Software Engineering for the service of Services

Naouel Moha

École de Technologie Supérieure de Montréal

Naouel.Moha@etsmtl.ca

Séminaire du Master 2 informatique
de l'université de Montpellier,
December 10th, 2021

ÉTS
Le génie pour l'industrie

Outline

- A Brief History of Services
- SOA to Microservices, and beyond
- Current and Open Challenges
- Our contributions *“Migrating Legacy Systems to SOA”*

Outline

- A Brief History of Services
- SOA to Microservices, and beyond
- Current and Open Challenges
- Our contributions “Migrating Legacy Systems to SOA”

The Evolution of Software Architecture

- Italian food perspective

1990's

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

2000's

LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)

2010's

RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)

WHAT'S NEXT?

PROBABLY PIZZA-ORIENTED ARCHITECTURE

https://benorama.com/the-evolution-of-software-architecture-bd6ea674c477

20 years ago…


BUZZWORD

2000's


WEB 2.0


EXCHANGE
PROTOCOL
SYSTEM
DEVELOPER
PROVIDER
WEB
SOAP
WSDL
COMMUNICATION
NETWORK
SERVICE
XML
INTEROPERABILITY
REQUESTER
TESTING
API
INTERFACE
DATA
DEVICES
SOFTWARE
INTERACTION
CODE


SOA
SERVICE ORIENTED ARCHITECTURE
VECTOR

Nowadays!

Today

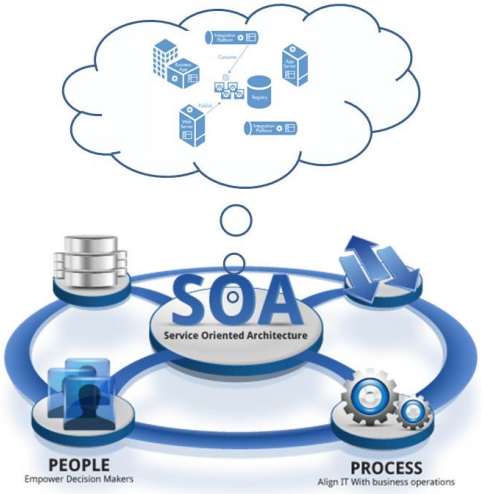
Code
DevOps
Two-Pizza
Sourcecode
Directory
Search
Microservices
Agility
ESB
APIs
Hybrid
Domain
Cloud
Routing
Sprint
Autonomy
Agile
Testing
CICD
Discovery
Integration
Open
Containers
Automation

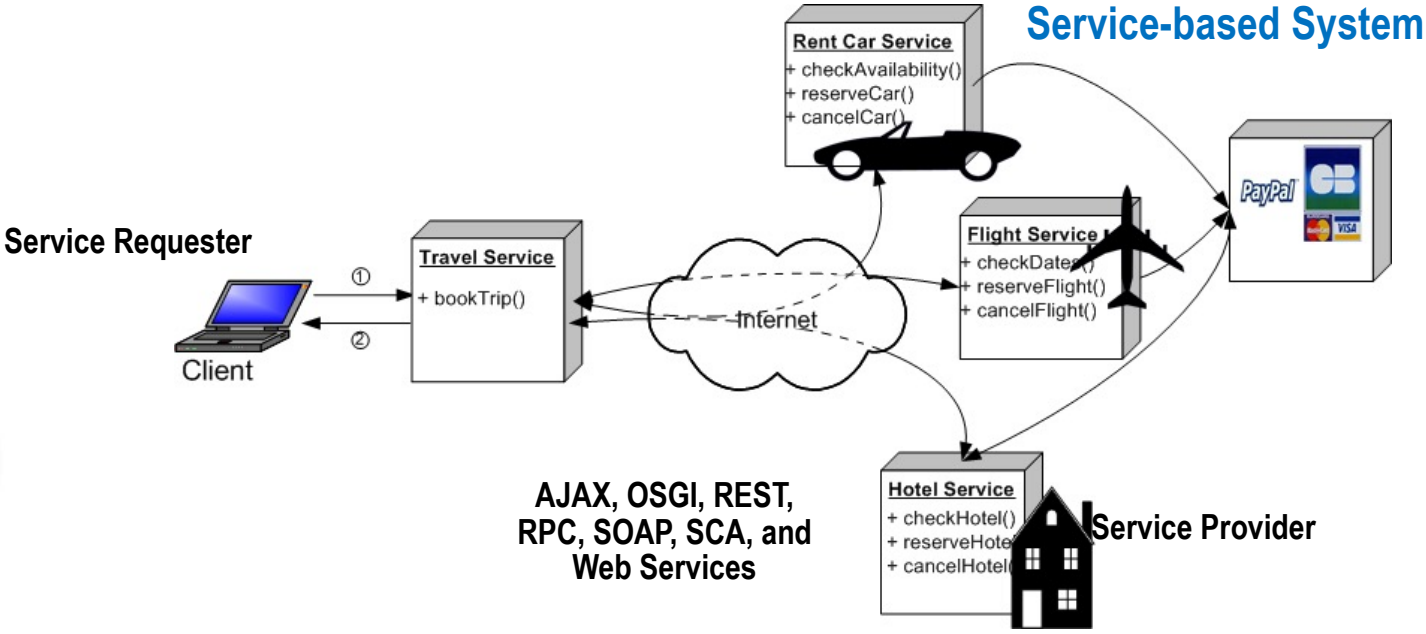
Outline

- A Brief History of Services
- SOA to Microservices, and beyond
- Current and Open Challenges
- Our contributions “Migrating Legacy Systems to SOA”

Service-Oriented Architecture

- « Architectural style encapsulating business functionality into separate services, which can be freely composed to realize higher-level business processes »

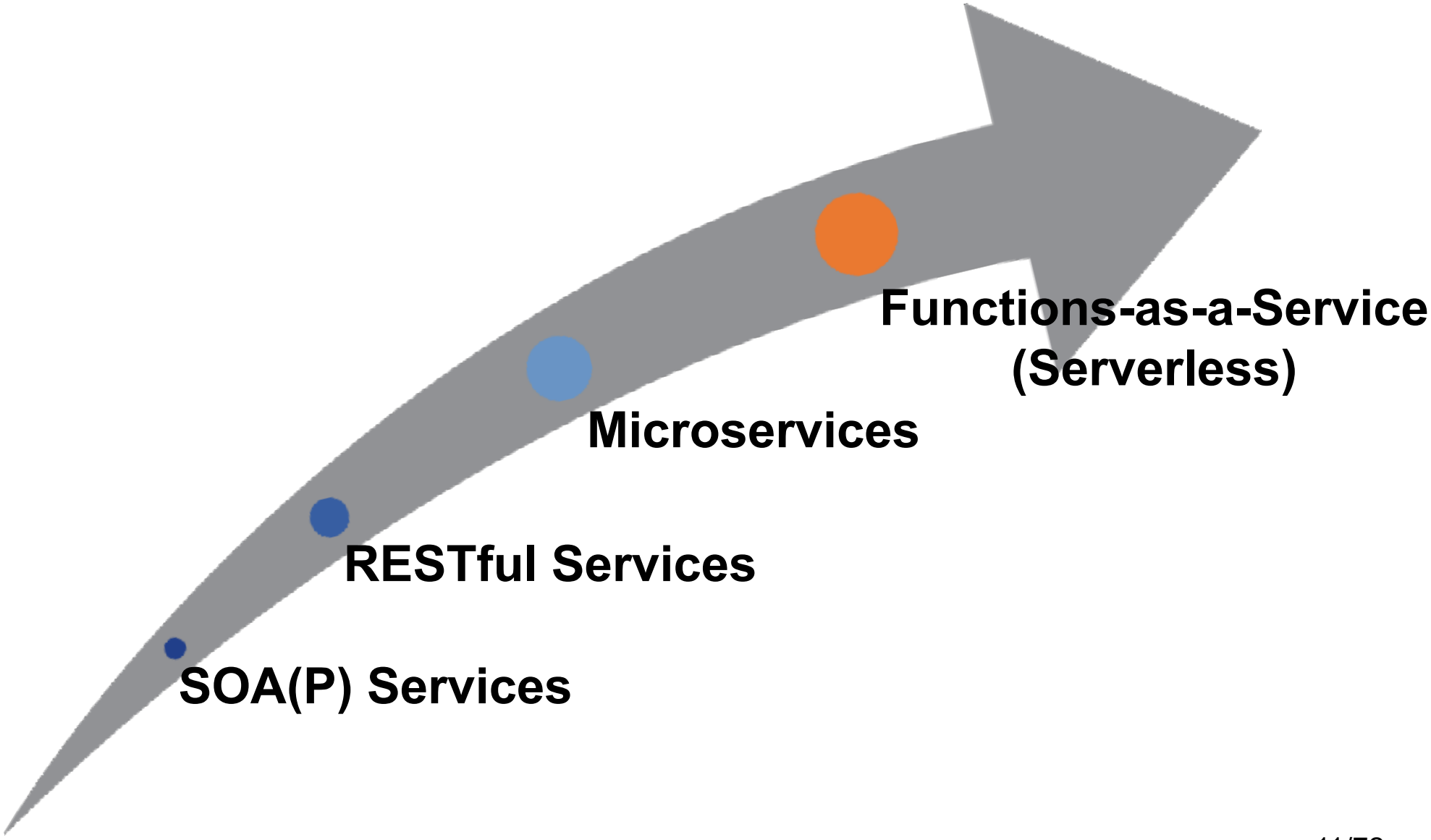




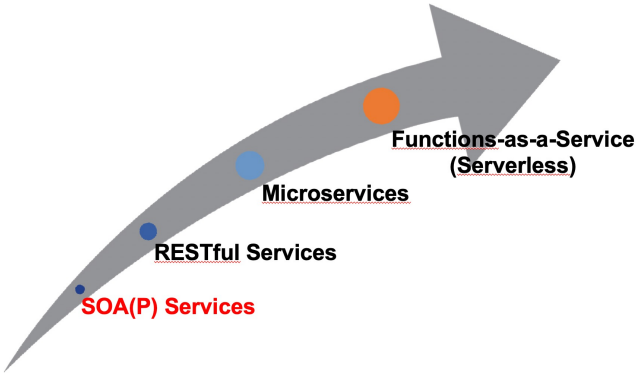
- « A web service is a program with a well-defined interface (contract mostly specified in WSDL) and an id (URI), which can be located, published, and invoked through standard Web protocols »

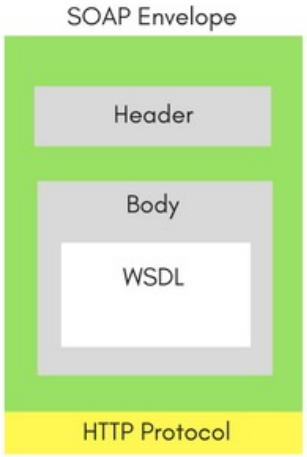
Antonopoulos, Exarchakos, Li, Liotta. Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications, January, 2010

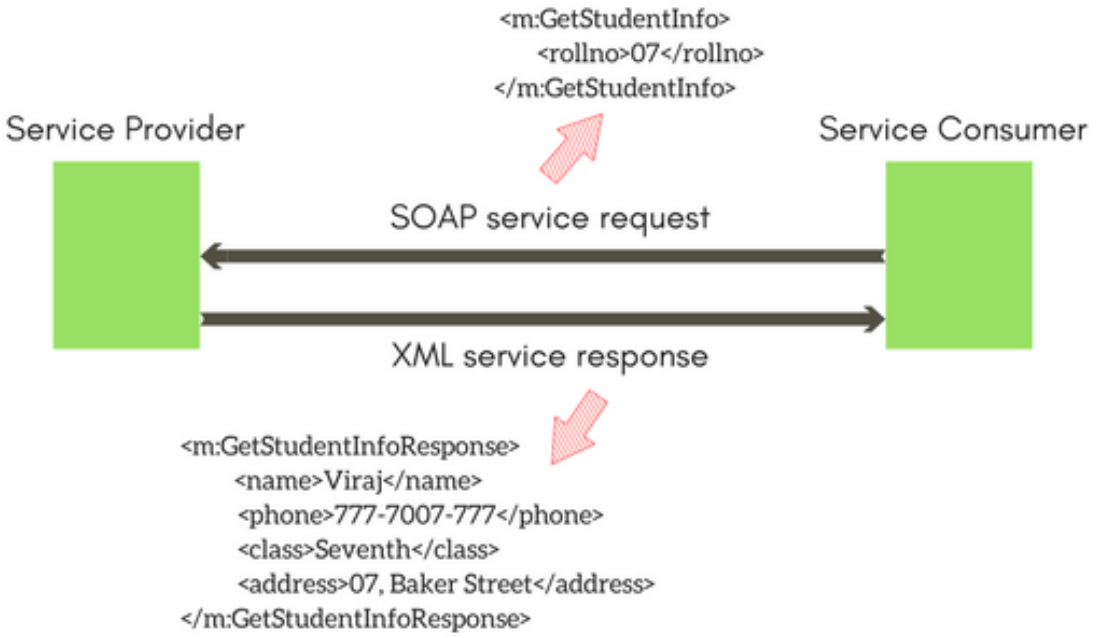
SOA to Microservices, and beyond



SOA(P) Services





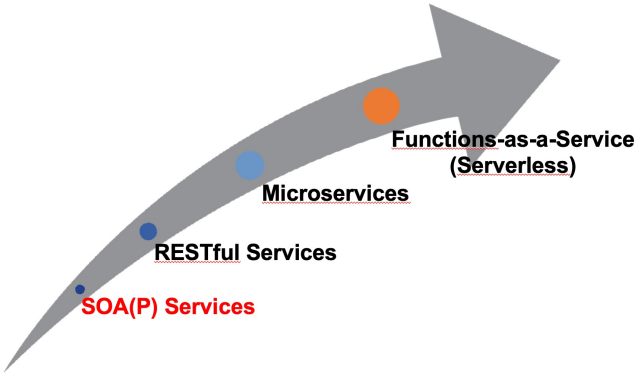


SOAP (Simple Object Access Protocol) is an XML-based protocol

WSDL (Web Service Description Language) = contract btw Service and Client

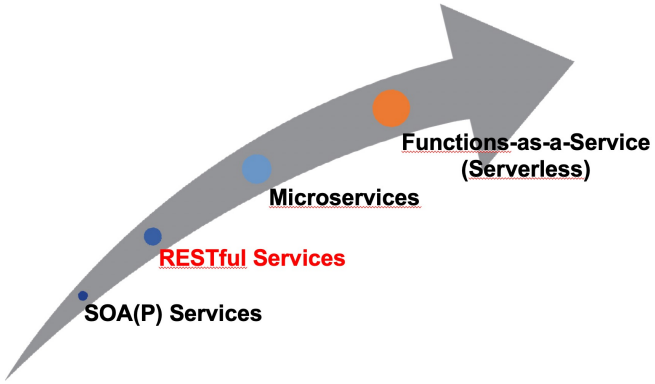
https://www.studytonight.com/rest-web-service/types-of-webservices

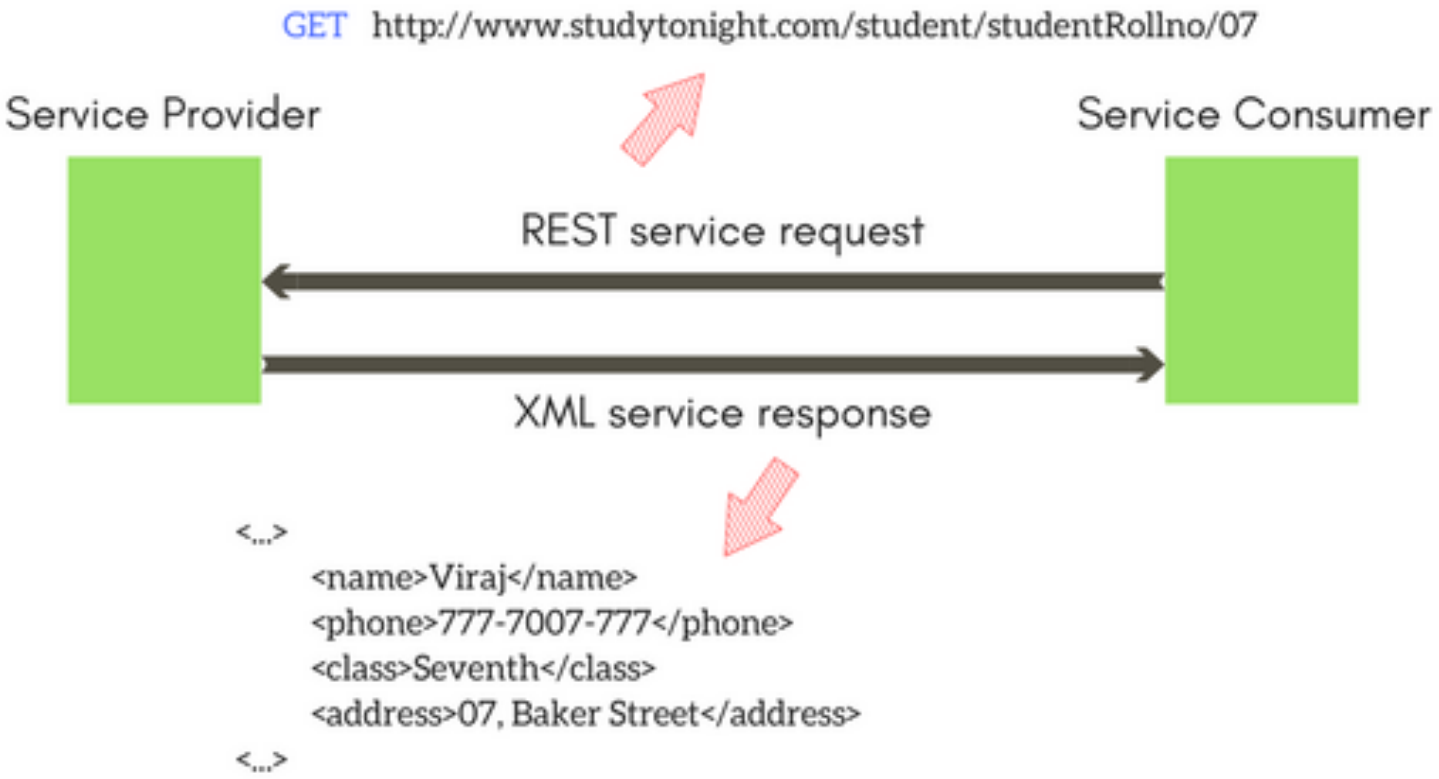
SOA(P) Services



- **Composition** of Services
  - Definition of clear/standard steps (modeling, binding, executing, and verifying) of Web Service composition
  - Classification of compositions into
    - Workflow-based industry solutions (extending existing languages, e.g., WS-BPEL and BPEL4WS)
    - Semantics-based academic solutions, using planning/AI upon semantic languages such as OWL-S

RESTful Services

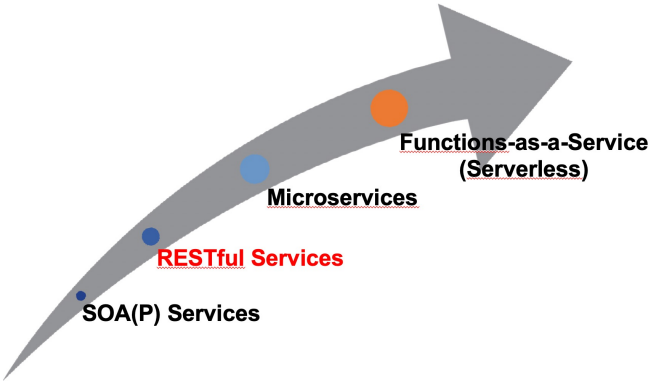




REST (REpresentational State Transfer) = HTTP methods PUT, POST, GET, and DELETE

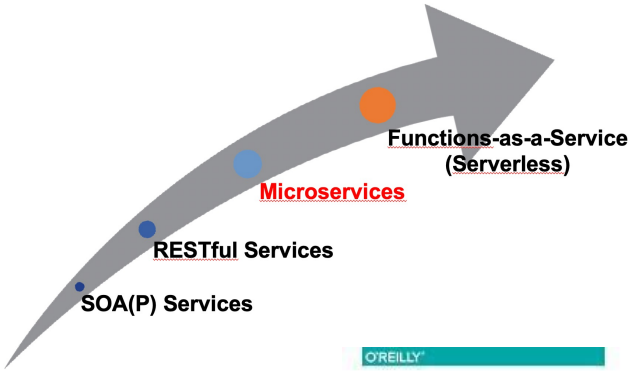
https://www.studytonight.com/rest-web-service/types-of-webservices

