tools in the era of cloud-native systems, DOI: 10.48550/arXiv.2205.08527

EXPERIMENTAL EVALUATION
WHAT PROVED TO WORK

We have applied our proposed approach to many problems

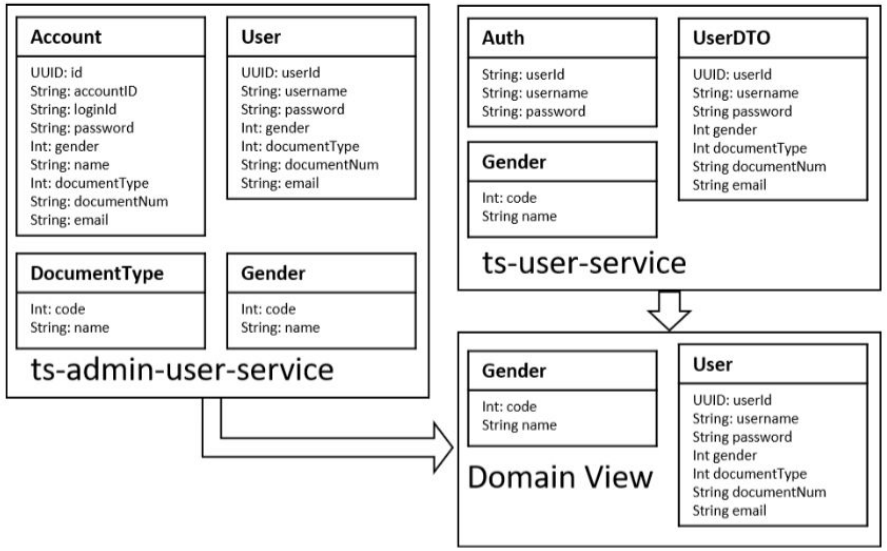
1. Software Architecture Reconstruction (SAR)
2. Visualization of microservice system architecture
3. Reasoning about access policy consistencies (JSR-375) in microservices
4. Detecting microservice bad smells
5. Reasoning about microservice semantic code clones

1. SOFTWARE ARCHITECTURE RECONSTRUCTION (SAR)

Show the decentralized architecture as the system centric perspective

Phases: Extraction | Construction | Manipulation | Reasoning (analysis)

Views: Domain view | Technology view | Service view | Operation view



ts-admin-user-service

Account
UUID: id
String: accountID
String: loginId
String: password
Int: gender
String: name
Int: documentType
String: documentNum
String: email

User
UUID: userId
String: username
String: password
Int: gender
Int: documentType
String: documentNum
String: email

DocumentType
Int: code
String: name

Gender
Int: code
String: name

ts-user-service

Auth
String: userId
String: username
String: password

UserDTO
UUID: userId
String: username
String password
Int gender
Int documentType
String documentNum
String email

Gender
Int: code
String name

Domain View

Gender
Int: code
String name

User
UUID: userId
String: username
String password
Int gender
Int documentType
String documentNum
String email

