On the Migration from a Monolithic System to a Microservices Architecture: A Study of Automated Decomposition Approaches

Khaled Sellami

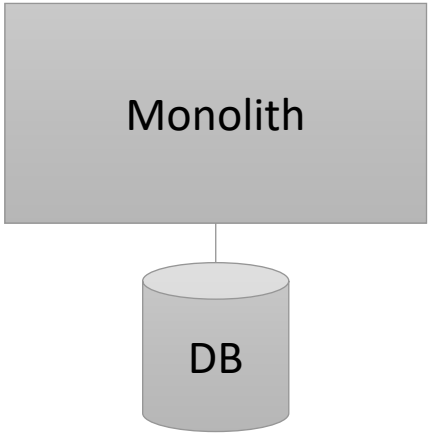
19 April 2024

Faculty of Science and Engineering
Department of Computer Science and Software Engineering

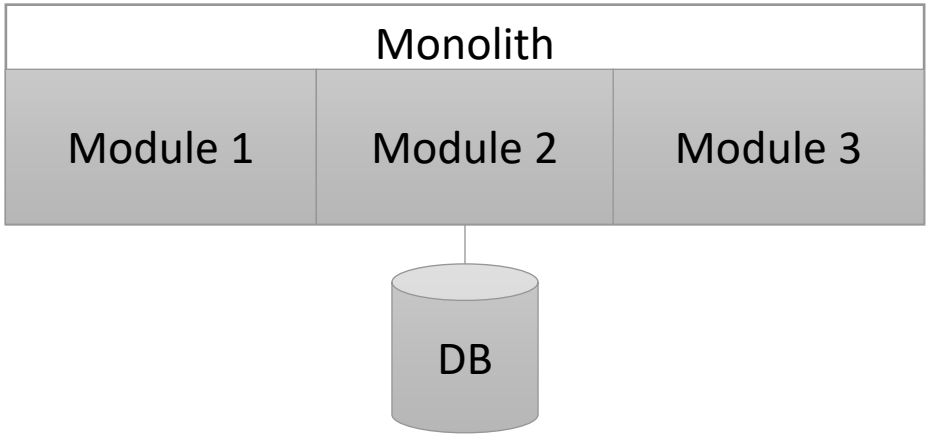
UNIVERSITÉ LAVAL

The Monolithic Architecture (1/2)

- **Definition:** A Monolith is a single unit of deployment.



Single Process Monolith



Modular Monolith

- Simple development process

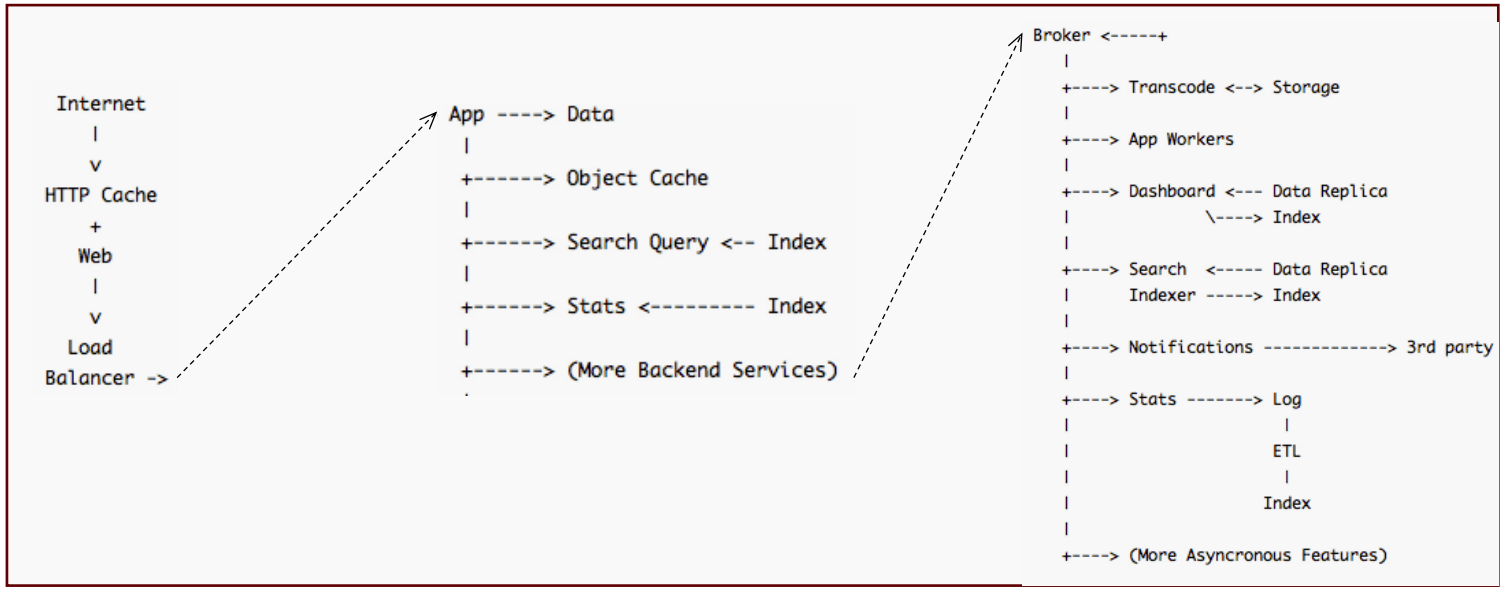
- Lack of scalability
- High coupling

The Monolithic Architecture (2/2)