RESTful Services

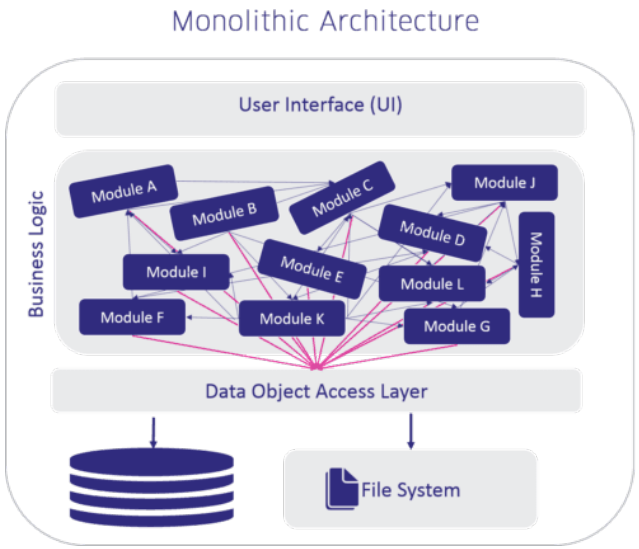


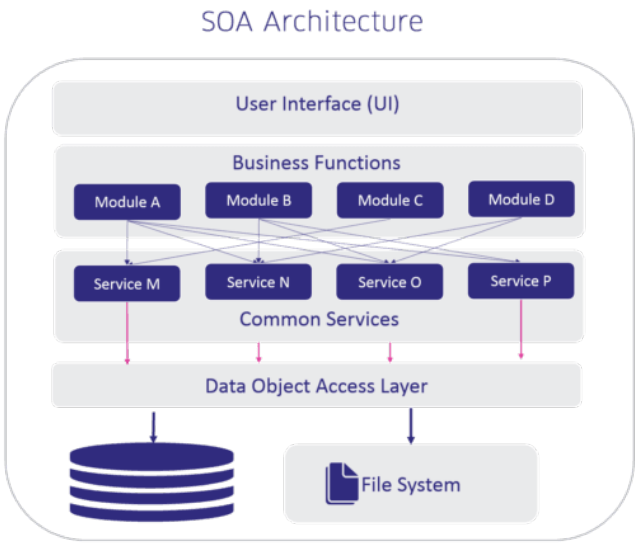
- Still, reuse and composition under discussion in the RESTful era
  - Definition of clear/standard steps (modeling, binding, executing, and verifying) of Web Service composition
  - Classification of compositions into
    - Workflow-based industry solutions : BPEL4REST
    - Semantic-based annotations, planning-based solutions, and formalization

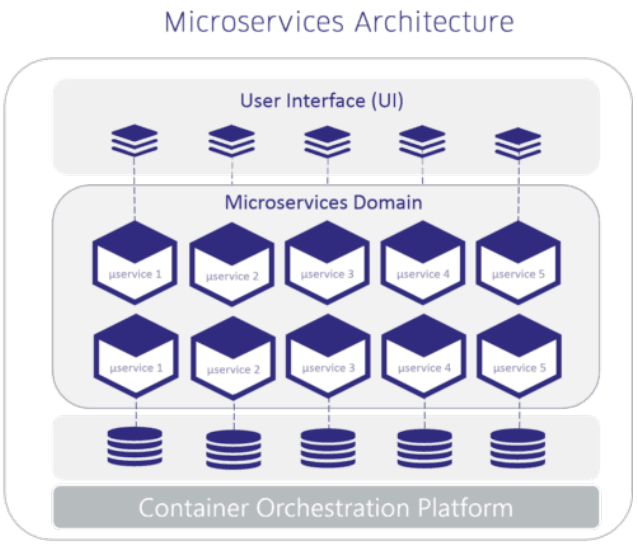
Microservices



- « Microservices are small, autonomous services that work together »
  - SOA = *share-as-much-as-possible*
  - Microservices = *share-as-little-as-possible*

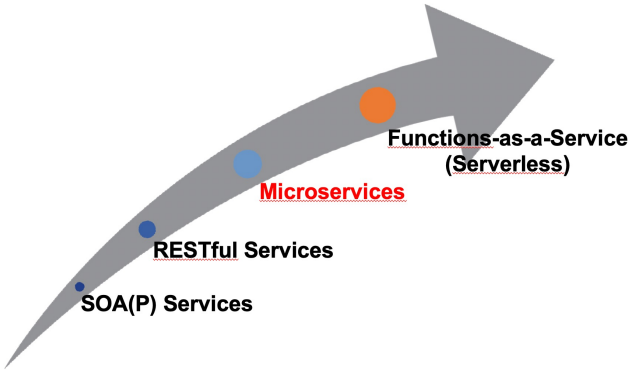






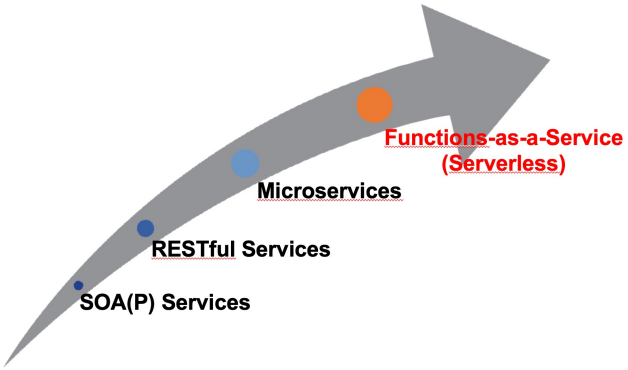
https://www.sumerge.com/what-is-microservices-architecture/

Microservices

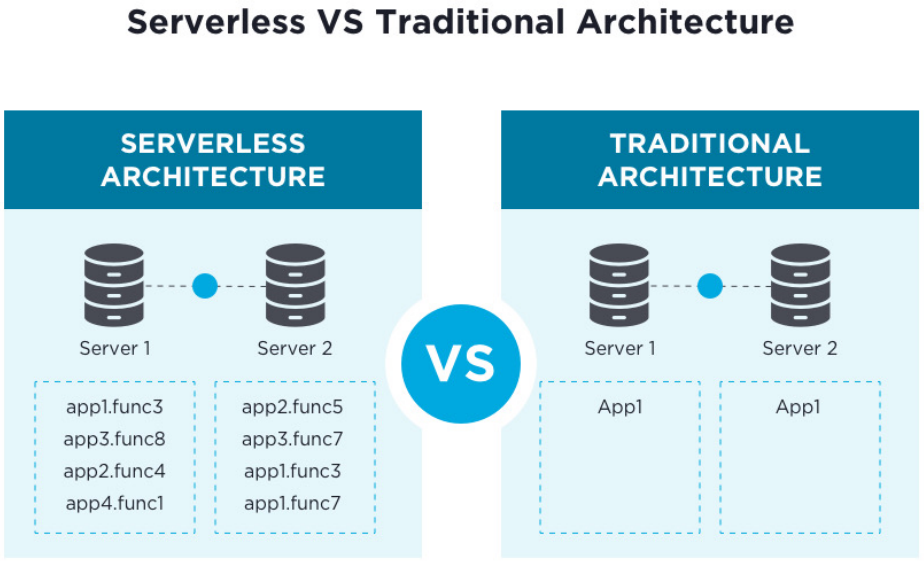


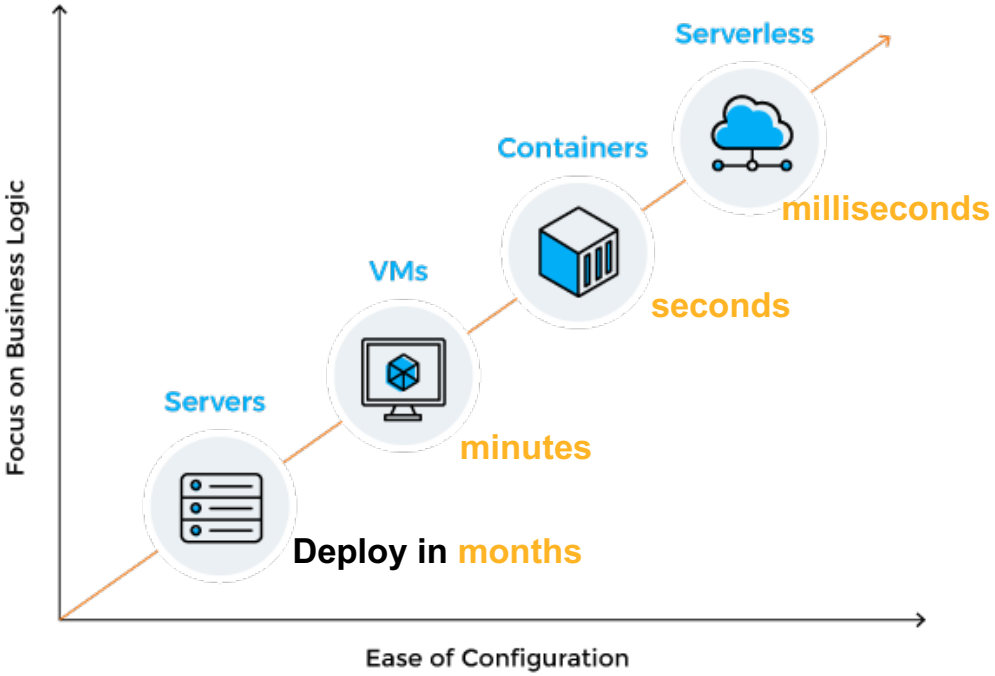
- Issues related to defining, classifying, and characterizing services and composition solutions are overcome
- New challenges posed by the IoT, pervasive computing, and mobile apps
- Environment more open, dynamic and ever changing

FaaS - Serveless



- « Function as a service (FaaS) is a cloud computing model that allows users to develop applications and deploy functionalities without managing a server, thus improving process efficiency »
- Pay for only what you use





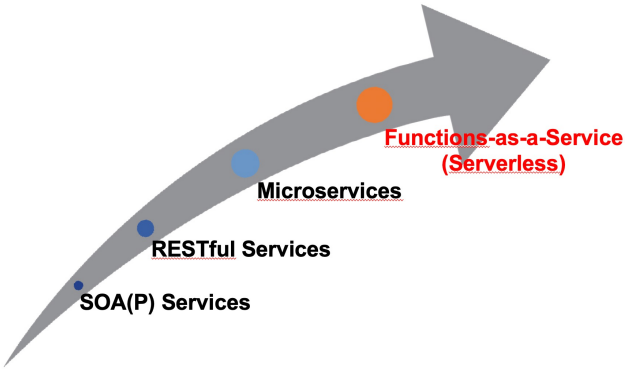
AWS Lambda

Azure Functions

Google Cloud Functions

https://www.plutora.com/blog/serverless
https://www.onelogin.com/learn/serverless-computing

FaaS - Serveless

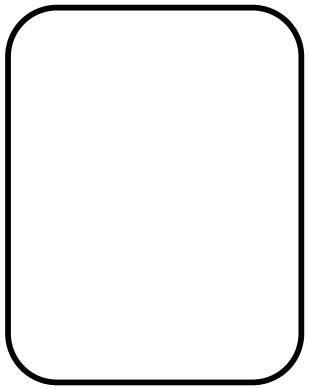


- Challenges
  - Determine the sweet spots where running code in a FaaS environment can deliver economic benefits
  - Automatically profile existing code to offload computation to serverless functions
  - Bring adequate isolation among functions
  - Determine the right granularity to exploit data and code locality
  - Provide methods to handle state (given that functions are stateless by definition)
  - Increase the number of out-of-the-box tools to test and deploy functions locally

SOA to Microservices, and beyond

Monolith

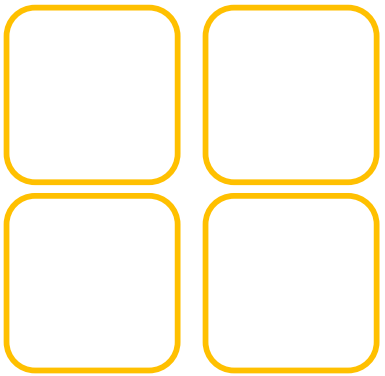
Single Unit



+ Simplified arch.
+ Less to deploy
+ Less to manage
- Inflexible
- Slow updates

SOA

Coarse-grained

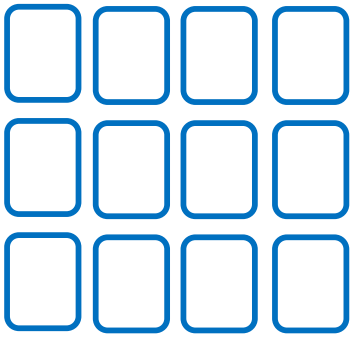


+ Separation of concerns
+ Specular to business
- Pre-cloud
- No infrastructure focus

Microservices

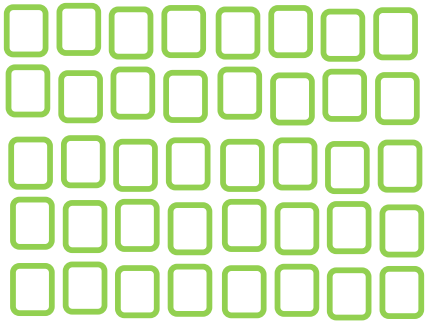
Fine-grained

Canonical (container based)



+ Container-based
+ Easy to migrate
+ Reproducible
+ Vendor-agnostic
- Manual admin
- Running costs

Serverless FaaS (platform)



+ Scalability
+ Cost
+ Zero admin
- Resource limits
- Size limit
- Vendor lock-in

SOA to Microservices, and beyond

Monolithic

Startups,
Small Apps

Small resource
base

are architecture types

SOA

Enterprise apps with
complex operations

Microservices

Complex large-scale
systems

Multiple skilled teams

Serverless

Client-heavy apps

Fast-growing and
rapidly changing apps

High-latency
background tasks

Our Contribs : Preserving and Improving Software Quality in Service-based Systems

SOFA Service Oriented Framework For Analysis

Mosser, Caissy, Juroszek, Vouters, Moha. Charting Microservices to Support Services' Developers: The Anaximander Approach. ICSOC 2020: 36-44

Tighilt, Abdellatif, Moha, Mili, El Boussaidi, Privat, Guéhéneuc. On the Study of Microservices Antipatterns: a Catalog Proposal. EuroPLoP 2020: 34:1-34:13

Palma, Moha, Guéhéneuc. UniDoSA: The Unified Specification and Detection of Service Antipatterns. *IEEE Transactions on Software Engineering (TSE)*, 45(10):1024-1053, 2019.

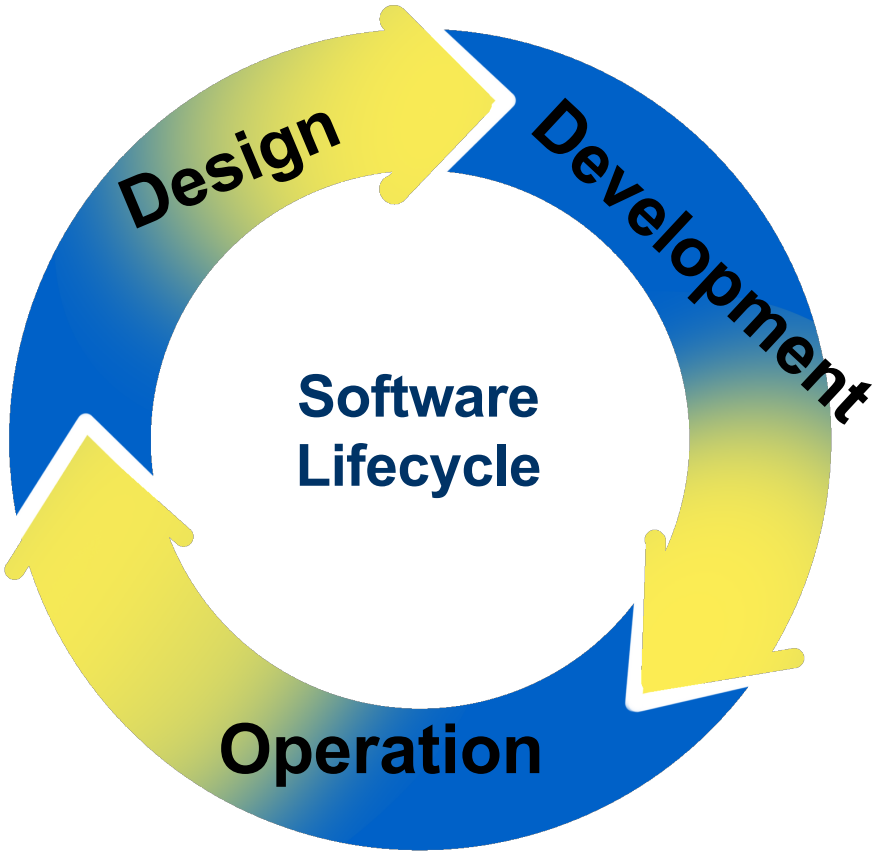
Abdellatif, et al. A multi-dimensional study on the state of the practice of REST APIs usage in Android apps. Automated Software Engineering 27(3): 187-228 (2020)

Moha, Palma, *et al*. Specification and Detection of SOA Antipatterns. *10th International Conference on Service-Oriented Computing (ICSOC),* 2012. Best Runner-Up Paper

Outline

- A Brief History of Services
- SOA to Microservices, and beyond
- Current and Open Challenges
- Our contributions “Migrating Legacy Systems to SOA”

Challenges



Microservices: The Evolution and Extinction of Web Services?

Luciano Baresi and Martin Garriga

Abstract In the early 2000s, service-oriented architectures (SOA) emerged as a paradigm for distributed computing, e-business processing, and enterprise integration. Rapidly, SOA and web services became the subject of hype, and virtually every organization tried to adopt them, no matter their actual suitability. Even worse, there were nearly as many definitions of SOA as people adopting it. This led to a big fail on many of those attempts, as they tried to change the problem to fit the solution. Nowadays, microservices are the new weapon of choice to achieve the same (and even more) goals posed to SOA years ago. Microservices ("SOA done right") describe a particular way of designing software applications as suites of independently deployable services, bringing dynamism, modularity, distributed development, and integration of heterogeneous systems. However, nothing comes for free: new (and old) challenges appeared, including service design and specification, data integrity, and consistency management. In this chapter, we identify such challenges through an evolutionary view from the early years of SOA to microservices, and beyond. Our findings are backed by a literature review, comprising both academic and gray literature. Afterwards, we analyze how such challenges are addressed in practice, and which challenges remain open, by inspecting microservice-related projects on GitHub, the largest open-source repository to date.