Fig. 1. Merged Domain View from TrainTicket

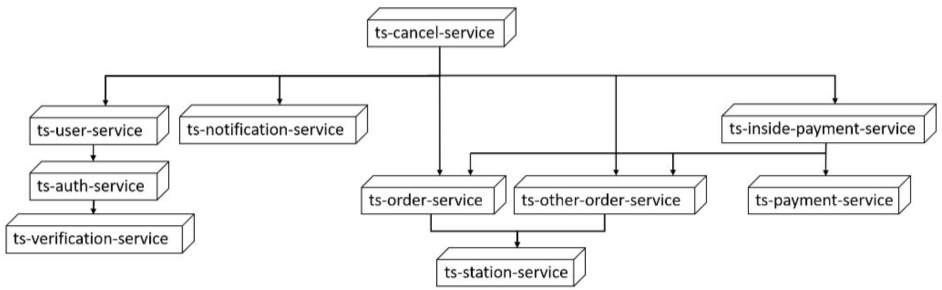


Fig. 2. Service View from TrainTicket

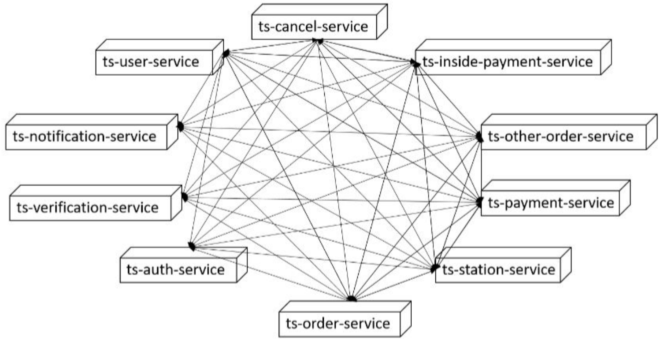


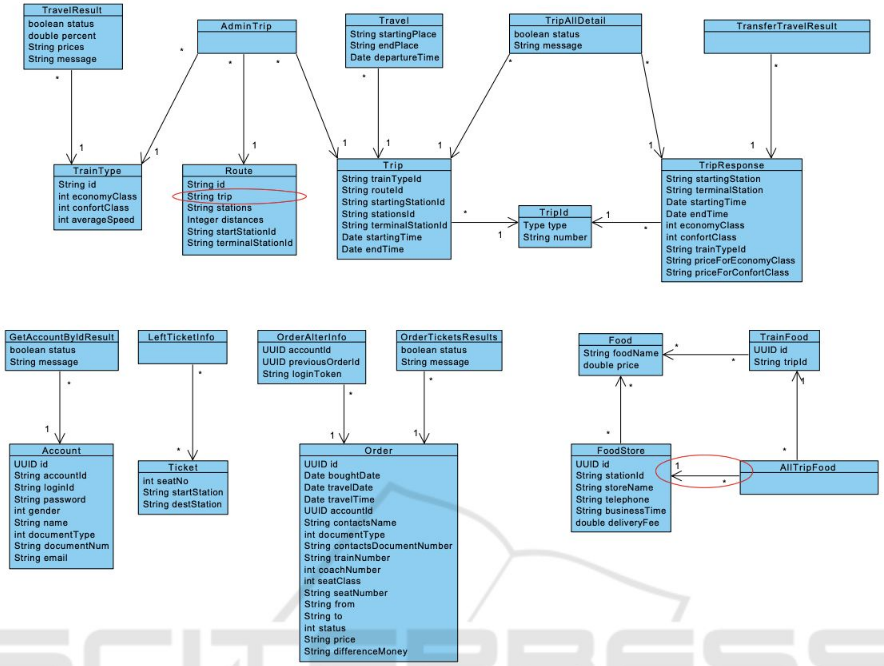
Fig. 3. Operation View from TrainTicket

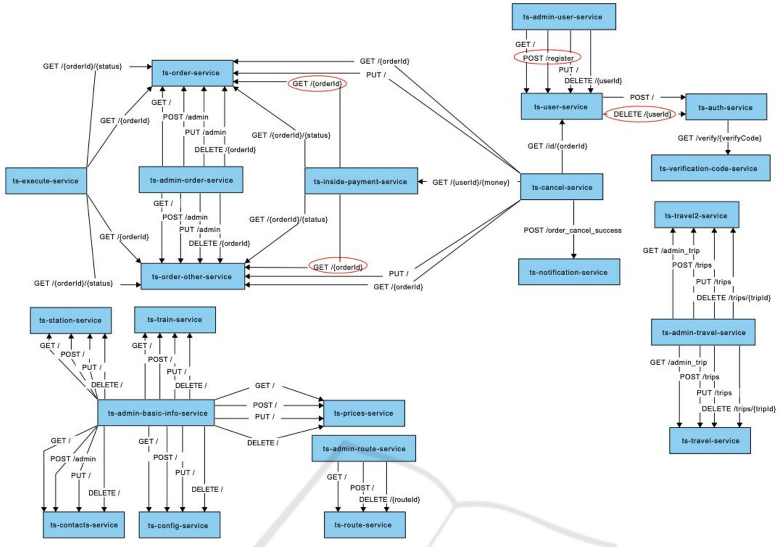
On Automatic Software Architecture Reconstruction of Microservice Applications April 2021 DOI: 10.1007/978-981-33-6385-4_21

2. VISUALIZATION OF MICROSERVICES 1 OF 3

Using SAR to extract a visual view facilitates human-centered reasoning

- Common direction conventional models
  - Problems:
    - two-dimensional space; no interaction
    - does not fit the volume of microservices hundreds



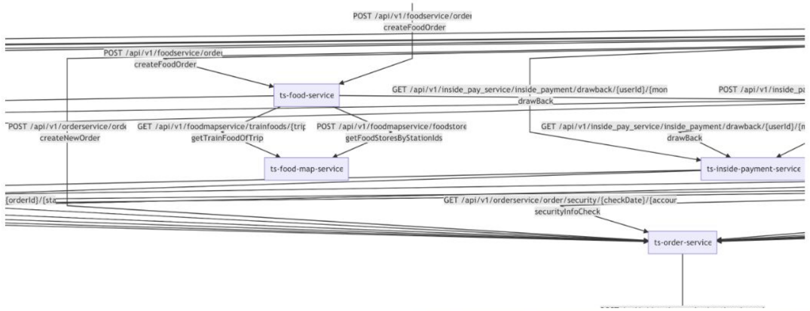


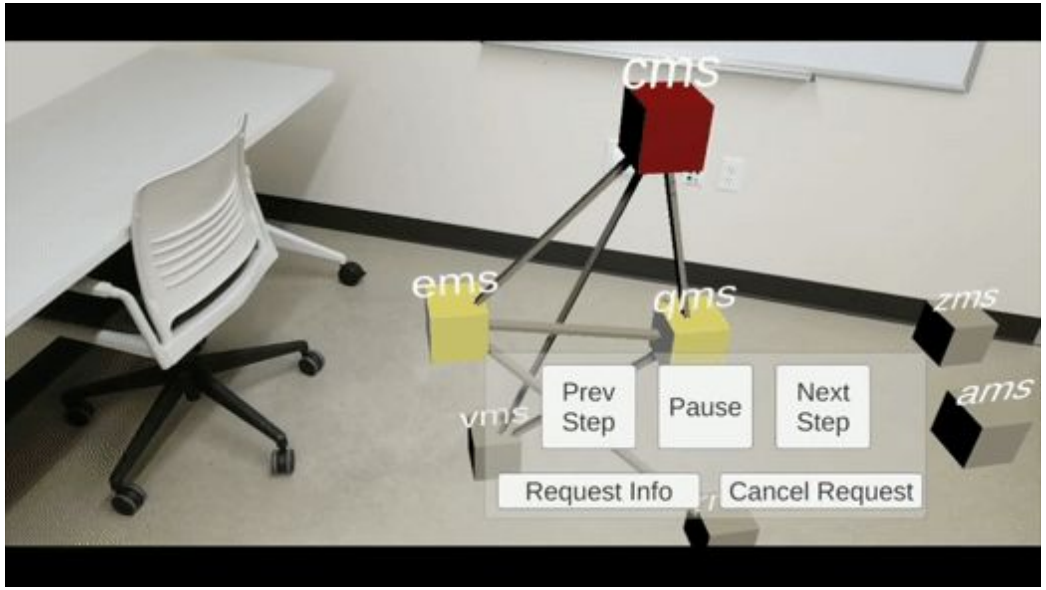
Reconstructing the Holistic Architecture of Microservice Systems using Static Analysis, DOI: 10.5220/0011032100003200

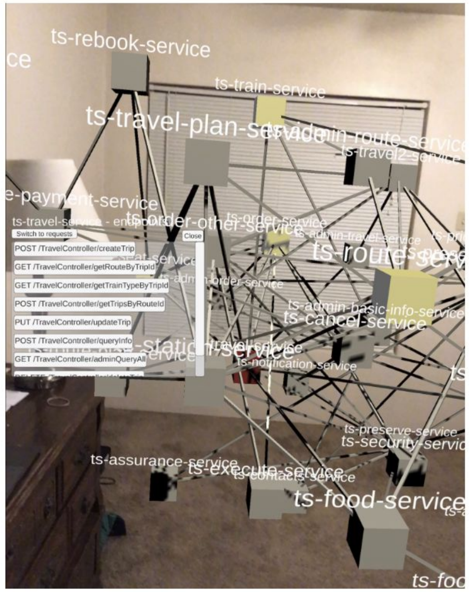
2. VISUALIZATION OF MICROSERVICES 2 OF 3

More ambitions

- Something to fit the volume of microservices
  - Three-dimensional space
  - Augmented reality
  - Interaction
  - Microvision prototype
    - https://www.youtube.com/watch?v=7arBUbglEko