Overview of SoundCloud's first architecture (2007) [1]:

```
Internet -> Web (Apache) -> App (Rails) -> Data (MySQL)
```

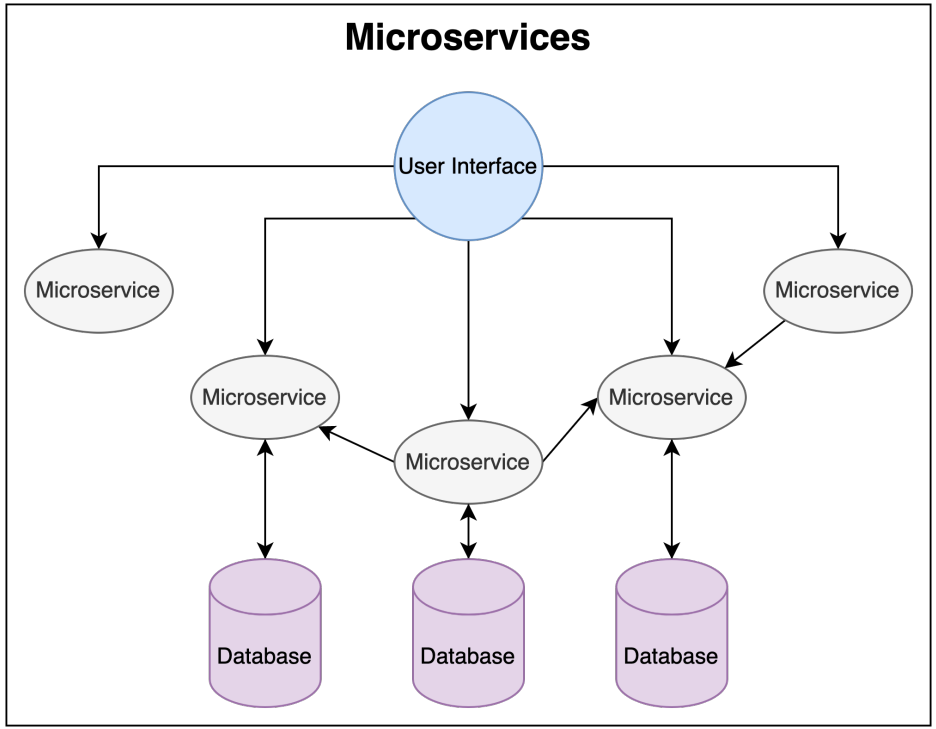
Overview of SoundCloud's last monolithic architecture (2012) [1]:



[1] Sean Treadway: https://developers.soundcloud.com/blog/evolution-of-soundclouds-architecture

The Microservices Architecture (1/2)

- Independent deployability
- Business Domain Driven
- Small microservices
- Modularity