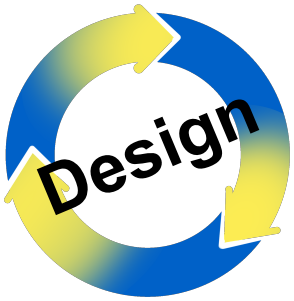
L. Baresi
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
e-mail: luciano.baresi@polimi.it

M. Garriga (✉)
Faculty of Informatics, National University of Comahue, Neuquán, Argentina

CONICET, National Scientific and Technical Research Council, Buenos Aires, Argentina
e-mail: martin.garriga@fi.uncoma.edu.ar

A. Bucchiarone et al. (eds.), *Microservices*,
https://doi.org/10.1007/978-3-030-31646-4_1

Challenges



- Lack of academic efforts regarding the design practices and patterns
  - Resilience patterns such as Circuit-breaker
- Dimensioning microservices i.e. finding the right granularity
  - Trade-off between size and number of microservices
- Security by design
  - Proliferation of endpoints in microservice ecosystems

Challenges

- Lack of proposals for asynchronous interaction models
- Concurrent versions and incremental releases
  - Recent efforts on standardizing RESTful APIs through OpenAPI specs
- Data persistency and consistency
- Resilience Testing
  - Testing the failure-handling capabilities of microservices

Challenges

- Increase in consumption of computing and network resources
  - Mistrust regarding built-in solutions of cloud providers → too rigide or cumbersome to configure and adjust
- Operational complexity
  - Challenging to locate and coordinate microservices

Challenges

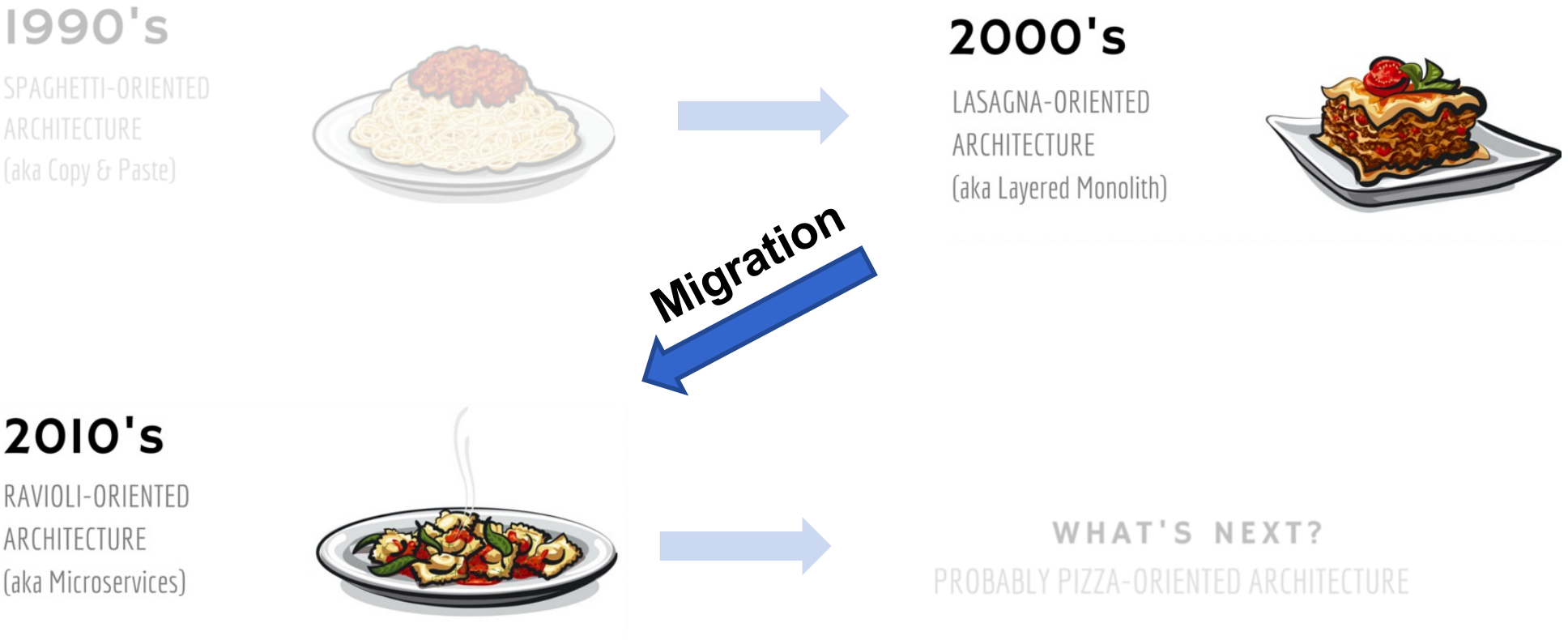
- Due to their novelty and intrinsic complexity and distribution
- Their design, development, and operation is hampered
  - Business logic is heavily distributed over many independent and asynchronously evolving microservices

Outline

- A Brief History of Services
- SOA to Microservices, and beyond
- Current and Open Challenges
- Our contributions “Migrating Legacy Systems to SOA”

Migrating Legacy Systems to SOA

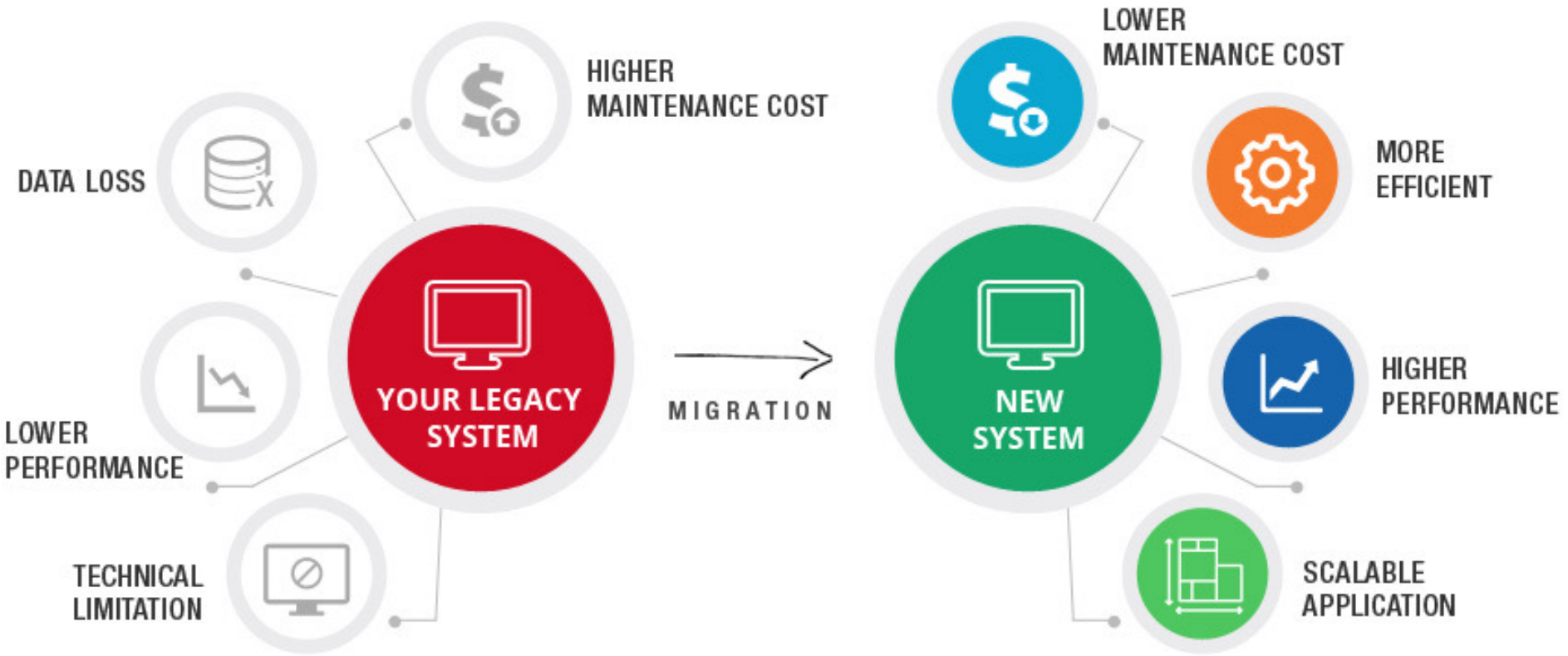
- The Evolution of Software Architecture



https://benorama.com/the-evolution-of-software-architecture-bd6ea674c477

Migrating Legacy Systems to SOA

- Modernisation of Legacy Systems



Source : www.truecodemasters.com

FINANCIER WORLDWIDE corporatefinanceintelligence

Legacy systems: the replacement and modernisation headache

August 2017 | COVER STORY | BANKING & FINANCE

Financier Worldwide Magazine

August 2017 Issue

70% of today's transactional operations are managed by legacy systems

Legacy systems are critical

but challenging to maintain

Shortcomings

- Maintenance Costs
- Lack of Flexibility
- Lack of Scalability
- Lack of Support

Migrating your legacy system

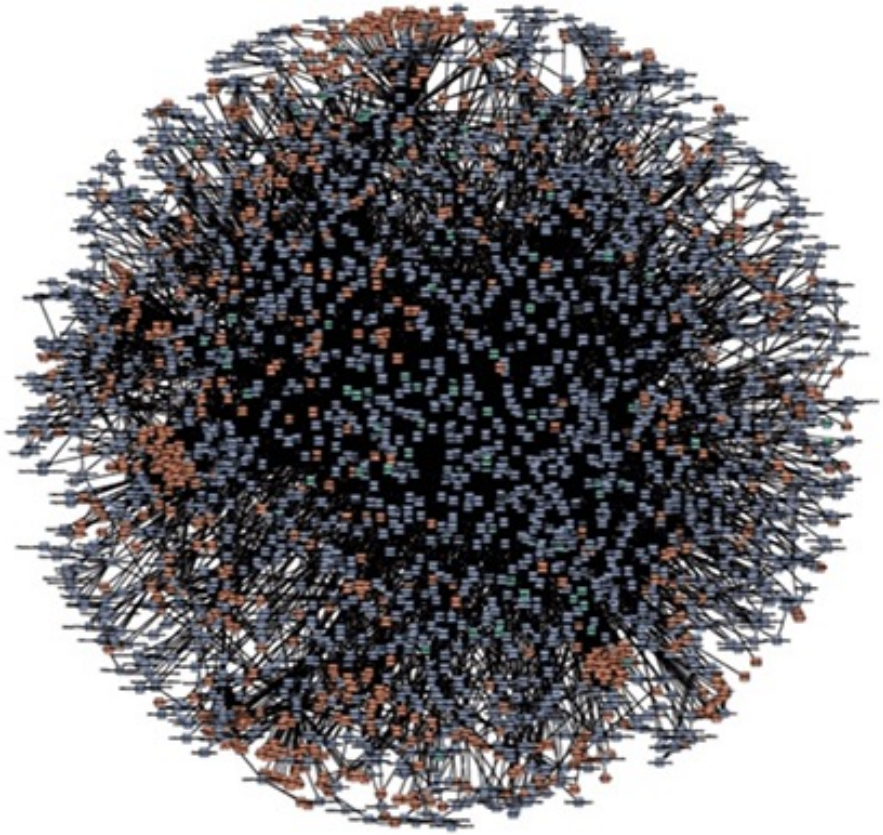
to the SOA stack

Legacy systems migration remains essential to ease their maintenance and make them more evolvable without loosing their business values

Migrating Legacy Systems to SOA

https://www.divante.com/blog/10-companies-that-implemented-the-microservice-architecture-and-paved-the-way-for-others

Migrating Legacy Systems to SOA



amazon.com

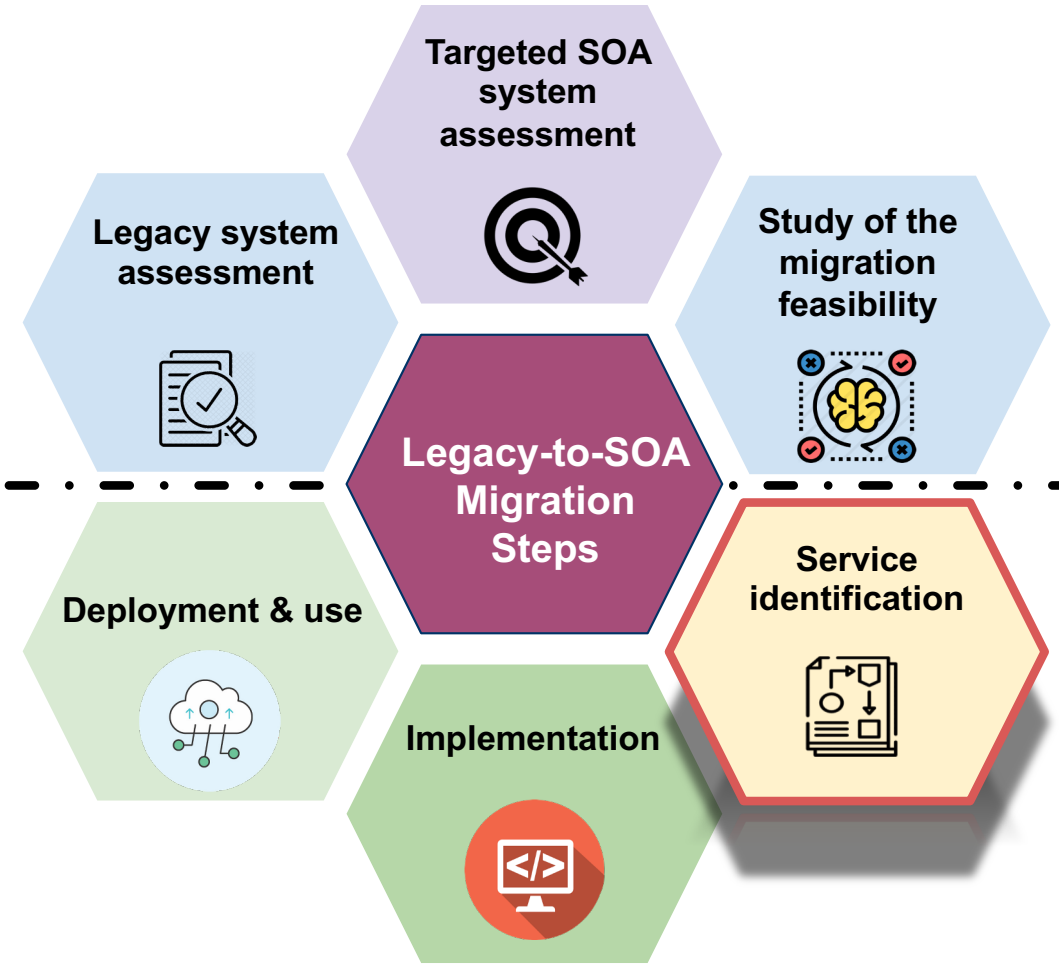
NETFLIX

https://www.divante.com/blog/10-companies-that-implemented-the-microservice-architecture-and-paved-the-way-for-others