POST /api/v1/foodservice/order
createFoodOrder
POST /api/v1/foodservice/order
createFoodOrder
ts-food-service
GET /api/v1/inside_pay_service/inside_payment/drawback/{userId}/{mon
drawBack
POST /api/v1/inside_pa
POST /api/v1/orderservice/orde
createNewOrder
GET /api/v1/foodmapservice/trainfoods/{trip
getTrainFoodOfTrip
POST /api/v1/foodmapservice/foodstore
getFoodStoresByStationIds
GET /api/v1/inside_pay_service/inside_payment/drawback/{userId}/{n
drawBack
ts-food-map-service
ts-inside-payment-service
{orderId}/{sta
GET /api/v1/orderservice/order/security/{checkDate}/{accour
securityInfoCheck
ts-order-service




ts-travel-service - endpoints

Switch to requests

Close

POST /TravelController/createTrip

GET /TravelController/getRouteByTripId

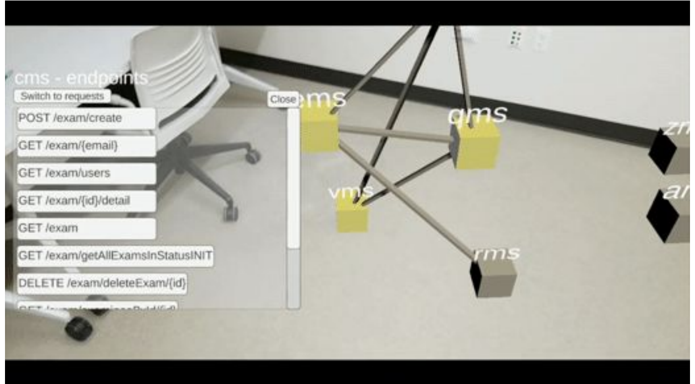
GET /TravelController/getTrainTypeByTripId

POST /TravelController/getTripsByRouteId

PUT /TravelController/updateTrip

POST /TravelController/queryInfo

GET /TravelController/adminQueryAll


cms - endpoints

Switch to requests

Close

POST /exam/create

GET /exam/{email}

GET /exam/users

GET /exam/{id}/detail

GET /exam

GET /exam/getAllExamsInStatusINIT

DELETE /exam/deleteExam/{id}

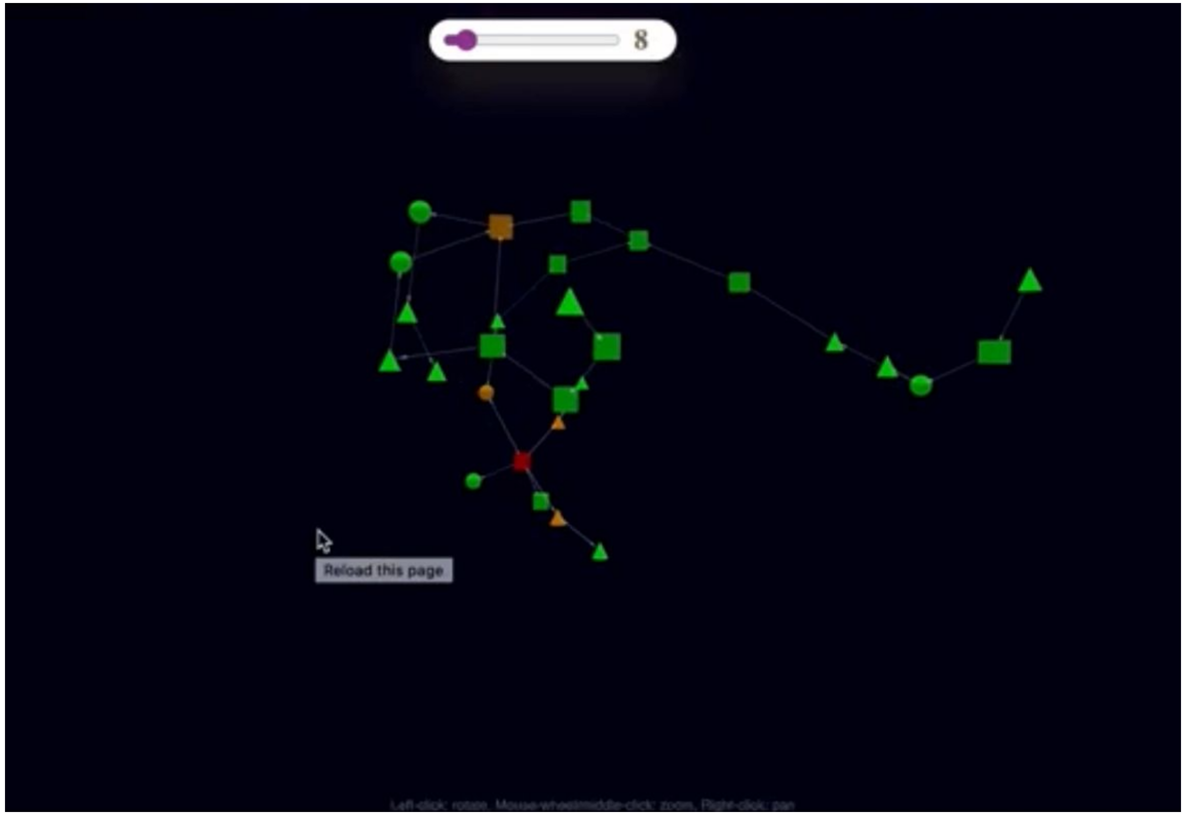
Microvision: Static analysis-based approach to visualizing microservices in augmented reality, SOSE 2022

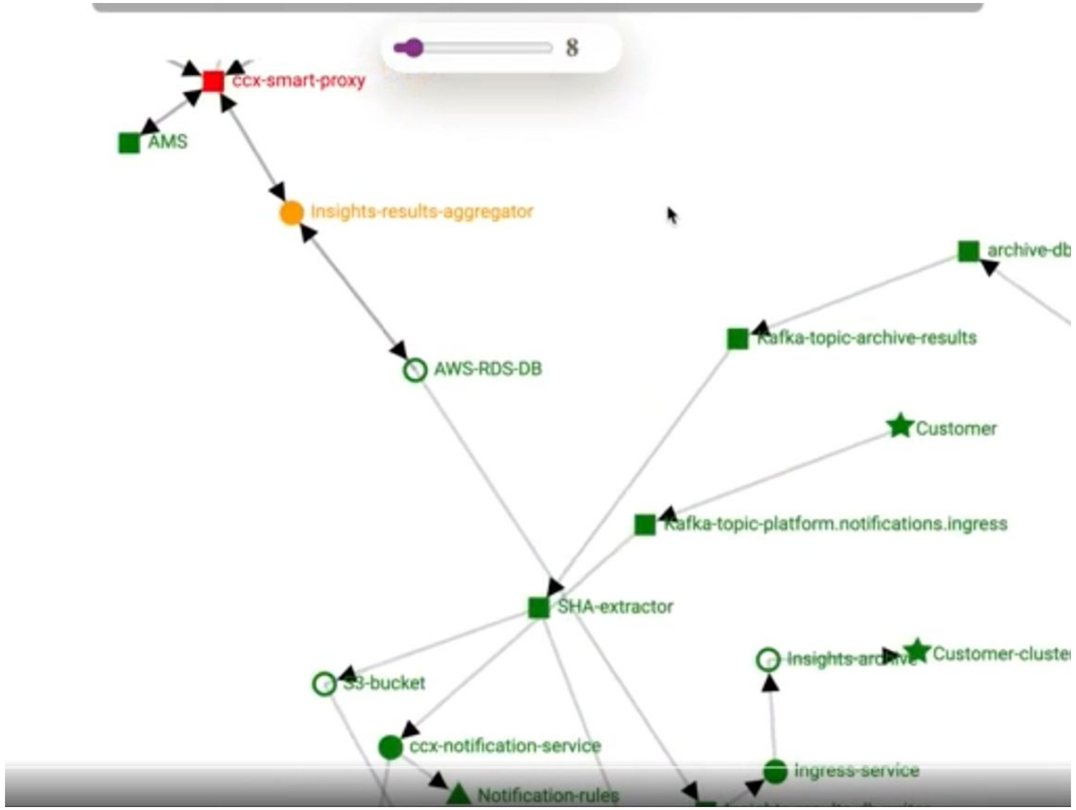
what are you looking for?

Left-click: rotate, Mouse-wheel/middle-click: zoom, Right-click: pan

2. VISUALIZATION OF MICROSERVICES 3 OF 3

- Fall 2022 student research


8
Reload this page
Left-click: rotate, Mouse-wheel/middle-click: zoom, Right-click: pan



1 of 3

3. REASONING : ACCESS POLICY CONSISTENCY

We can detect **endpoint access policy enforcements**

- Components are recognized with all properties
  - i.e., JSR-375 Role-based access control
- Check whether different endpoints apply equivalent access policy
- Perform across microservices (through inter-service communications)
  - **Detect consistency errors across microservices**

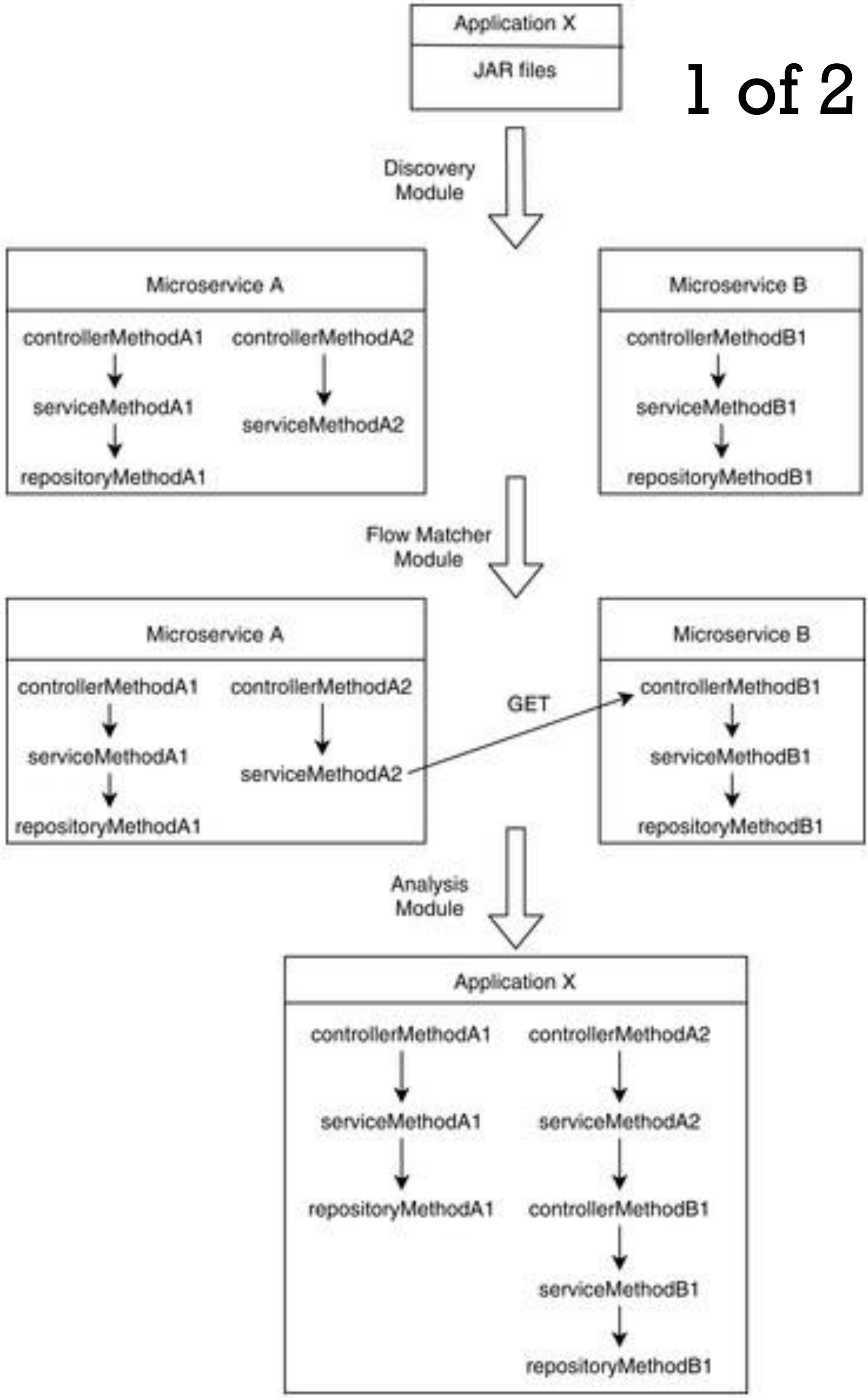
On automated RBAC assessment by constructing a centralized perspective for microservice mesh, DOI:10.7717/peerj-cs.376