Migration Steps

Planification of the migration

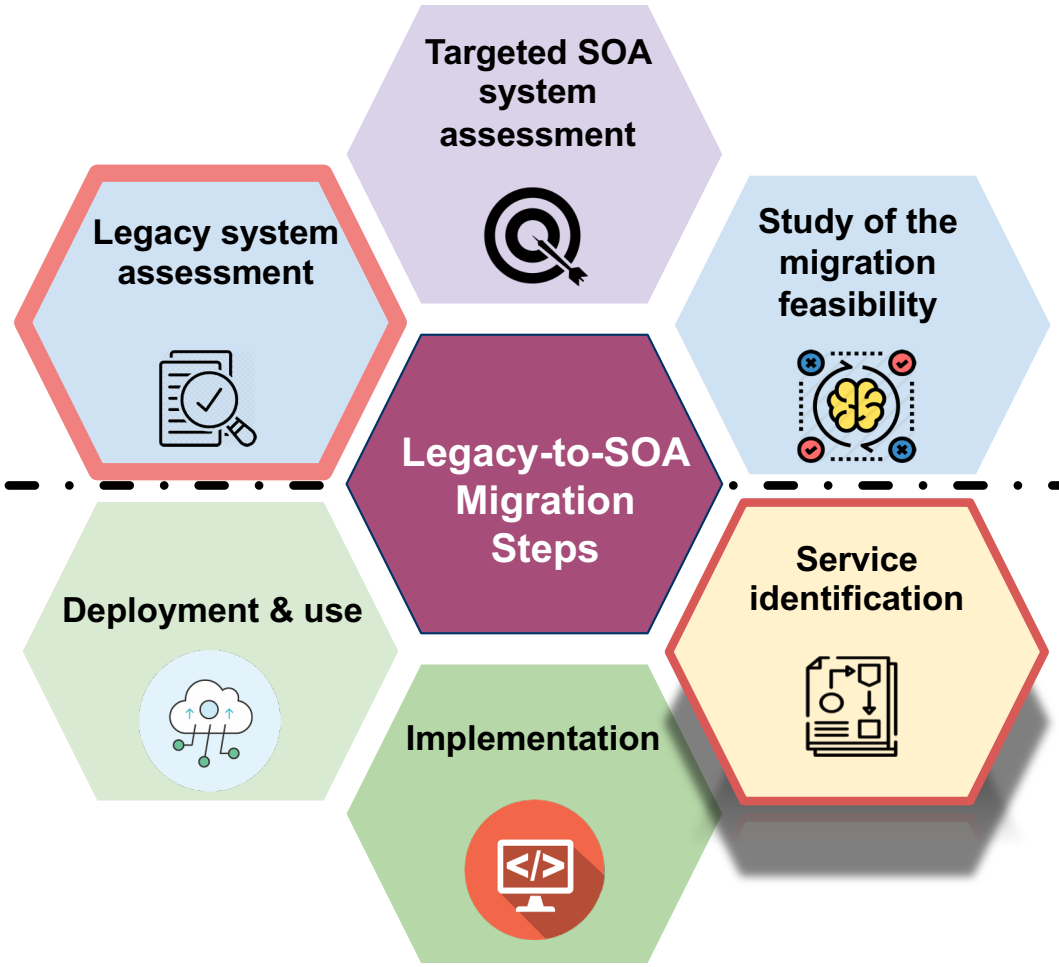
Implementation



Migration Steps

Planification of the migration

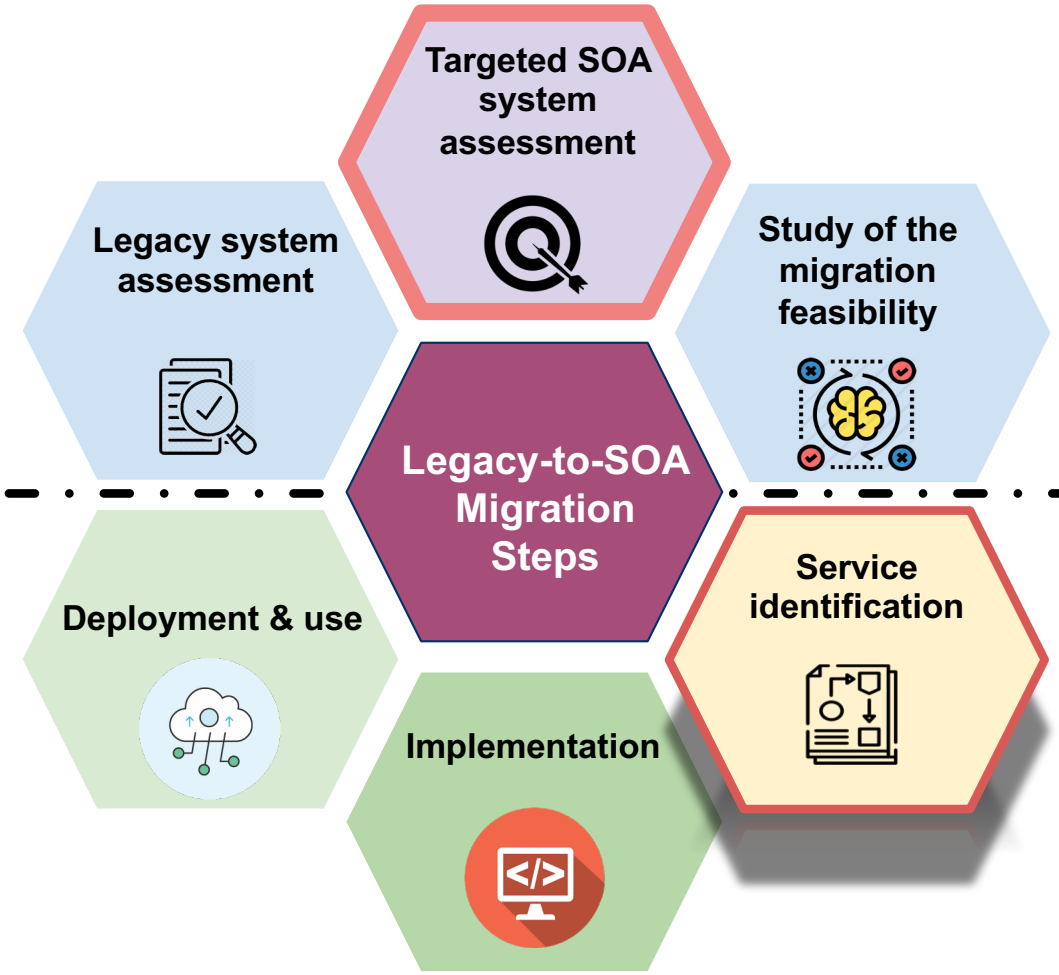
Implementation



Migration Steps

Planification of the migration

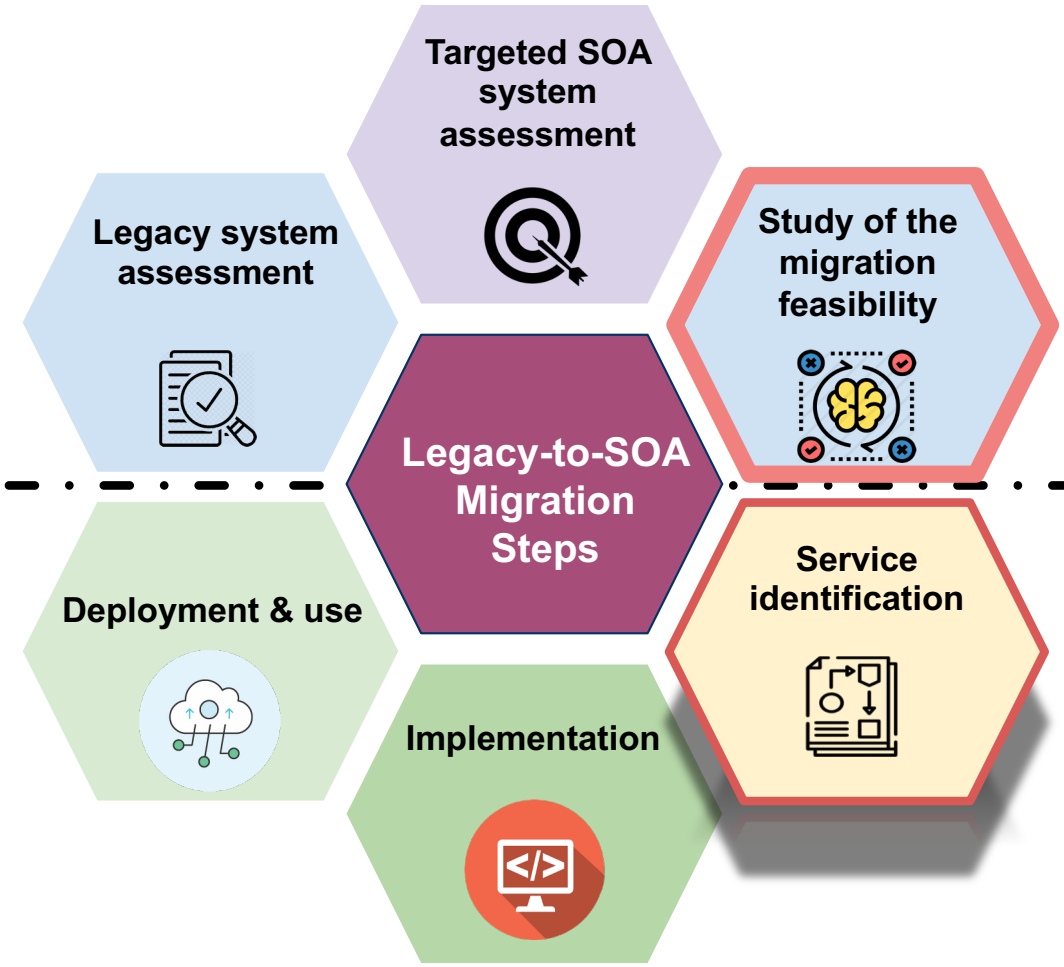
Implementation



Migration Steps

Planification of the migration

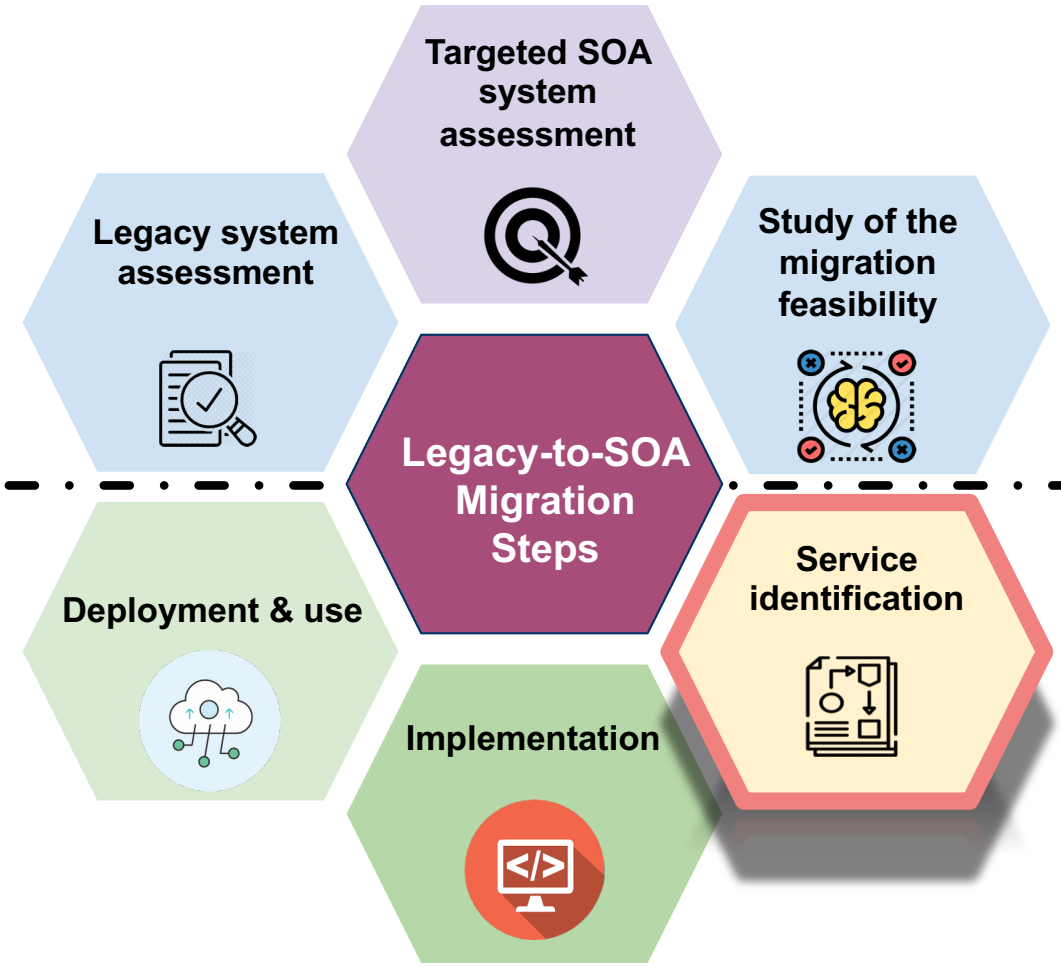
Implementation



Migration Steps

Planification of the migration

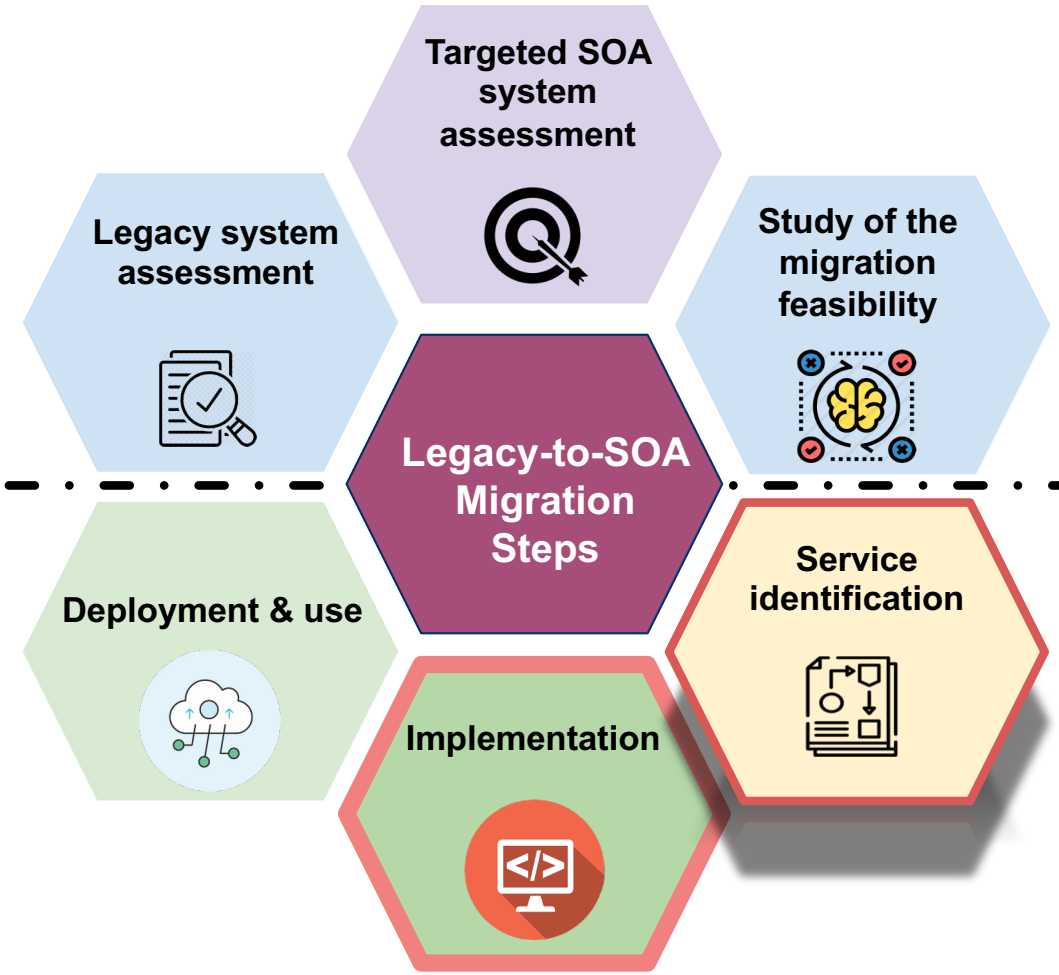
Implementation



Migration Steps

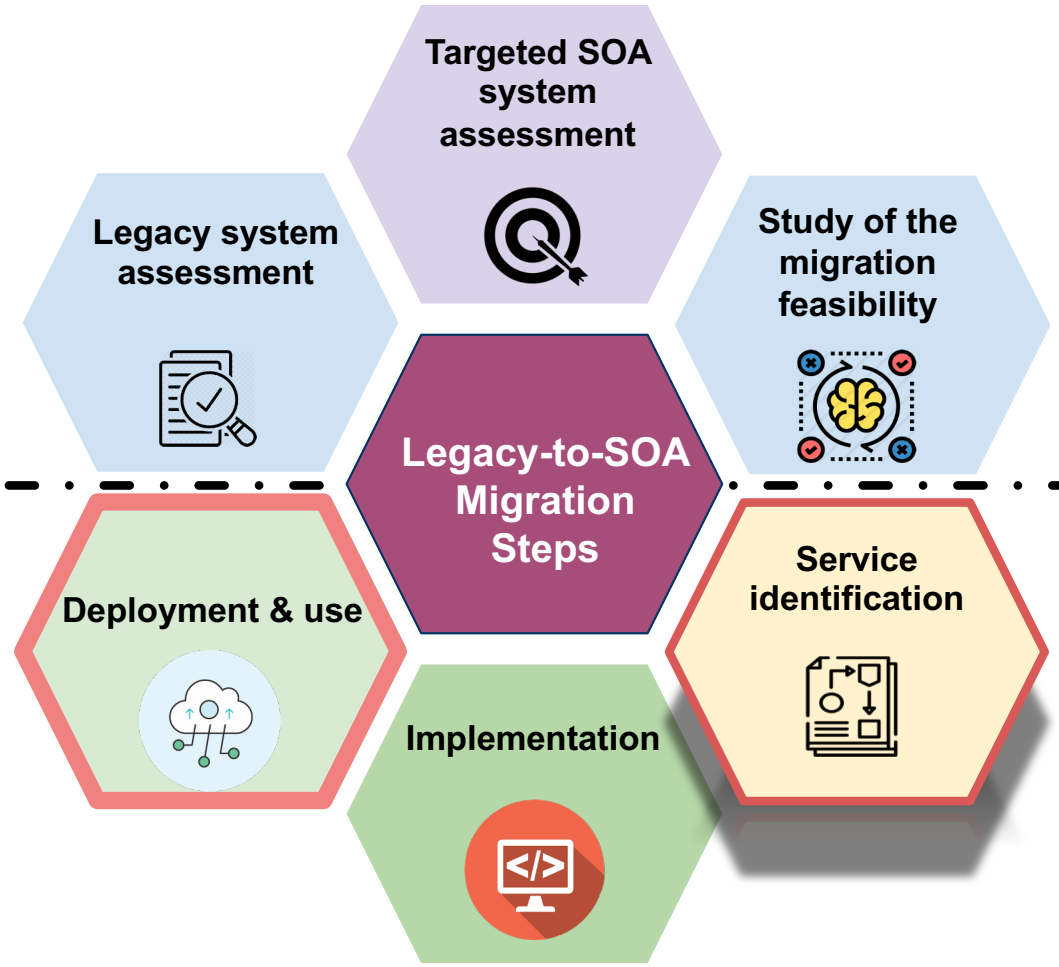
Planification of the migration

Implementation



Migration Steps

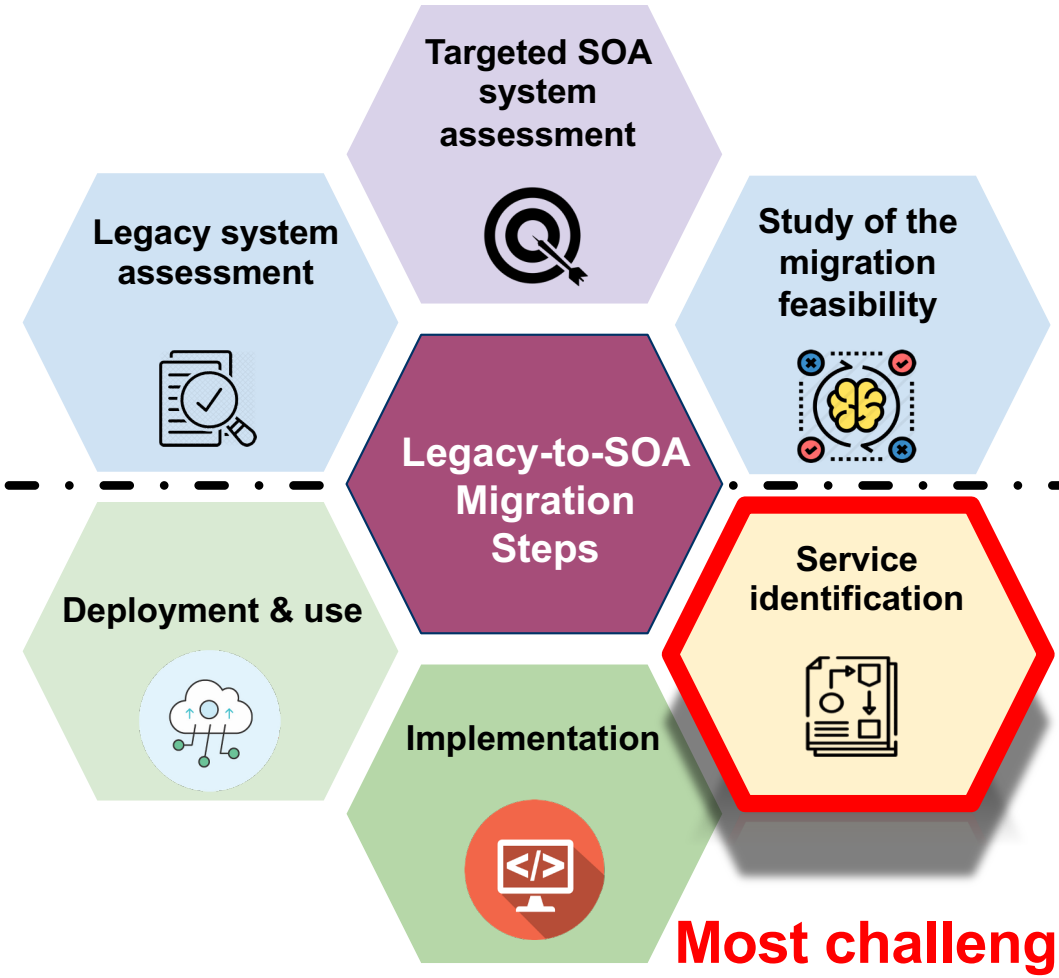
Planification of the migration



Implementation

Migration Steps

Planification of the migration



Implementation

Most challenging phase

Service Identification to Support the Migration of Legacy Systems

Manel Abdellatif

POLYTECHNIQUE MONTRÉAL

Imen Trabelsi

ÉTS
Le génie pour l'industrie

Yann-Gaël Guéhéneuc
Concordia University

Naouel Moha
ÉTS

Hafedh Mili
UQAM

Ghizlane El Boussaidi
ÉTS

Jean Privat
UQAM

Sébastien Mosser
UQAM

Service Identification Approaches

Gap between academia and industry

Limited identification accuracy

Lack of automation

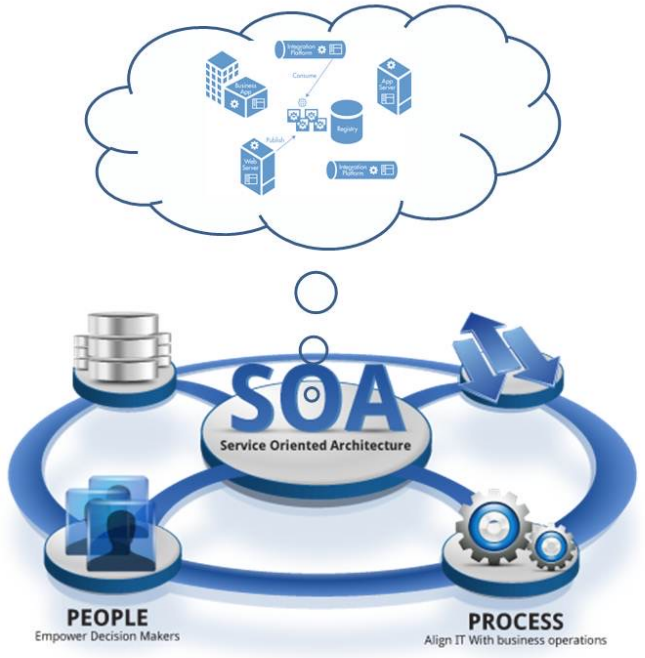
Lack of type-sensitive SIAs

Input types & availability

Service Identification

- Difficult Phase of the migration to SOA
  - The services identified must meet a set of requirements related to their functionality, quality of service and ease of use

- Some research on the subject but little knowledge of industry practices



Industry Practices on Migration

Survey

Publication of the survey

Target companies for legacy modernization

accenture
FRESCHE LEGACY
IBM
Infosys
Hydro Québec
TATA CONSULTANCY SERVICES
benchmark
hp
NTT DATA

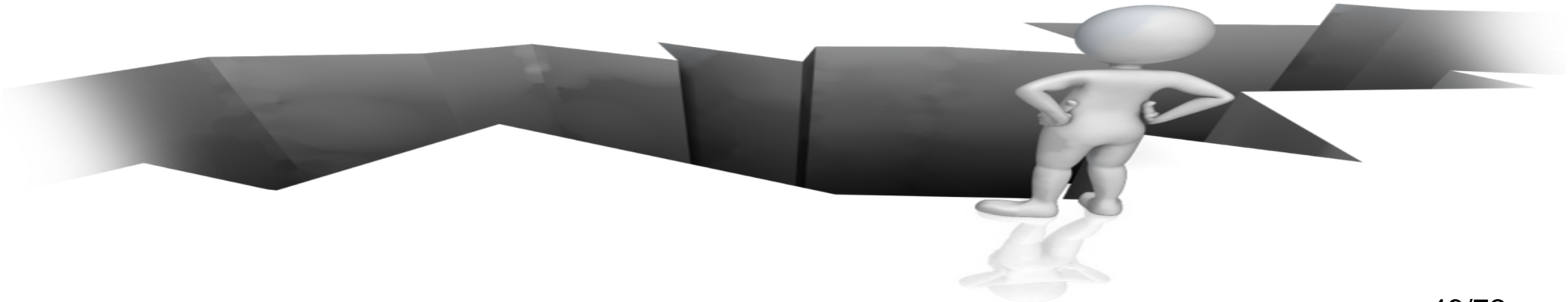
Voluntary participation (no incentives)

We reached around 298 practitioners

45 participants fully completed the survey + 8 interviews

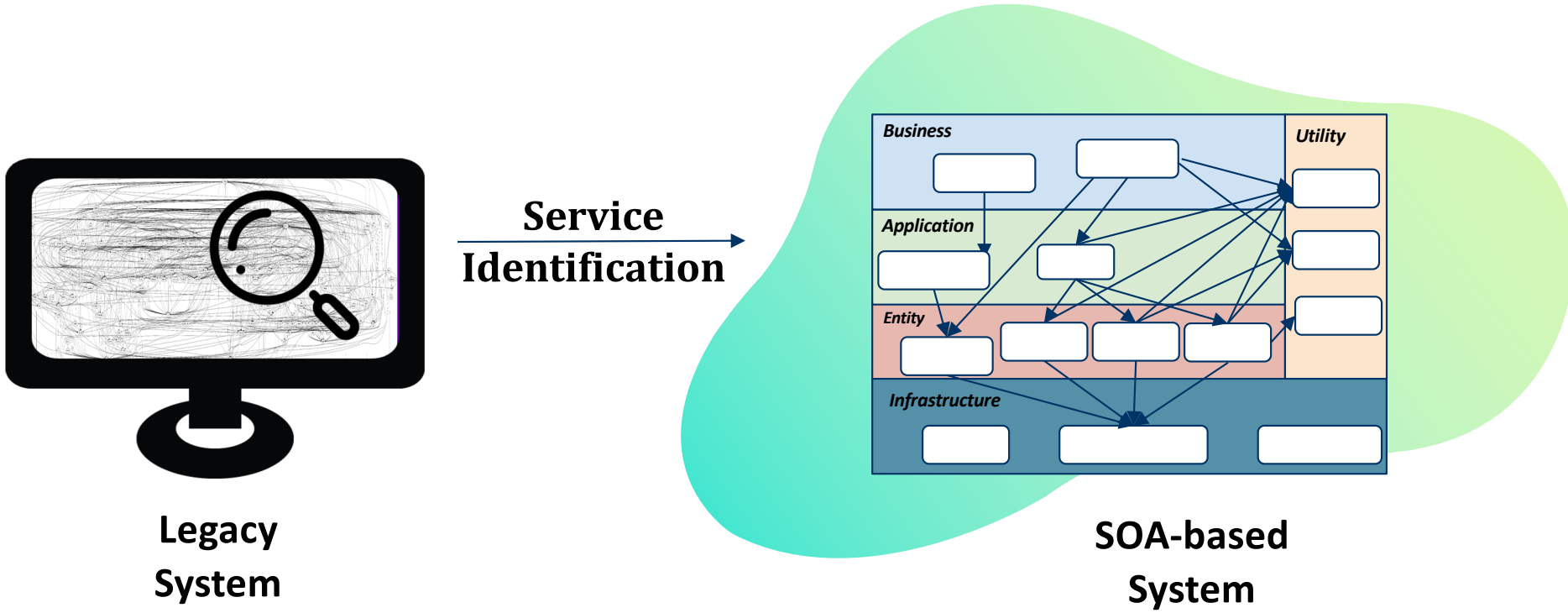
Industry Practices on Migration

- **Gap** between academia and industry
- **Poor knowledge transfer** between academia and industry in the context of legacy-to-SOA migration
- **Lack** of **cost-effective** academic service identification techniques
- Lack of **validation** on real **enterprise-scale** systems