1 of 2

3. ACCESS POLICY CONSISTENCY

We can detect **enforced endpoint access policy**

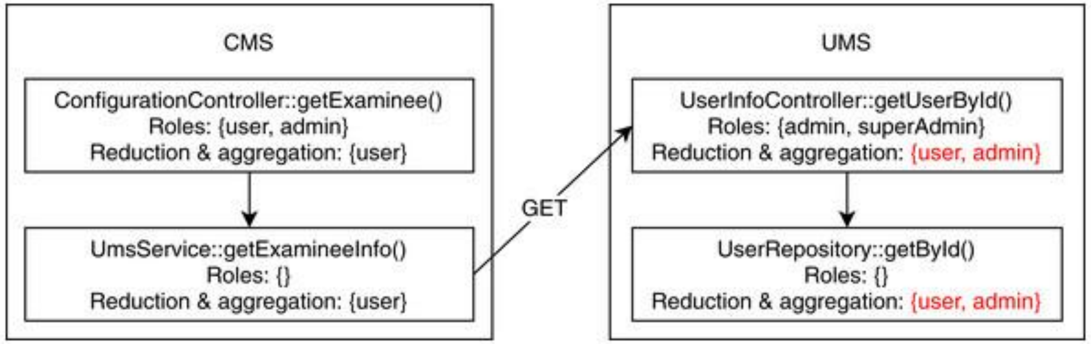
- Components recognized with all properties
  - i.e., JSR-375 Role-based access control
- Determine access policy equivalence across different endpoints
  - Perform across microservices
    - (through inter-service communications)
    - **Detect consistency errors**



On automated RBAC assessment by constructing a centralized perspective for microservice mesh, DOI:10.7717/peerj-cs.376

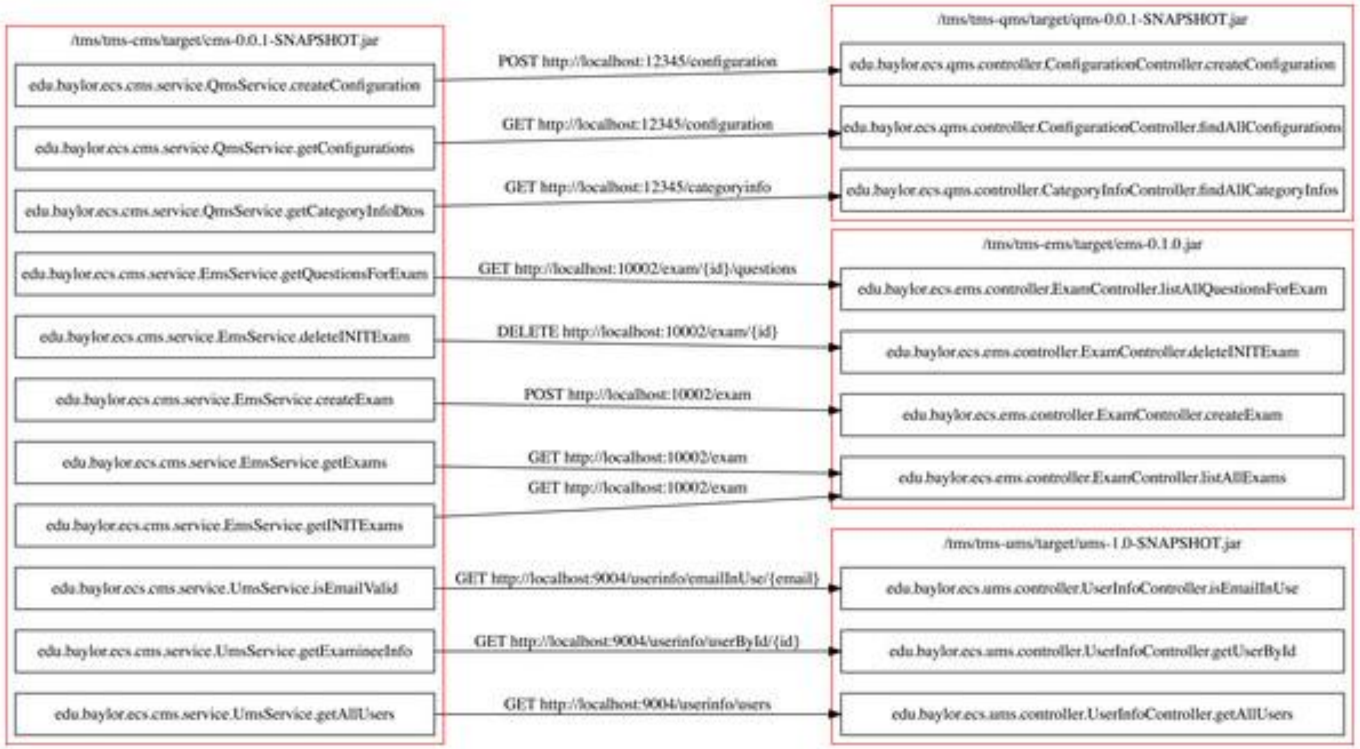
2 of 2

3. ACCESS POLICY CONSISTENCY


CMS
ConfigurationController::getExaminee()
Roles: {user, admin}
Reduction & aggregation: {user}
UmsService::getExamineeInfo()
Roles: {}
Reduction & aggregation: {user}
GET
UMS
UserInfoController::getUserById()
Roles: {admin, superAdmin}
Reduction & aggregation: {user, admin}
UserRepository::getById()
Roles: {}
Reduction & aggregation: {user, admin}

Detecting 5 violations

- Missing role violations
- Unknown access violations
- Entity access violations
- Conflicting hierarchy violations
- Unrelated access violations