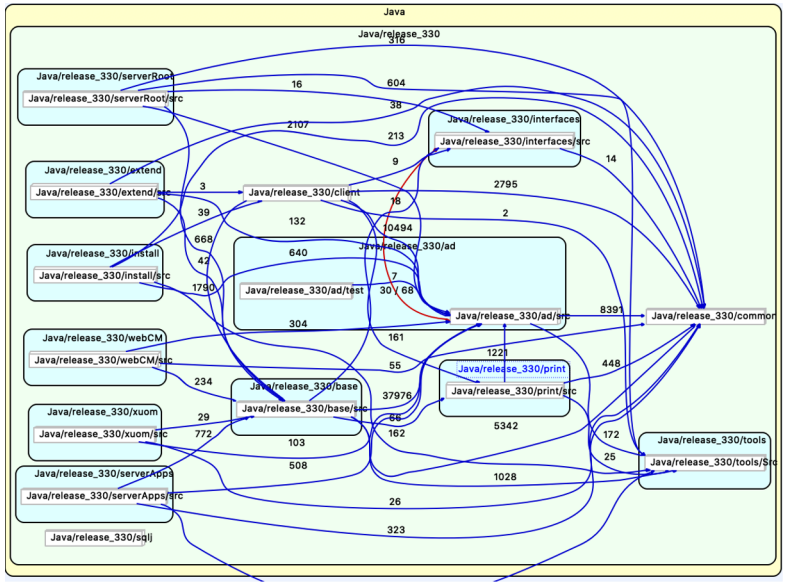


Service Identification

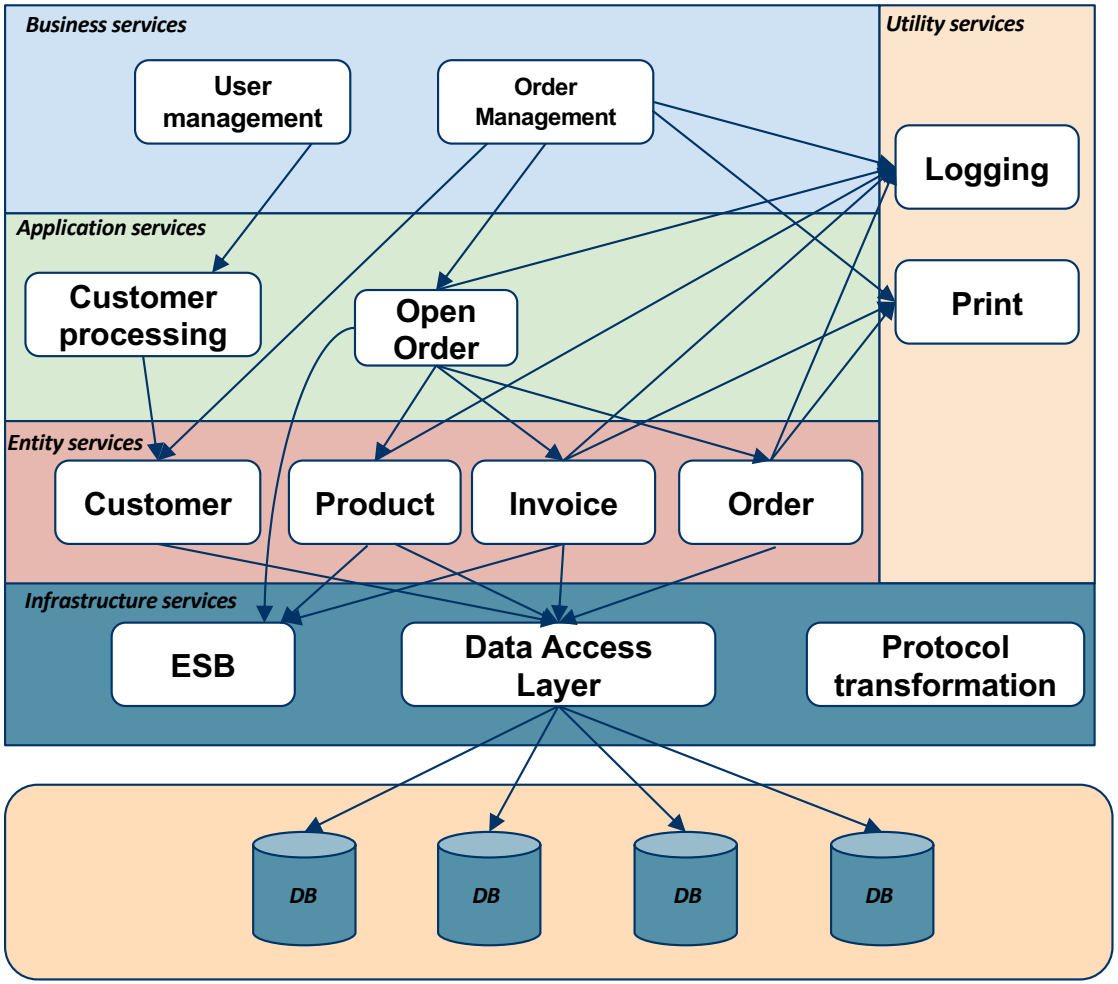
- Supporting legacy-to-SOA migration with a highly automated type-sensitive SI approach that relies on static and semantic analysis of legacy systems



SI should depend on service types



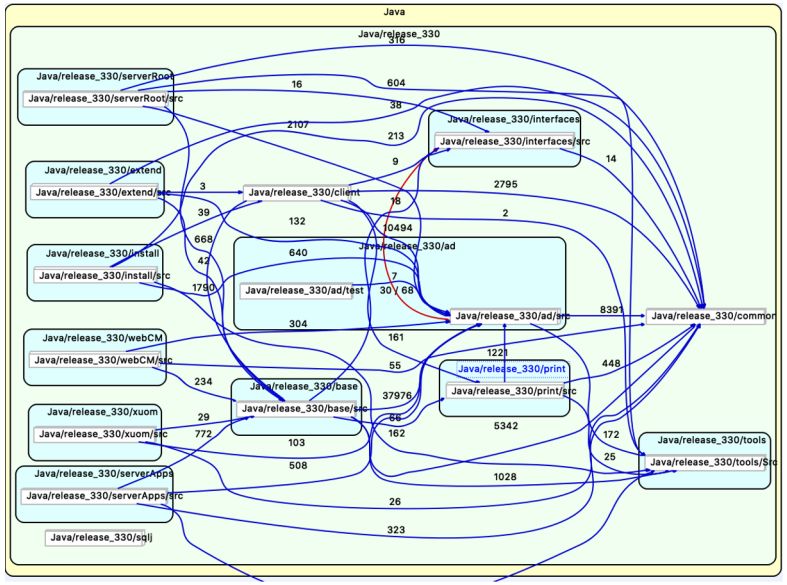
Migration



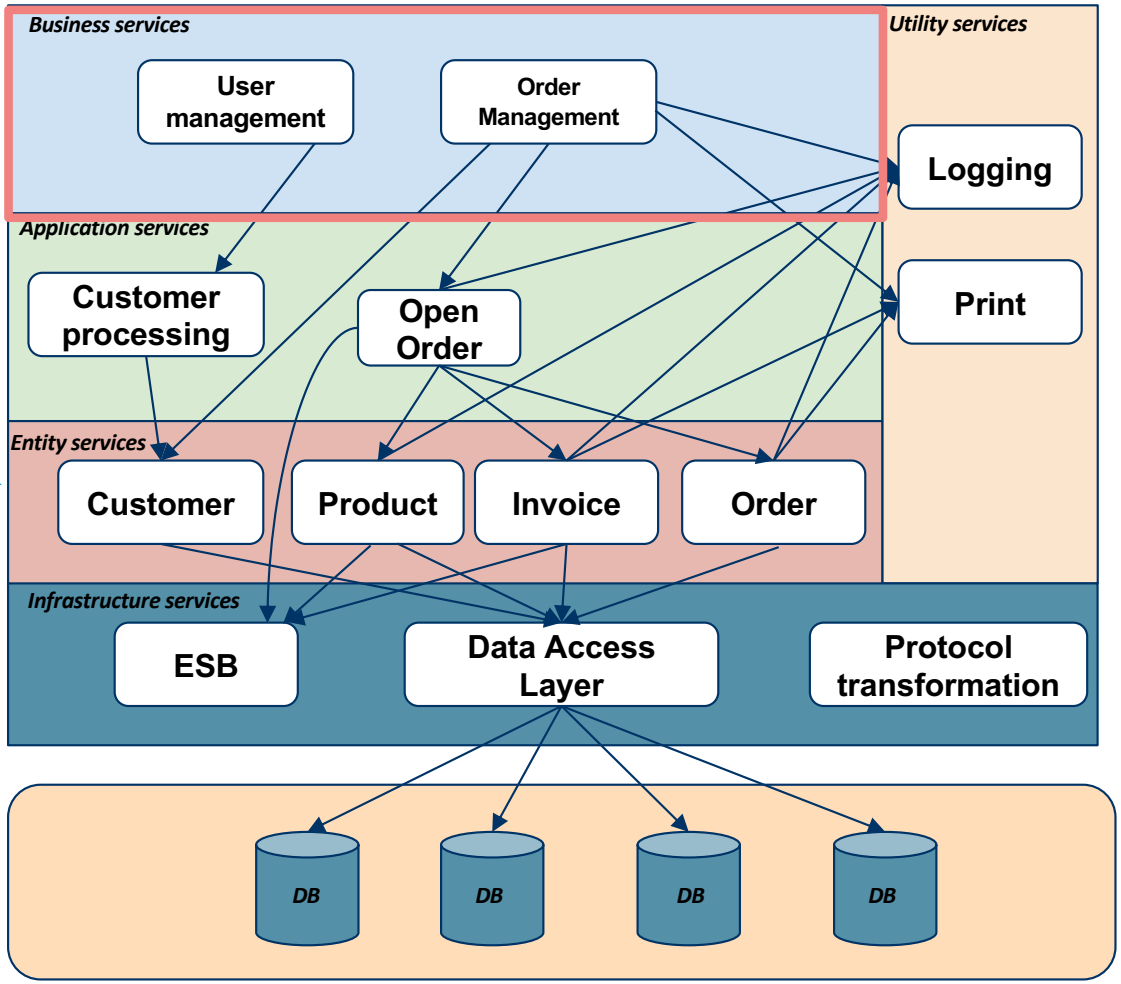
Legacy ERP

SOA-based ERP

SI should depend on service types



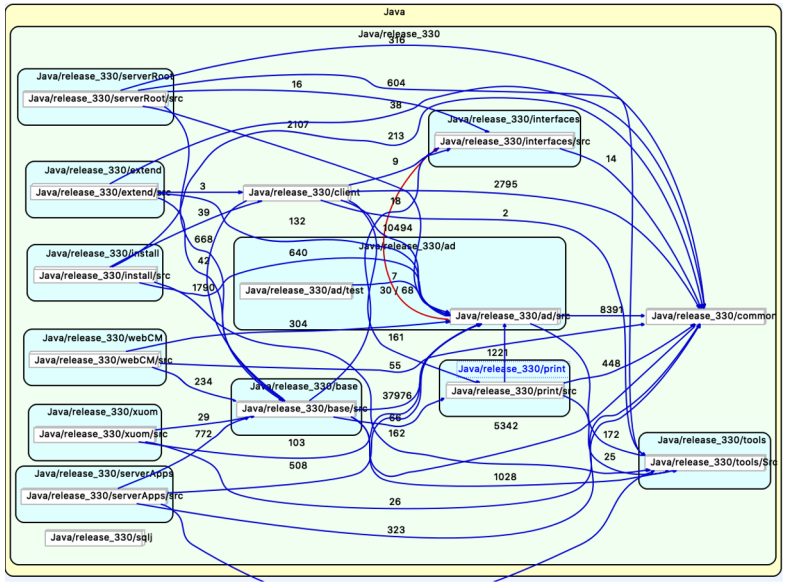
Migration



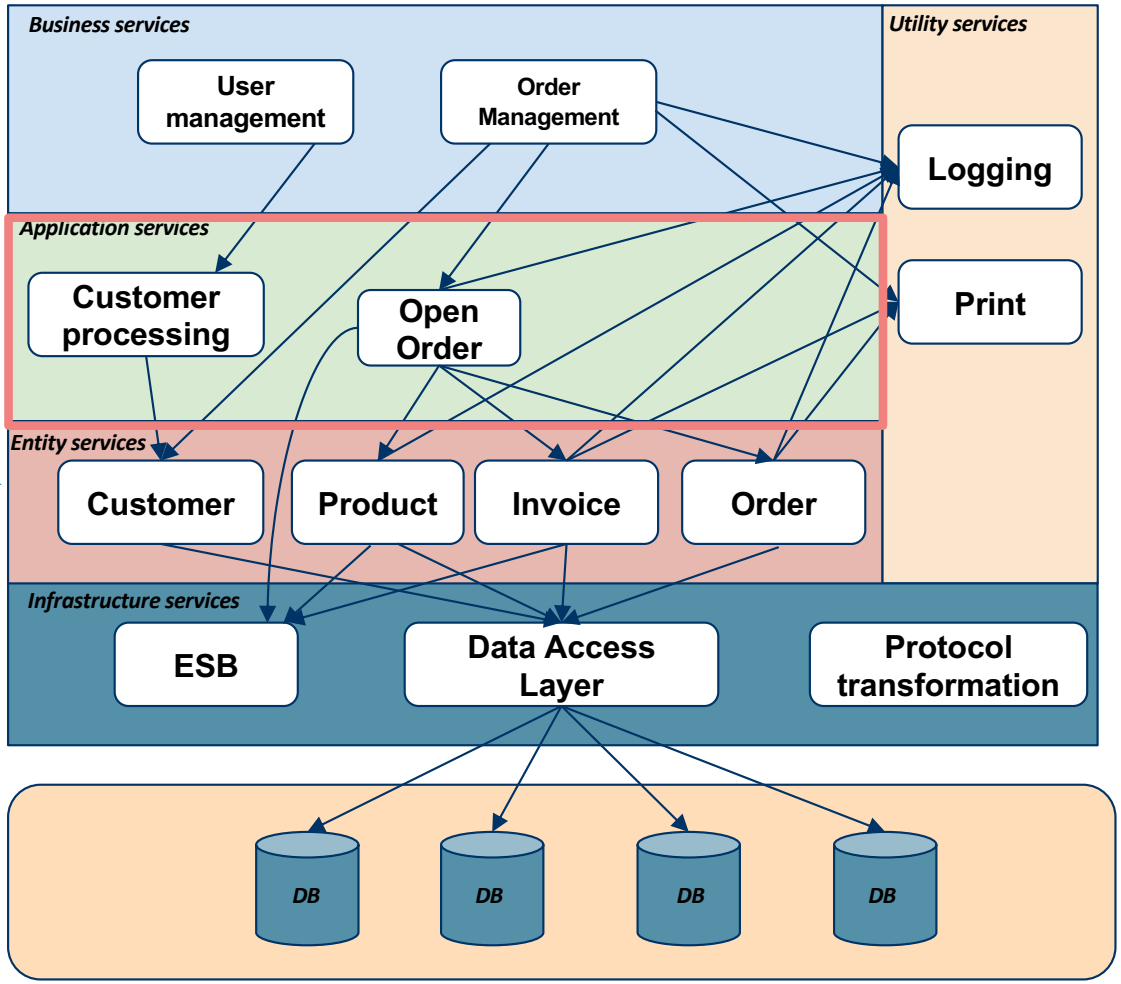
Legacy ERP

SOA-based ERP

SI should depend on service types



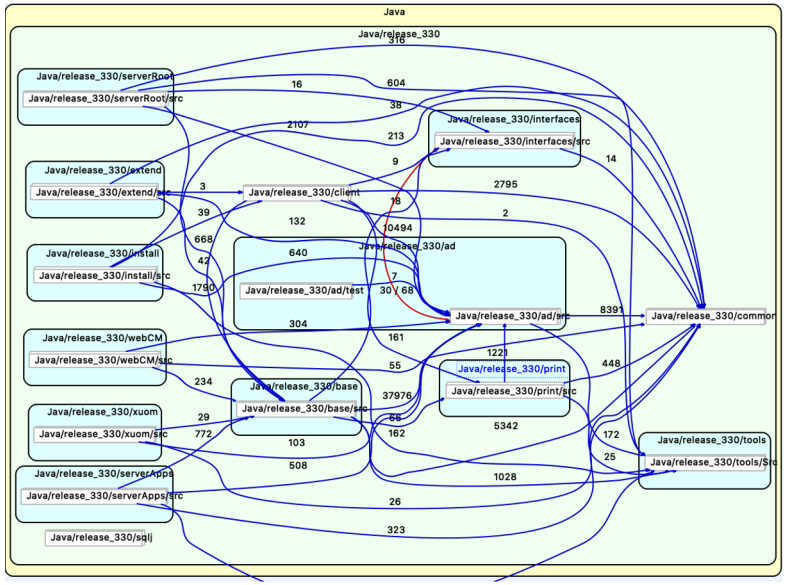
Migration



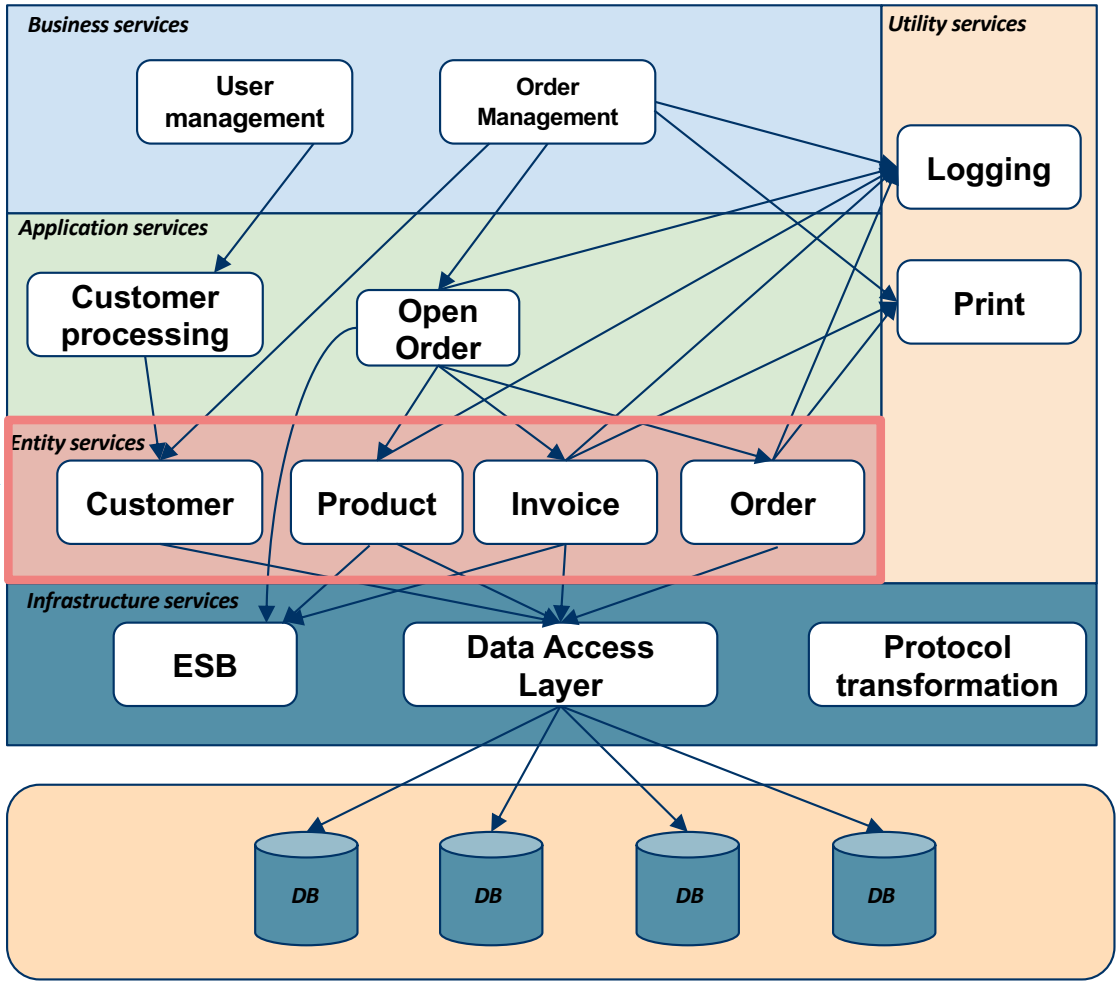
Legacy ERP

SOA-based ERP

SI should depend on service types



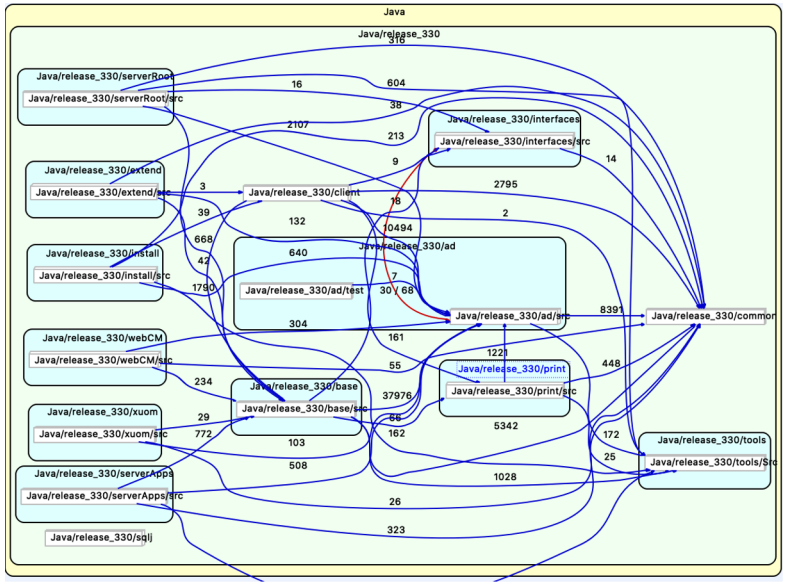
Migration



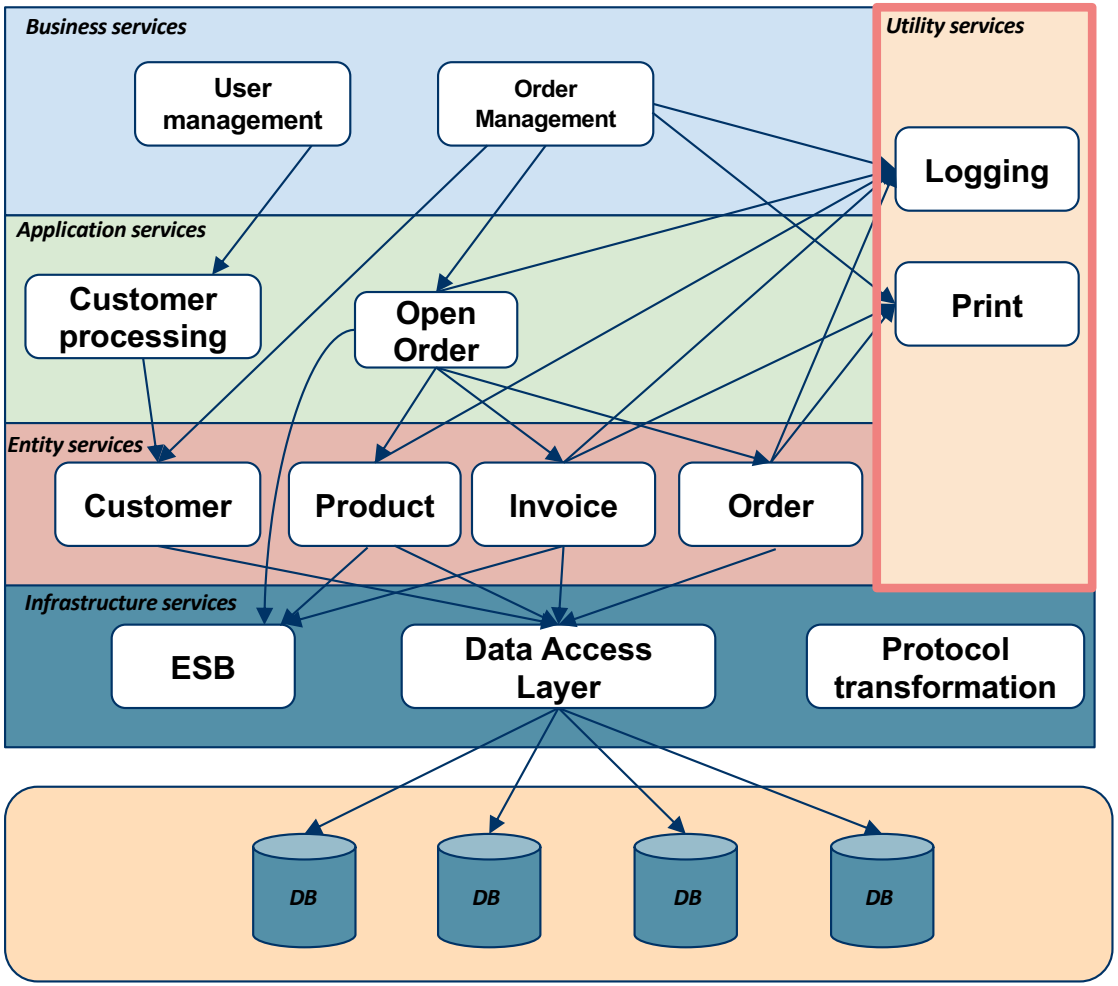
Legacy ERP

SOA-based ERP

SI should depend on service types



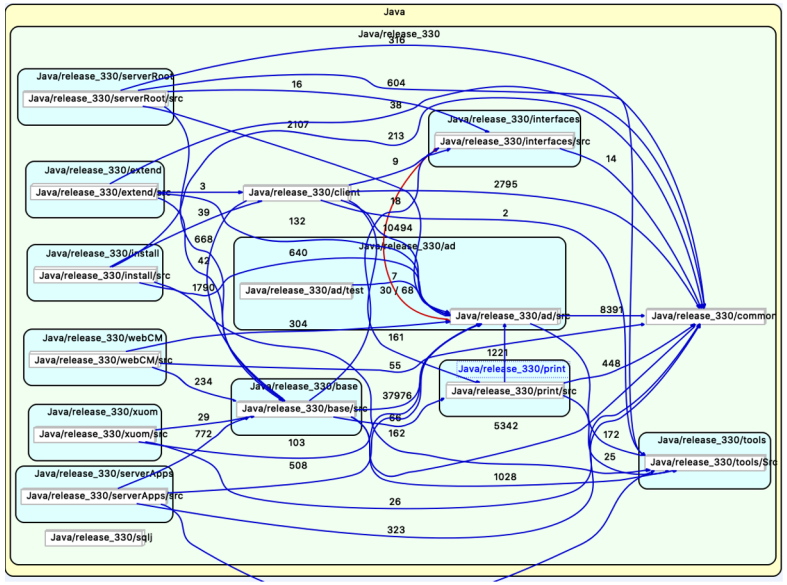
Migration



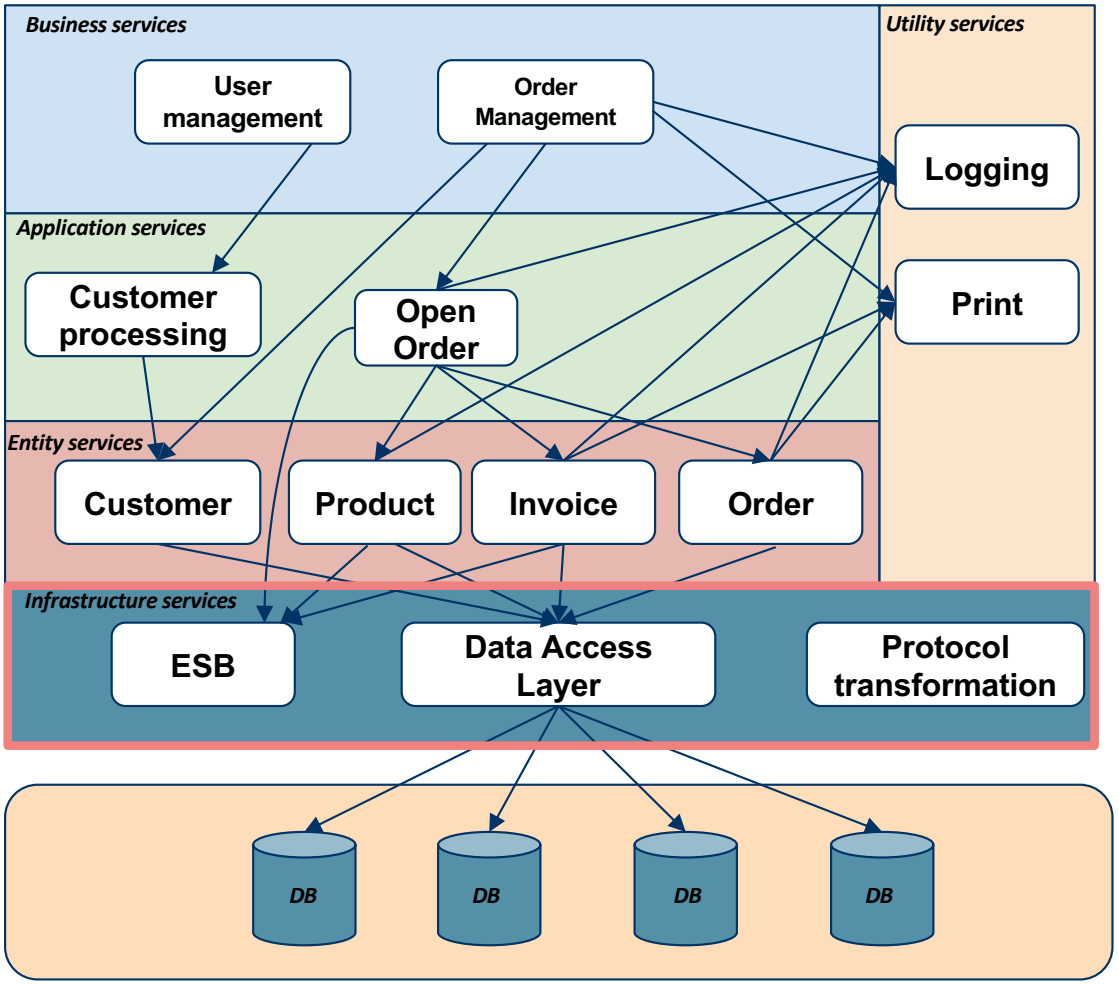
Legacy ERP

SOA-based ERP

SI should depend on service types



Migration

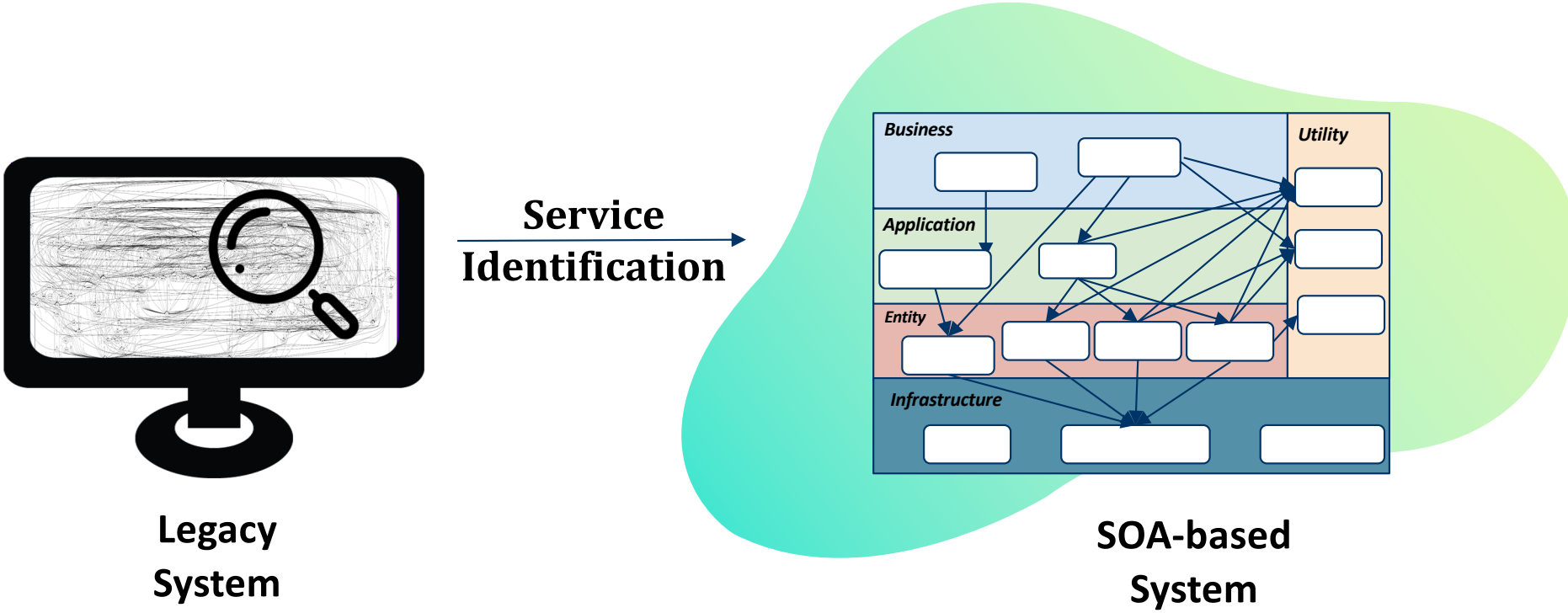


Legacy ERP

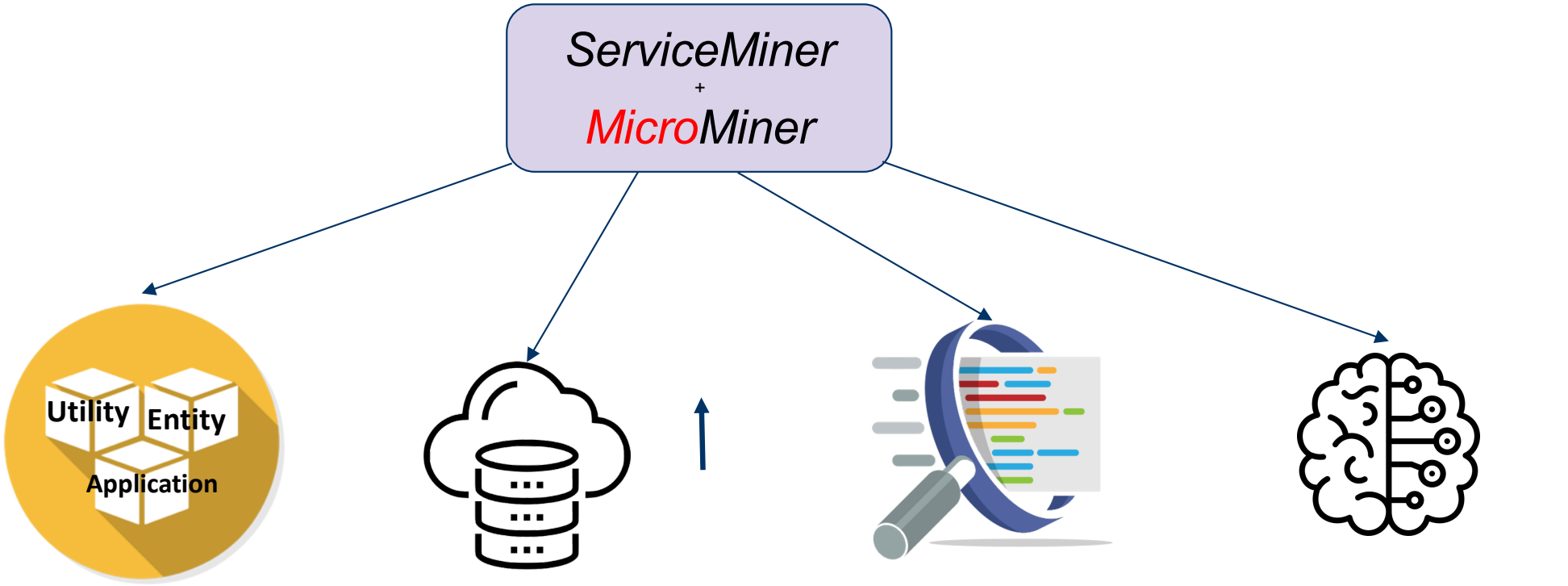
SOA-based ERP

Service Identification

- Supporting legacy-to-SOA migration with a highly automated type-sensitive SI approach that relies on static and semantic analysis of legacy systems