/tms/tms-cms/target/cms-0.0.1-SNAPSHOT.jar
edu.baylor.ecs.cms.service.QmsService.createConfiguration
edu.baylor.ecs.cms.service.QmsService.getConfigurations
edu.baylor.ecs.cms.service.QmsService.getCategoryInfoDtos
edu.baylor.ecs.cms.service.EmsService.getQuestionsForExam
edu.baylor.ecs.cms.service.EmsService.deleteINITExam
edu.baylor.ecs.cms.service.EmsService.createExam
edu.baylor.ecs.cms.service.EmsService.getExams
edu.baylor.ecs.cms.service.EmsService.getINITExams
edu.baylor.ecs.cms.service.UmsService.isEmailValid
edu.baylor.ecs.cms.service.UmsService.getExamineeInfo
edu.baylor.ecs.cms.service.UmsService.getAllUsers
POST http://localhost:12345/configuration
GET http://localhost:12345/configuration
GET http://localhost:12345/categoryinfo
GET http://localhost:10002/exam/{id}/questions
DELETE http://localhost:10002/exam/{id}
POST http://localhost:10002/exam
GET http://localhost:10002/exam
GET http://localhost:10002/exam
GET http://localhost:9004/userinfo/emailInUse/{email}
GET http://localhost:9004/userinfo/userById/{id}
GET http://localhost:9004/userinfo/users
/tms/tms-qms/target/qms-0.0.1-SNAPSHOT.jar
edu.baylor.ecs.qms.controller.ConfigurationController.createConfiguration
edu.baylor.ecs.qms.controller.ConfigurationController.findAllConfigurations
edu.baylor.ecs.qms.controller.CategoryInfoController.findAllCategoryInfos
/tms/tms-ems/target/ems-0.1.0.jar
edu.baylor.ecs.ems.controller.ExamController.listAllQuestionsForExam
edu.baylor.ecs.ems.controller.ExamController.deleteINITExam
edu.baylor.ecs.ems.controller.ExamController.createExam
edu.baylor.ecs.ems.controller.ExamController.listAllExams
/tms/tms-ums/target/ums-1.0-SNAPSHOT.jar
edu.baylor.ecs.ums.controller.UserInfoController.isEmailInUse
edu.baylor.ecs.ums.controller.UserInfoController.getUserById
edu.baylor.ecs.ums.controller.UserInfoController.getAllUsers

TrainTicket testbench

On automated RBAC assessment by constructing a centralized perspective for microservice mesh, DOI:10.7717/peerj-cs.376

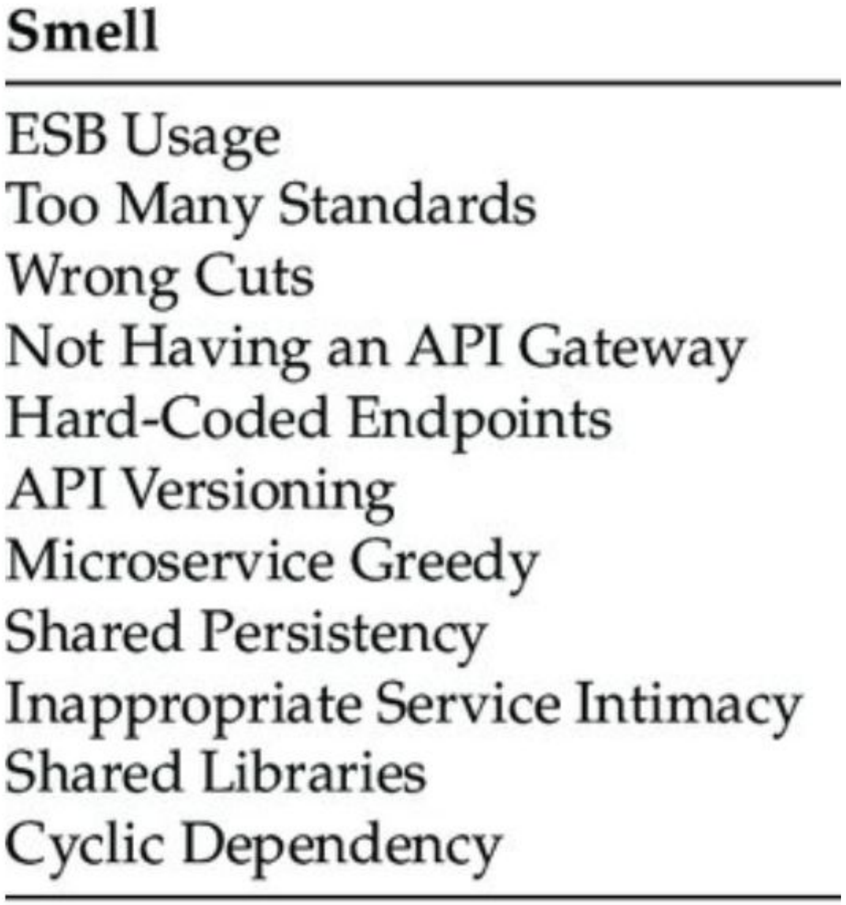
4. MICROSERVICE BAD

Having **System IR** and reading

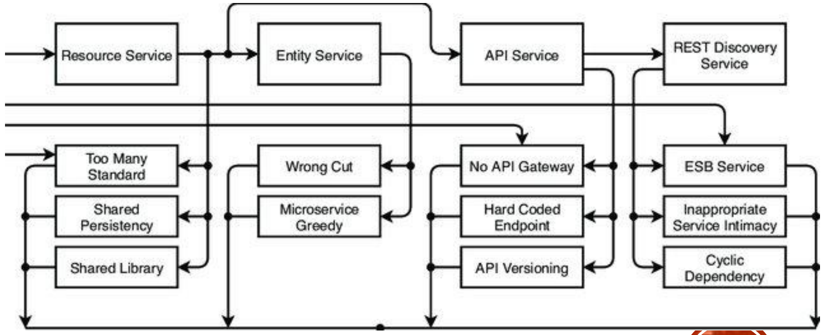
"On the Definition of Microservice Bad Smells, DOI: 10.1109/MS.2018.2141031"

Pattern matching on the holistic **system IR**

- Detecting 11 bad smells.
- MSANose tool



Smell
ESB Usage
Too Many Standards
Wrong Cuts
Not Having an API Gateway
Hard-Coded Endpoints
API Versioning
Microservice Greedy
Shared Persistency
Inappropriate Service Intimacy
Shared Libraries
Cyclic Dependency



Automated Code-Smell Detection in Microservices Through Static Analysis: A Case Study, DOI: 10.3390/app10217800

5. SEMANTIC CODE CLONES

With too much development autonomy, or upon system integration.

- Certain features might coexist but are hidden in heterogeneity.