Service Identification



Type-sensitive SI

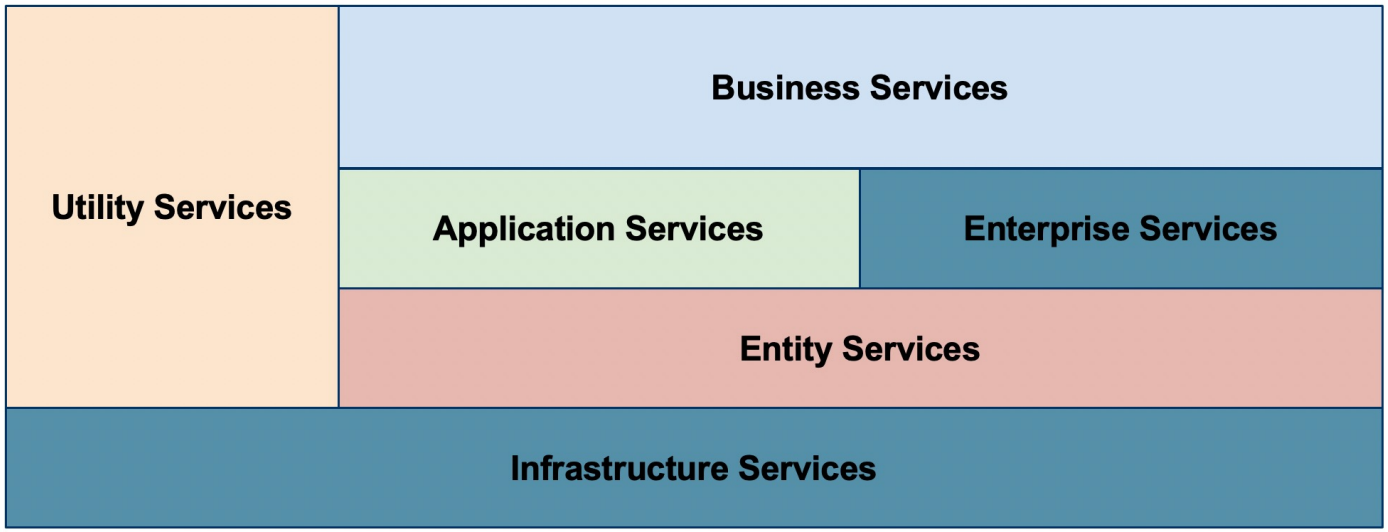
Bottom-up Approach

Static and Semantic Analysis

Machine Learning

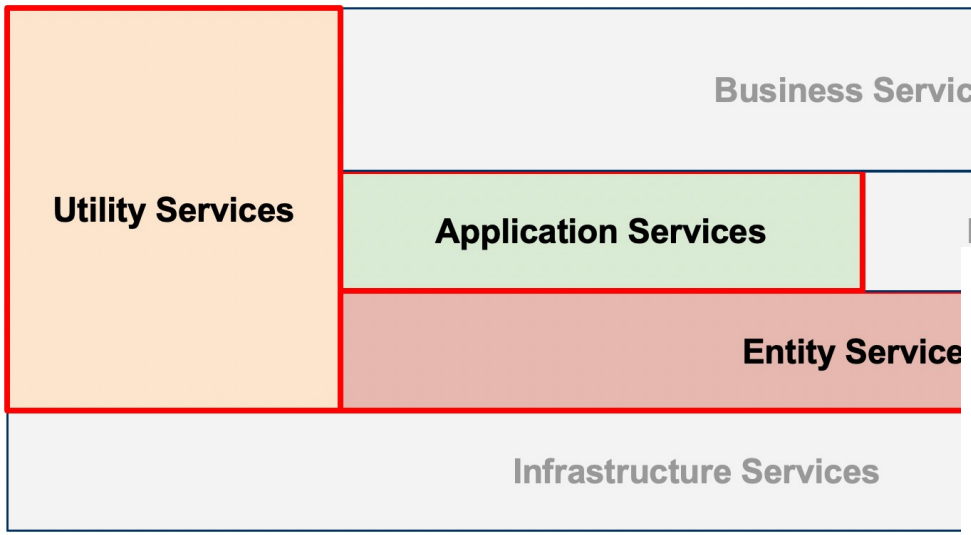
Service Identification

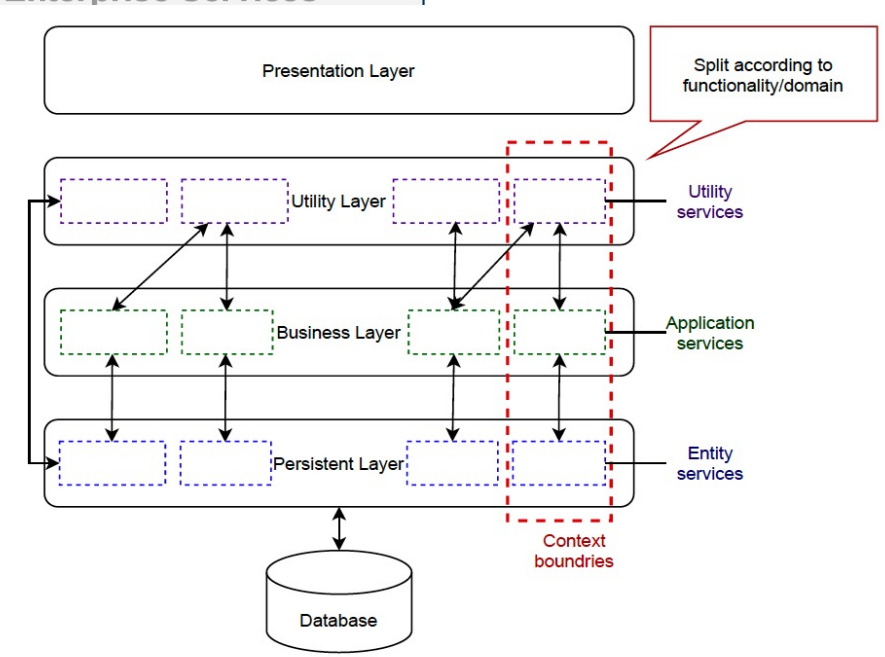
- Targeted Service Types



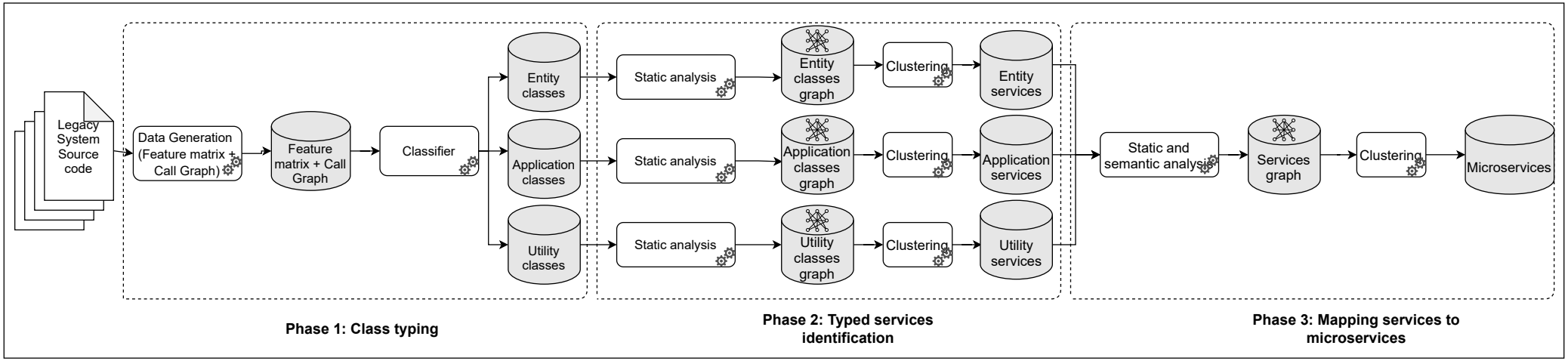
Service Identification

- Targeted Service Types





MicroMiner


Legacy System Source code
Data Generation (Feature matrix + Call Graph)
Feature matrix + Call Graph
Classifier
Entity classes
Application classes
Utility classes
Static analysis
Static analysis
Static analysis
Entity classes graph
Application classes graph
Utility classes graph
Clustering
Clustering
Clustering
Entity services
Application services
Utility services
Static and semantic analysis
Services graph
Clustering
Microservices
Phase 1: Class typing
Phase 2: Typed services identification
Phase 3: Mapping services to microservices

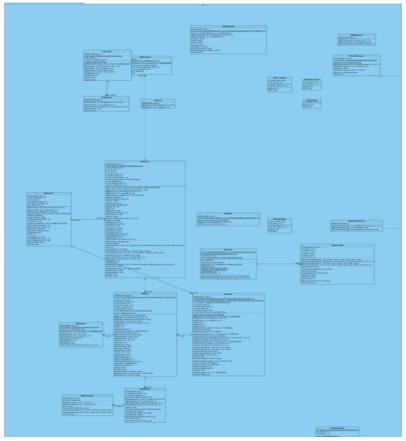
Case Studies

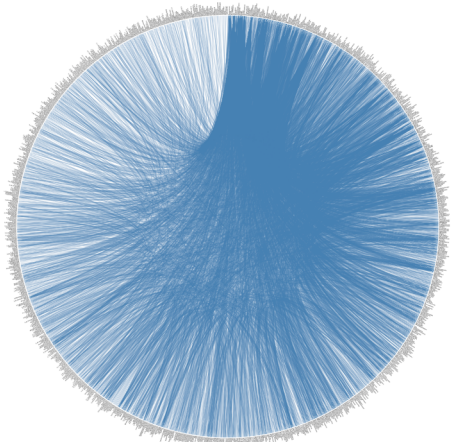
- We validate MicroMiner on **Compiere** and **POS**
  - **POS** is a Java-based point of sales system
  - **Compiere** is a **legacy** Java Enterprise **ERP** system
  - Compiere was first introduced by Aptean in **2003**
  - Provide **several features**
  - Not service-oriented
  - Support different databases
  - **2,716 classes** and 530 KLOC in Compiere
  - **54 classes** in POS

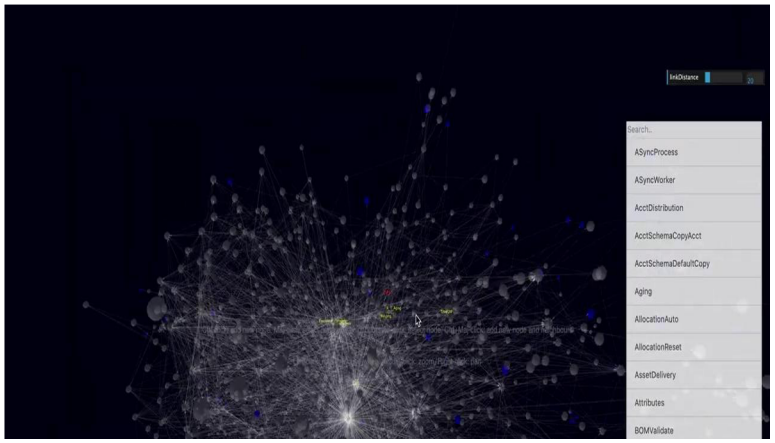
POS

Ground-Truth

- We build the **ground-truth** service-oriented architecture of the case studies to assess our approach
  - Analyzed the systems
  - Reviewed extensively the **documentation**
  - Generated **several views** of the call graphs
  - **Annotated the services manually** according to their **type**

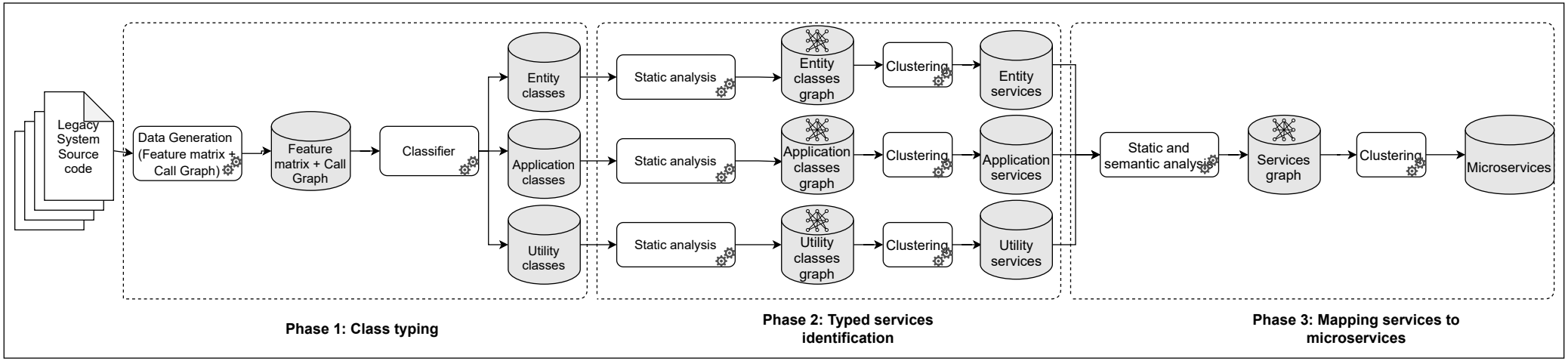


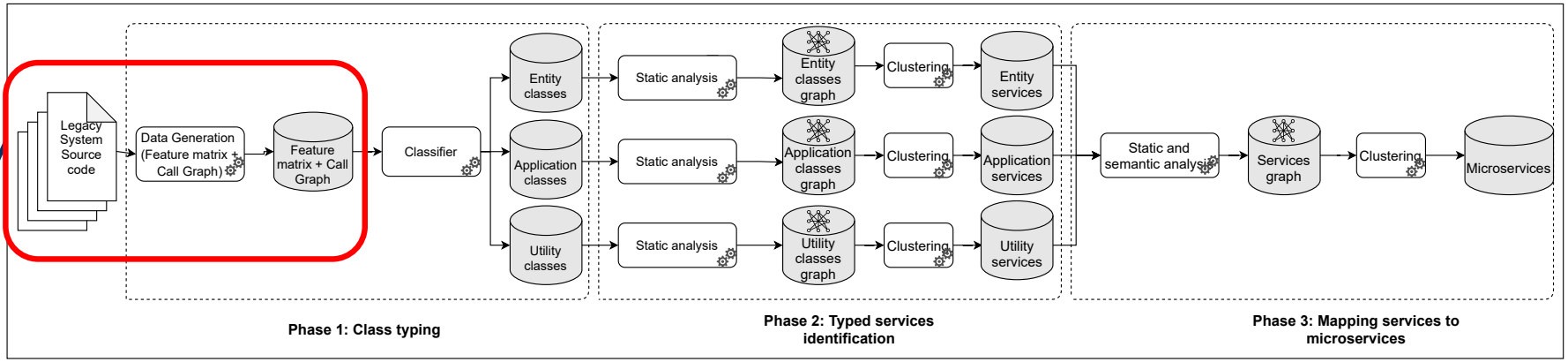


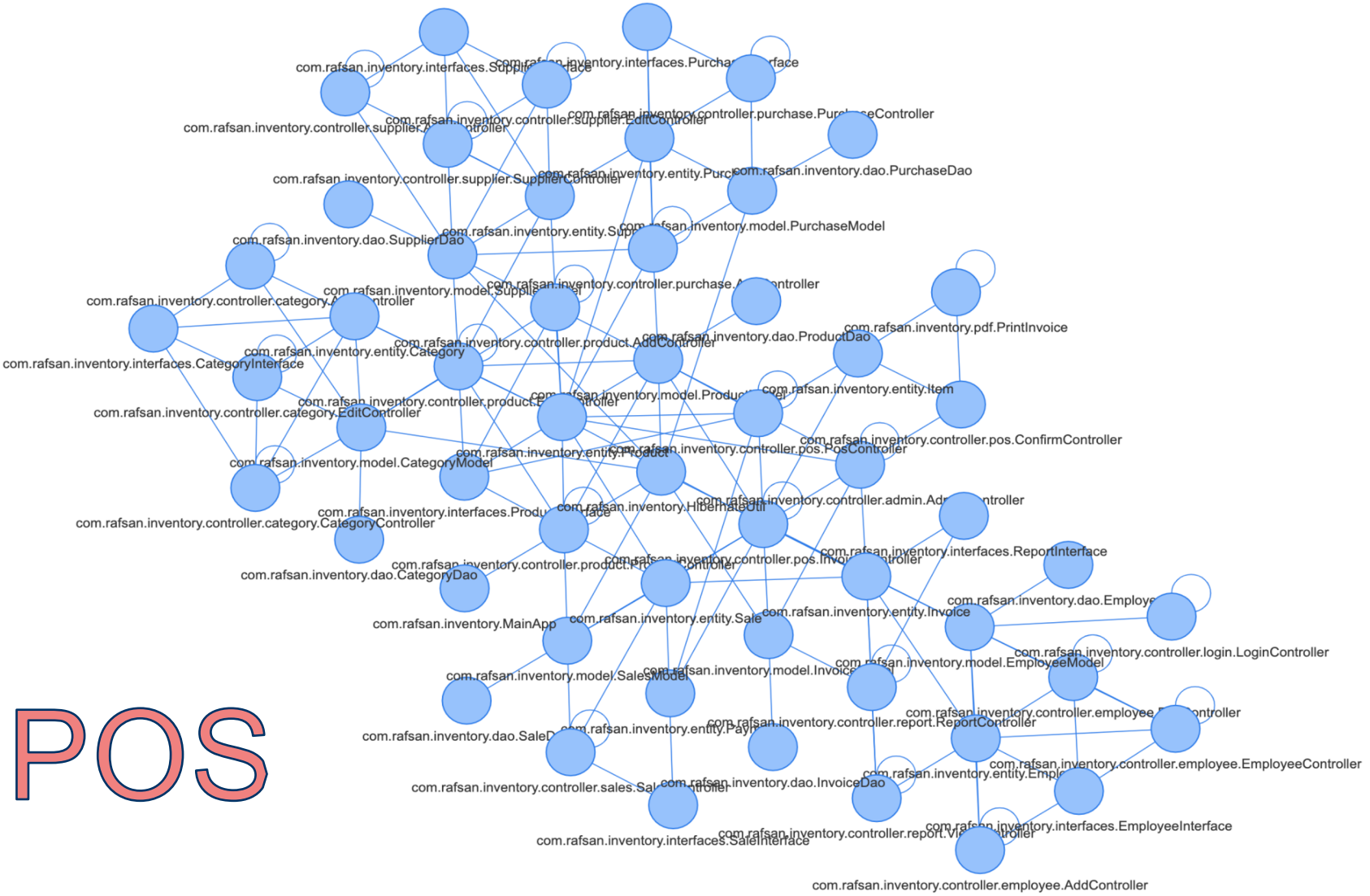


Legacy system	# System-Classes	# Entity Services	# Application Services	# Utility Services	# Microervices
Compiere	1,042	358	30	85	92
POS	55	9	10	3	9

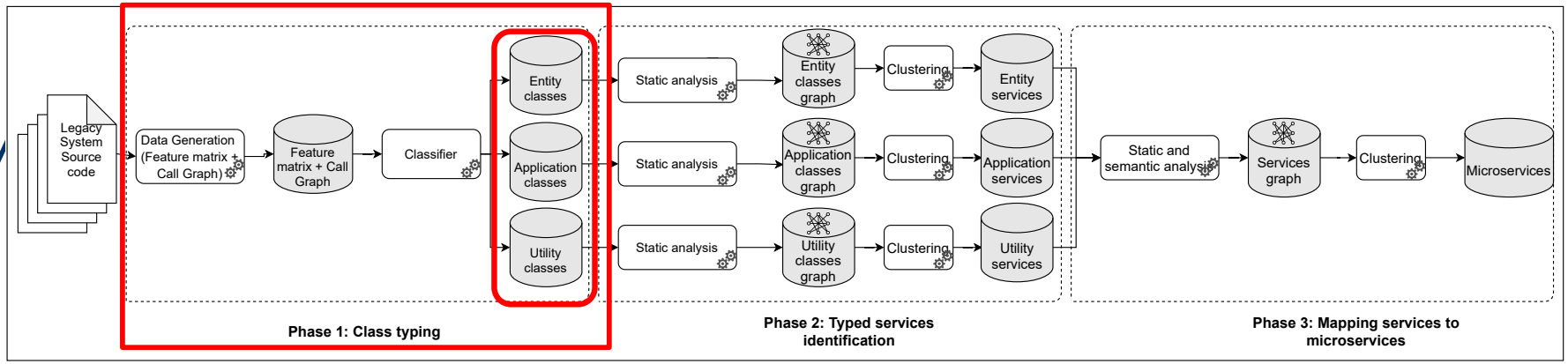
MicroMiner


Legacy System Source code
Data Generation (Feature matrix + Call Graph)
Feature matrix + Call Graph
Classifier
Entity classes
Application classes
Utility classes
Static analysis
Static analysis
Static analysis
Entity classes graph
Application classes graph
Utility classes graph
Clustering
Clustering
Clustering
Entity services
Application services
Utility services
Static and semantic analysis
Services graph
Clustering
Microservices
Phase 1: Class typing
Phase 2: Typed services identification
Phase 3: Mapping services to microservices


Legacy System Source code
Data Generation (Feature matrix + Call Graph)
Feature matrix + Call Graph
Classifier
Entity classes
Application classes
Utility classes
Static analysis
Static analysis
Static analysis
Entity classes graph
Application classes graph
Utility classes graph
Clustering
Clustering
Clustering
Entity services
Application services
Utility services
Static and semantic analysis
Services graph
Clustering
Microservices
Phase 1: Class typing
Phase 2: Typed services identification
Phase 3: Mapping services to microservices


com.rafsan.inventory.interfaces.SupplierInterface
com.rafsan.inventory.interfaces.PurchaseInterface
com.rafsan.inventory.controller.supplier.SupplierController
com.rafsan.inventory.controller.supplier.EditController
com.rafsan.inventory.controller.purchase.PurchaseController
com.rafsan.inventory.entity.Purchase
com.rafsan.inventory.dao.PurchaseDao
com.rafsan.inventory.dao.SupplierDao
com.rafsan.inventory.entity.Supplier
com.rafsan.inventory.model.PurchaseModel
com.rafsan.inventory.model.SupplierModel
com.rafsan.inventory.controller.purchase.AddController
com.rafsan.inventory.controller.category.AddController
com.rafsan.inventory.dao.ProductDao
com.rafsan.inventory.pdf.PrintInvoice
com.rafsan.inventory.entity.Category
com.rafsan.inventory.interfaces.CategoryInterface
com.rafsan.inventory.controller.product.AddController
com.rafsan.inventory.entity.Item
com.rafsan.inventory.controller.product.EditController
com.rafsan.inventory.model.ProductModel
com.rafsan.inventory.controller.category.EditController
com.rafsan.inventory.controller.pos.ConfirmController
com.rafsan.inventory.entity.Product
com.rafsan.inventory.controller.pos.PosController
com.rafsan.inventory.model.CategoryModel
com.rafsan.inventory.interfaces.ProductInterface
com.rafsan.inventory.HibernateUtil
com.rafsan.inventory.controller.admin.AdminController
com.rafsan.inventory.controller.category.CategoryController
com.rafsan.inventory.controller.pos.InvoiceController
com.rafsan.inventory.interfaces.ReportInterface
com.rafsan.inventory.dao.CategoryDao
com.rafsan.inventory.dao.EmployeeDao
com.rafsan.inventory.MainApp
com.rafsan.inventory.entity.Sale
com.rafsan.inventory.entity.Invoice
com.rafsan.inventory.controller.login.LoginController
com.rafsan.inventory.model.EmployeeModel
com.rafsan.inventory.model.InvoiceModel
com.rafsan.inventory.model.SalesModel
com.rafsan.inventory.controller.employee.EmployeeController
com.rafsan.inventory.controller.report.ReportController
com.rafsan.inventory.dao.SaleDao
com.rafsan.inventory.entity.Payment
com.rafsan.inventory.entity.Employee
com.rafsan.inventory.controller.sales.SalesController
com.rafsan.inventory.dao.InvoiceDao
com.rafsan.inventory.interfaces.EmployeeInterface
com.rafsan.inventory.controller.report.ViewController
com.rafsan.inventory.interfaces.SaleInterface
com.rafsan.inventory.controller.employee.AddController