Syntactic clone

- Looks the same / does it do the same thing

- Approach using **system IR** considering components in control flow as a heuristic to narrow our similarity identification, then detecting which operations we perform with data and whether the data seem similar.
  - High Accuracy received on TrainTicket Benchmark
  - List of microservices/endpoints that are similar, ordered by similarity

Semantic Code Clone Detection Method for Distributed Enterprise Systems, DOI: 10.5220/0011032200003200

5. SEMANTIC CODE CLONES

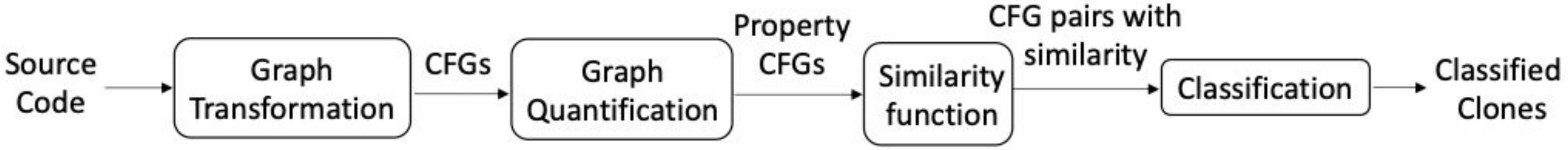
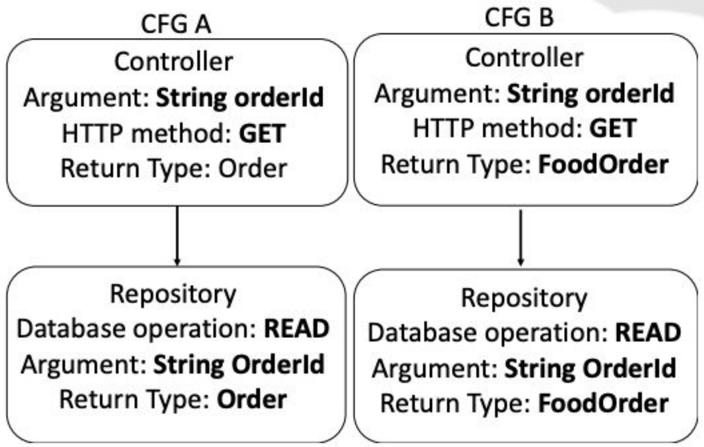
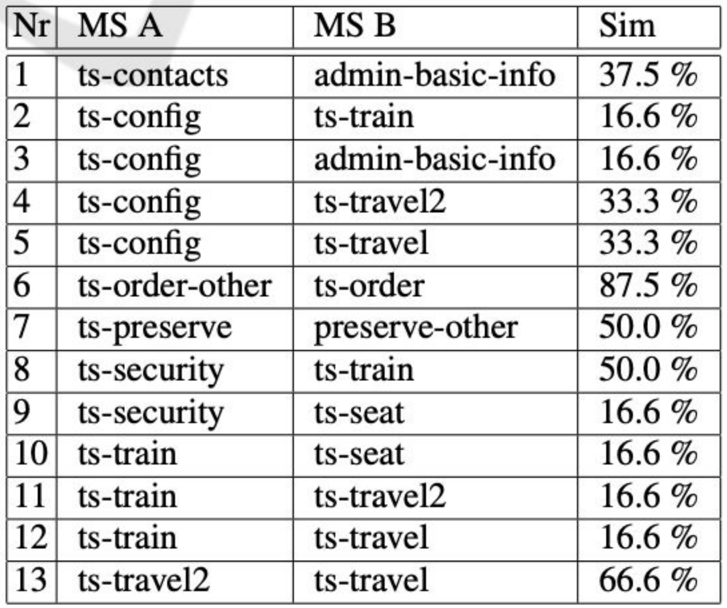


Figure 1: Schema of the algorithm.

$$sim(a_i,b_i) = ctr(a_i,b_i) + rfc(a_i,b_i) + rp(a_i,b_i)$$





Nr	MS A	MS B	Sim
1	ts-contacts	admin-basic-info	37.5 %
2	ts-config	ts-train	16.6 %
3	ts-config	admin-basic-info	16.6 %
4	ts-config	ts-travel2	33.3 %
5	ts-config	ts-travel	33.3 %
6	ts-order-other	ts-order	87.5 %
7	ts-preserve	preserve-other	50.0 %
8	ts-security	ts-train	50.0 %
9	ts-security	ts-seat	16.6 %
10	ts-train	ts-seat	16.6 %
11	ts-train	ts-travel2	16.6 %
12	ts-train	ts-travel	16.6 %
13	ts-travel2	ts-travel	66.6 %

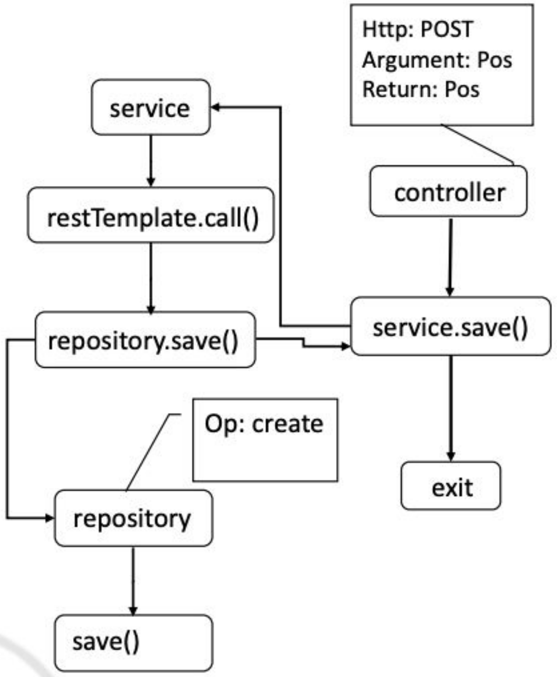
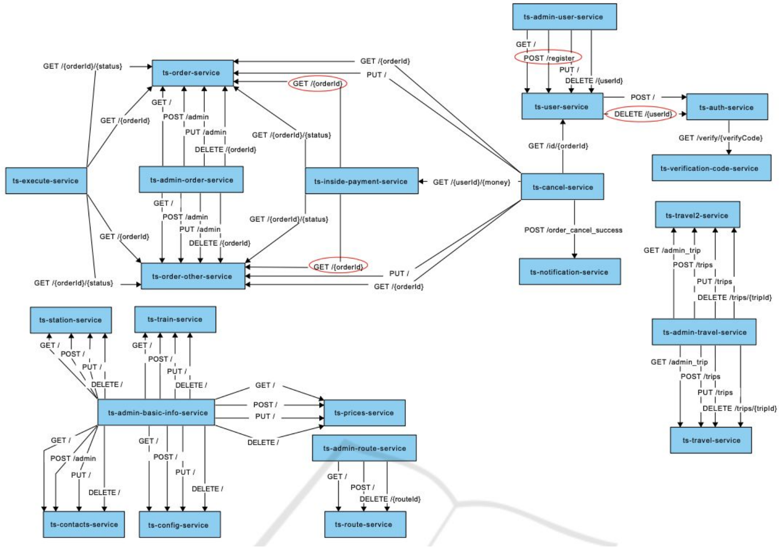