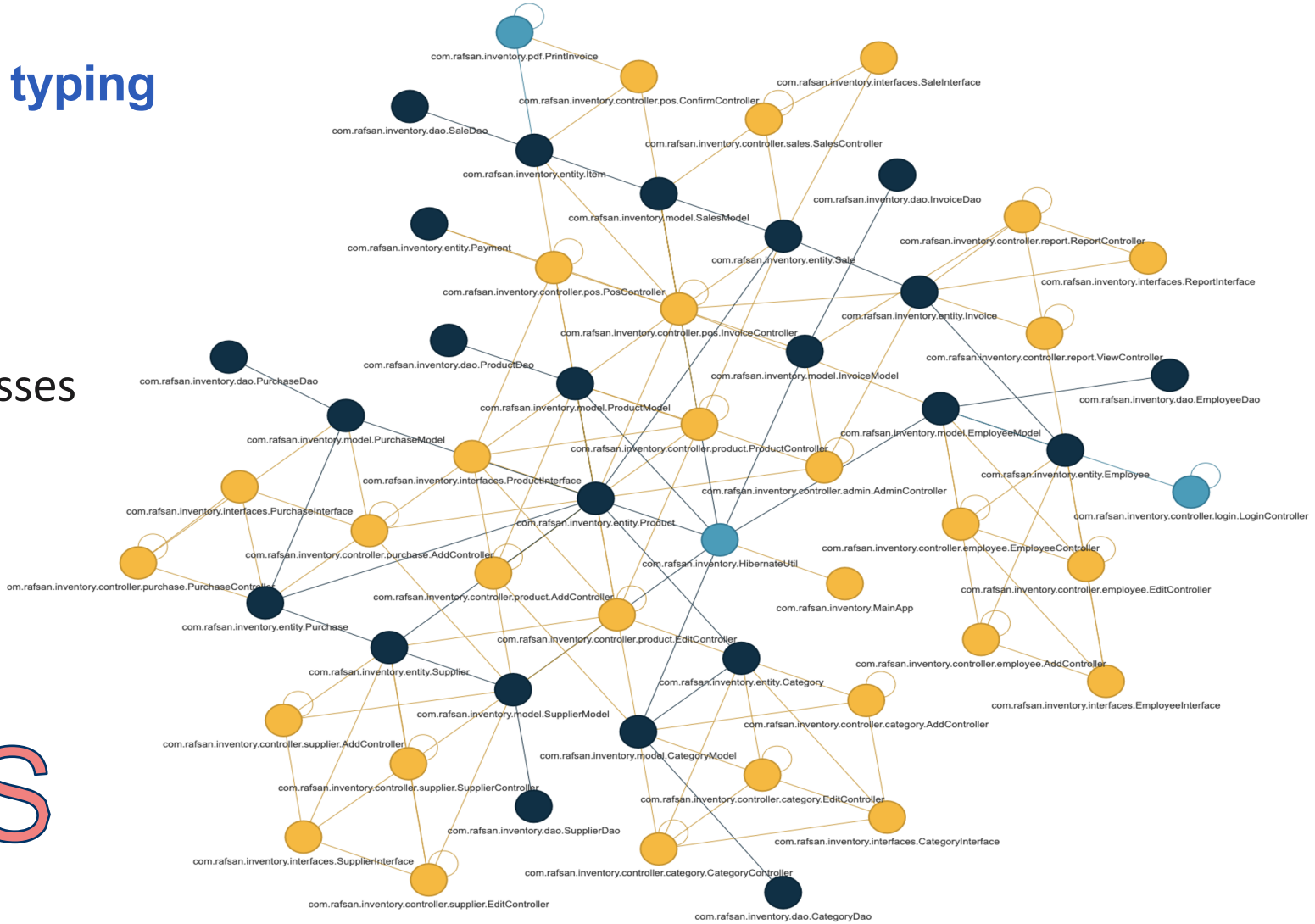
POS

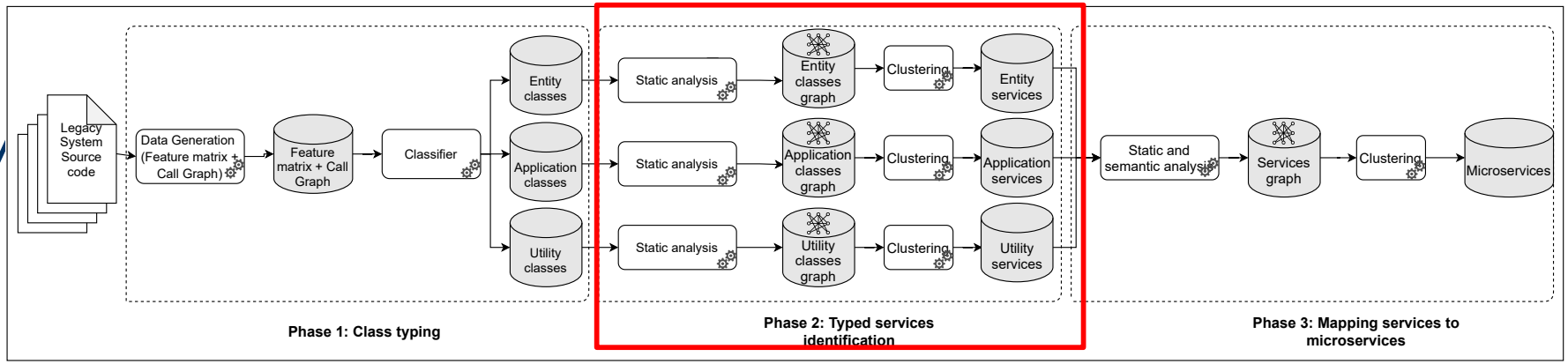


Phase 1: **Class typing**

- Utility classes
- Entity classes
- Application classes

POS

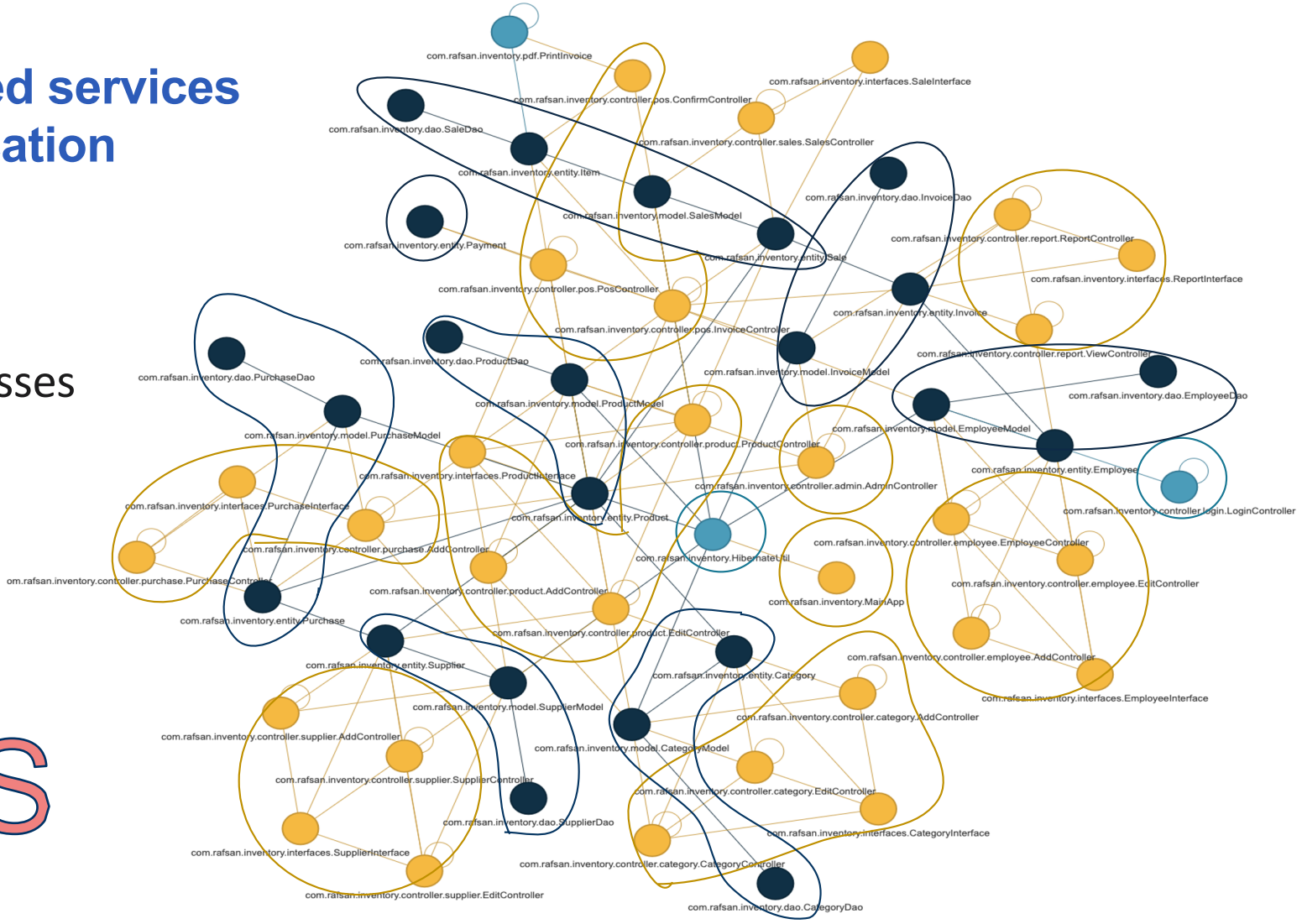


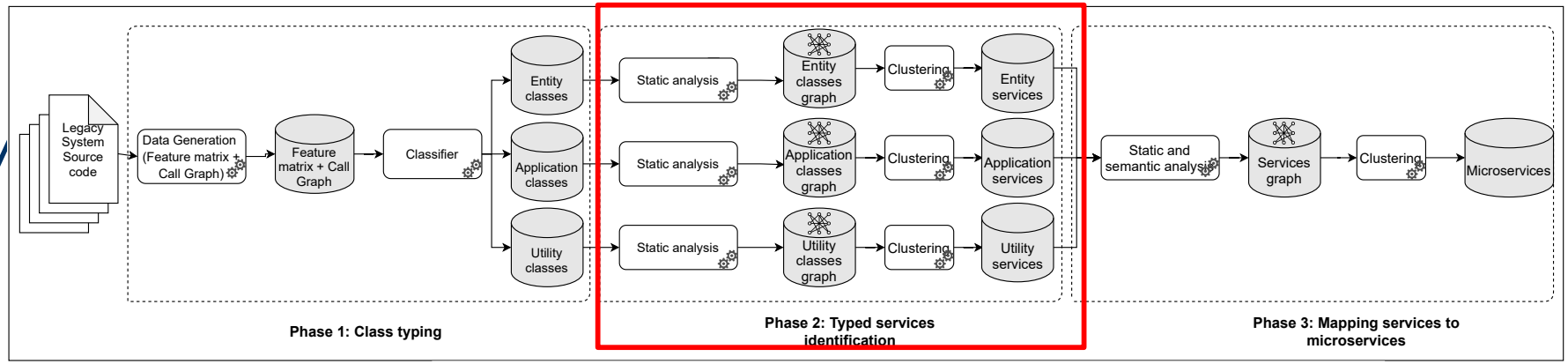


Phase 2: **Typed services identification**

- Utility classes
- Entity classes
- Application classes

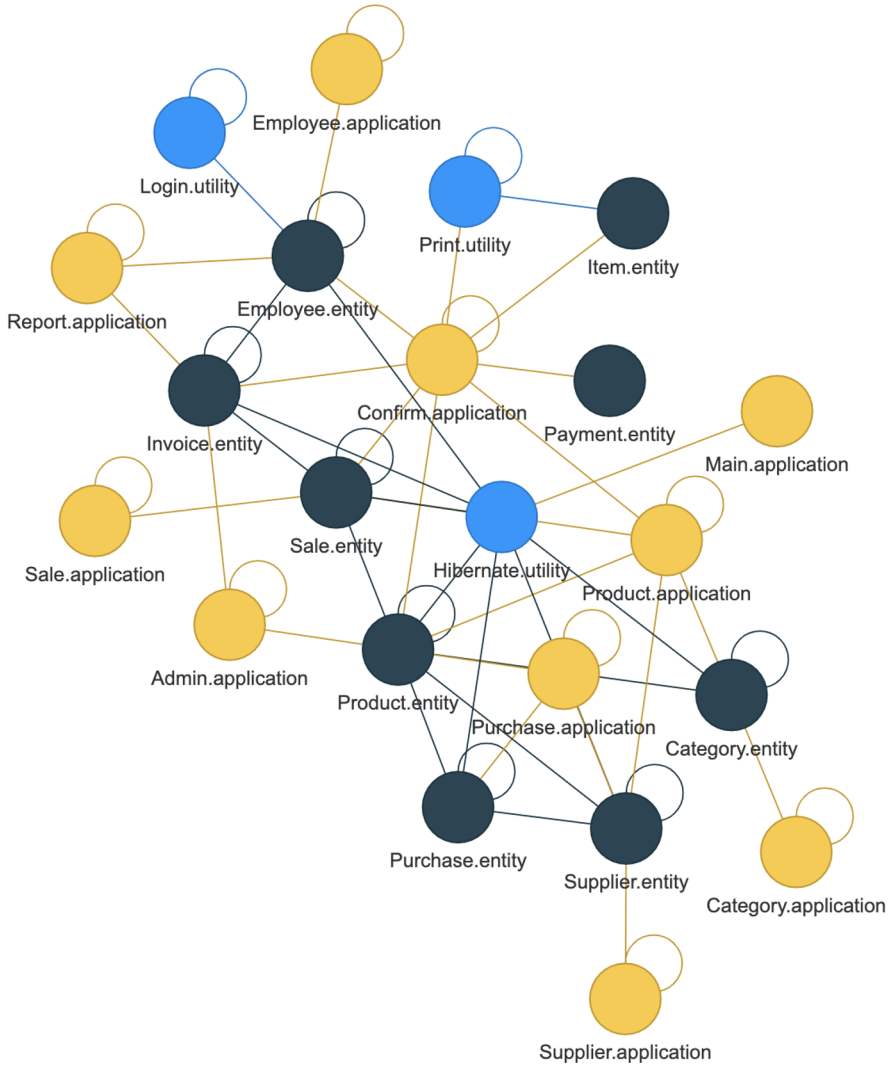
POS



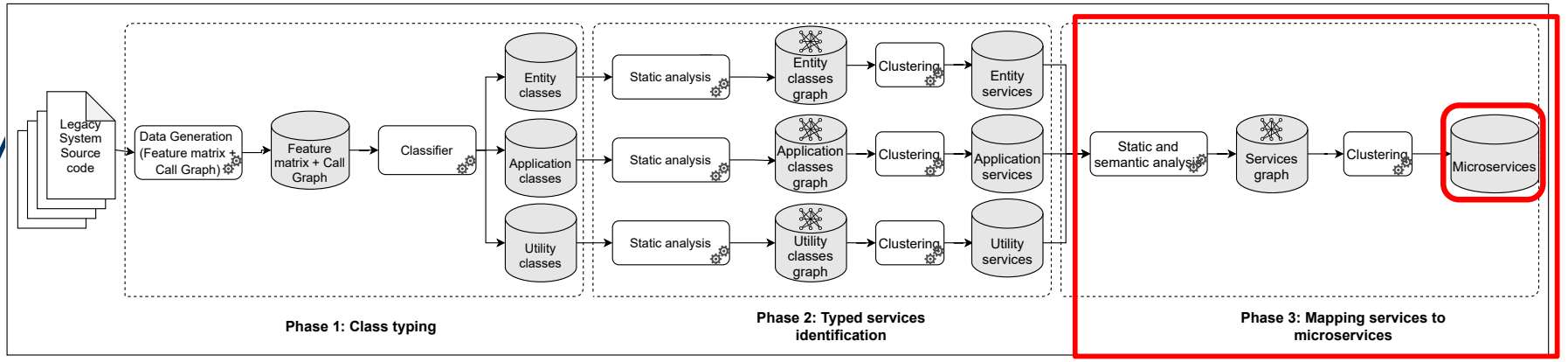


Phase 2: **Typed services identification**

- Utility classes
- Entity classes
- Application classes

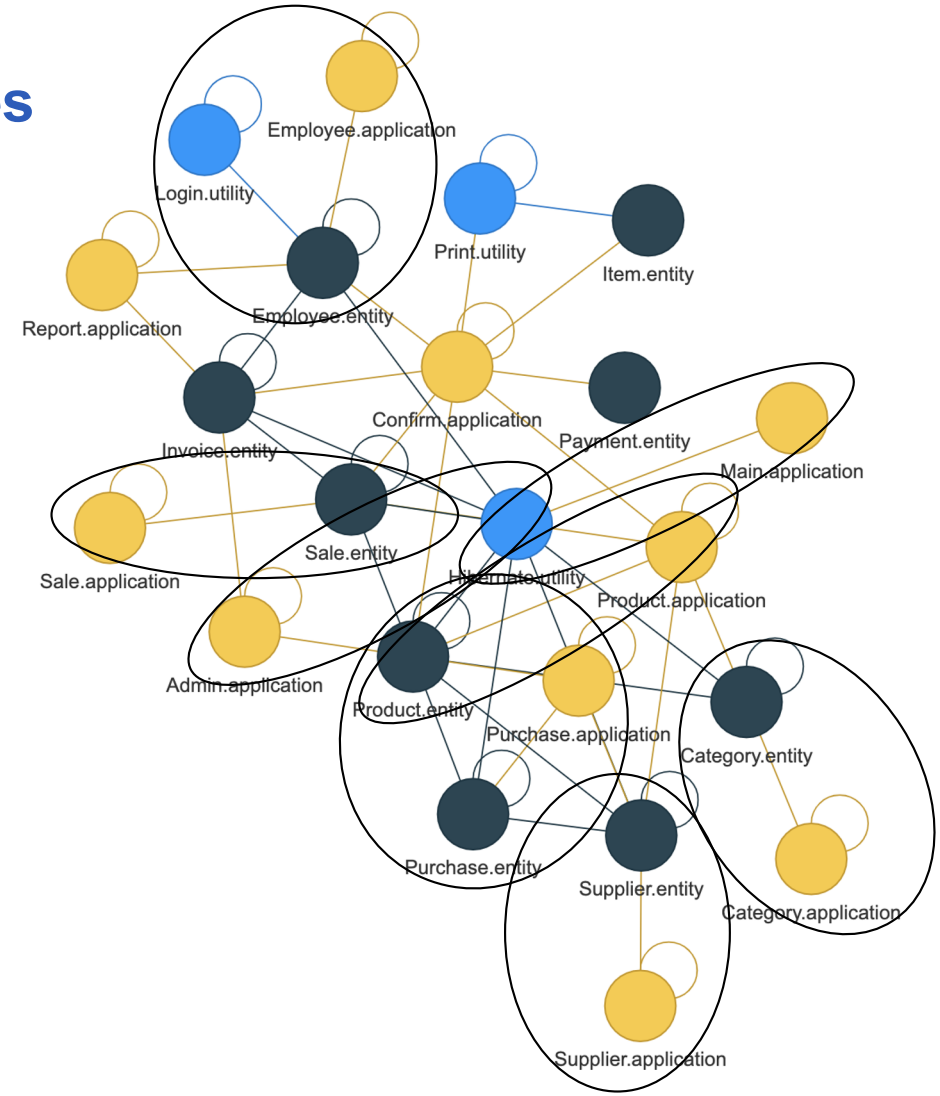


POS



Phase 3: **Mapping services to microservices**

- Utility classes
- Entity classes
- Application classes



POS

MicroMiner : Results

Legacy system	Precision	Recall	F-measure
Compiere	(67/89) 75.2%	(67/92) 72.8%	73.9%
POS	(8/11) 72%	(8/9) 88%	80%

Legacy system	Service type	Precision	Recall	F-measure
Compiere	Application	49.2%	81.3%	61.4%
	Entity	74.3%	90.7%	81.7%
	Utility	68.2%	83.9%	75.2%
	Total	68.2%	83.9%	75.2%
POS	Application	83.3%	100%	90.9%
	Entity	88.8%	88.8%	88.8%
	Utility	100%	33.3%	50%
	Total	86.3%	86.3%	86.3%

Conclusion

- A Brief History of Services
- SOA to Microservices, and beyond
- Current and Open Challenges
- Our contributions *“Migrating Legacy Systems to SOA”*
- Upcoming Faasification

Software Engineering for the service of Services

MERCI

Naouel Moha

École de Technologie Supérieure de Montréal

Naouel.Moha@etsmtl.ca

Séminaire du Master 2 informatique
de l'université de Montpellier,
December 10th, 2021

ÉTS
Le génie pour l'industrie