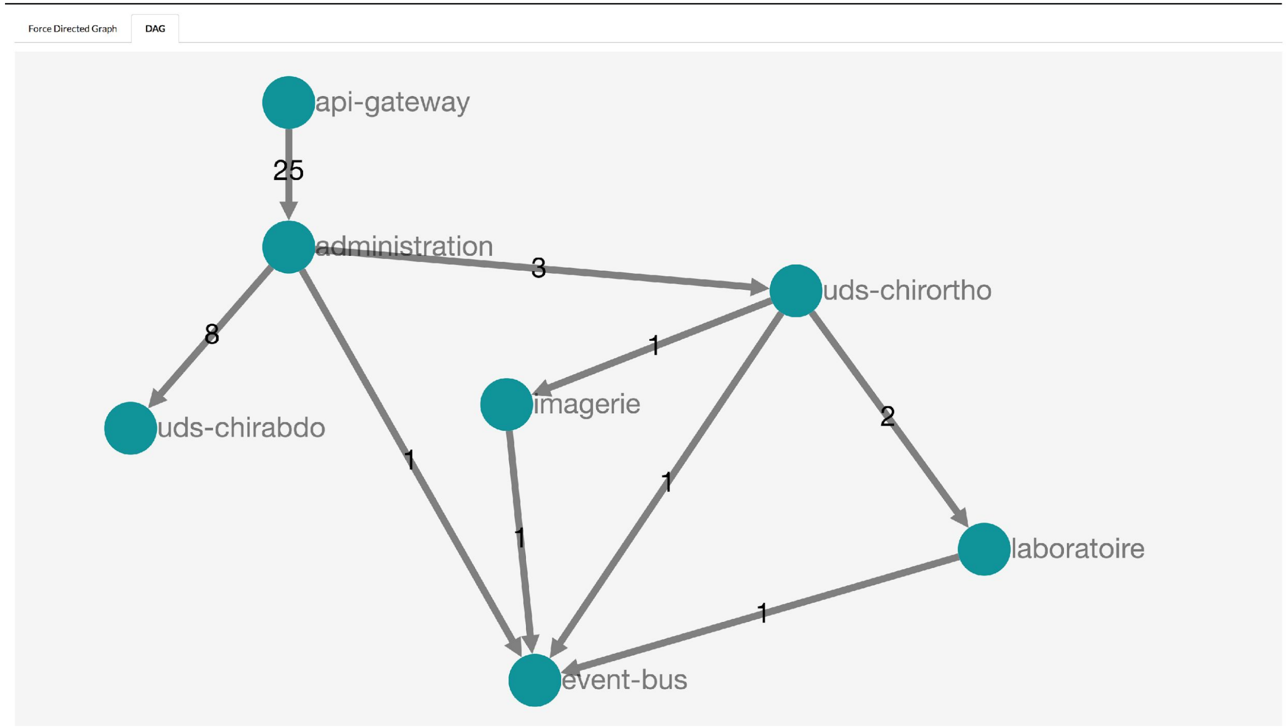
Figure 2: Example of control-flow graph.

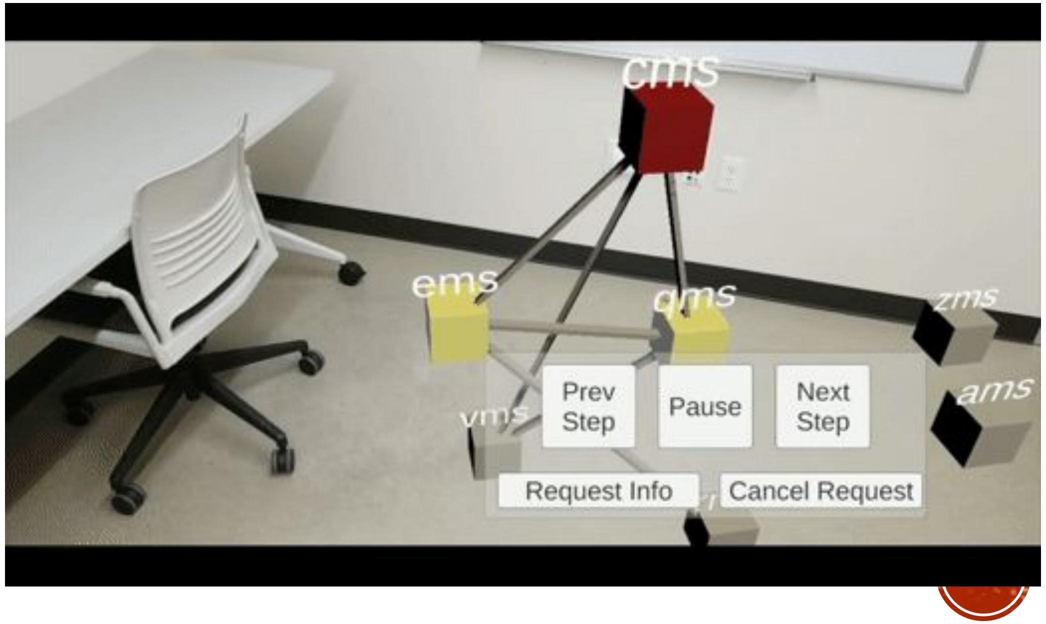
Semantic Code Clone Detection Method for Distributed Enterprise Systems, DOI: 10.5220/0011032200003200

WHAT DID WE DEMONSTRATE?



- Static analysis can be beneficial to decentralized system analysis.
- It can do much more! I.E., help developers with codebase changes.





WHAT DID WE DEMONSTRATE?

- Robustness of our System IR to various tasks for Microservices
  - Targeting problems/gaps in the Microservices

- Possibly foundation to holistic static analysis of Microservices

- With such promising results, we can broaden our future research.

This material is based upon work supported by the National Science Foundation under Grant No. 1854049, a grant from Red Hat Research.

CONCLUSION

- Arguing why static analysis is not used in cloud-native systems
- Recognizing barriers to progress
- Introducing our experimental solution
  - **System IR based on component awareness**
- Sharing promising evaluation
- Asking you to contribute!
  - https://github.com/cloudhubs

FUTURE WORK

- Architectural Degradation and Technical Debt detection
- Improve component and component dependency parsing
  - Broaden language support beyond Java/C++/Go
  - Heterogenous system benchmark study
- Messaging integration to get a more comprehensive perspective
  - work in progress
- Integration with dynamic analysis
- Continuous restructuring of microservices
  - IEEE SOSE 2022 Best Paper Award for Soft K-means approach[2]

[2] Monolith to Microservices: VAE-Based GNN Approach with Duplication Consideration, SOSE 2022

Questions?

WHAT DID WE MISS?

Post your questions/remark to
Tomas_Cerny@baylor.edu