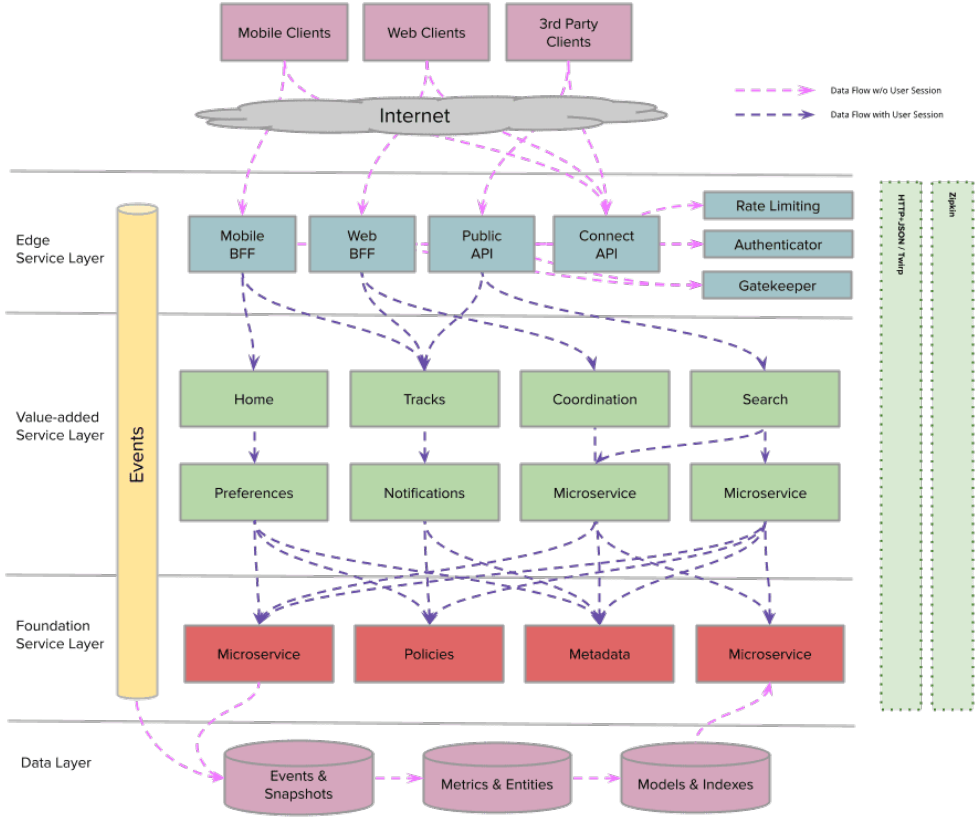


- Horizontal Scalability
- Robustness
- Technology diversity
- Clear isolation
- Convient for cloud and devops

- Implementation complexity
- Deployment overhead
- Adapting to new workflows

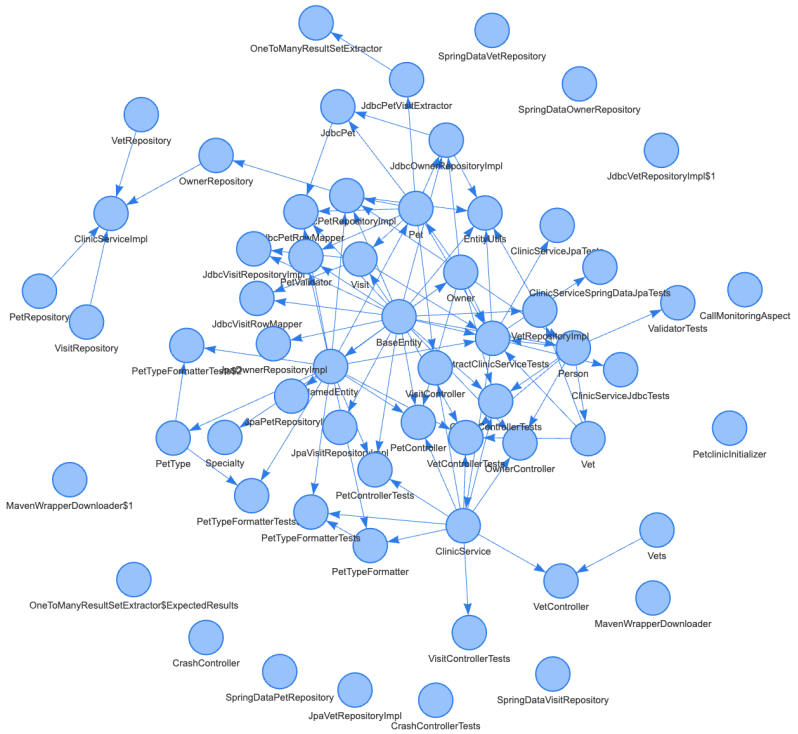
The Microservices Architecture (2/2)

Overview of SoundCloud's microservices' architecture (2024) [2]:

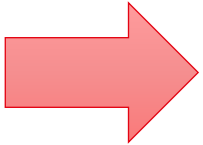


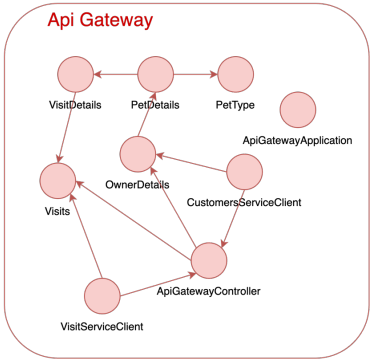
[2] Stephen Sun: https://www.fullstackexpress.io/p/evolution-soundcloud-architecture-final

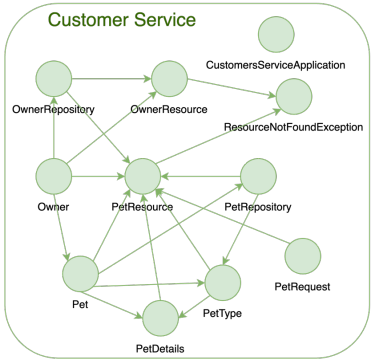
Migrating from the Monolith to Microservices: what is it?

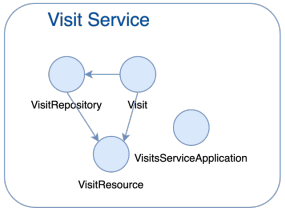


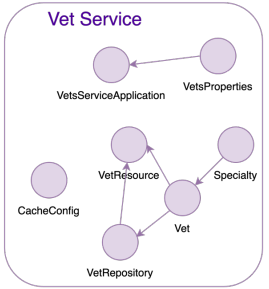
Monolith











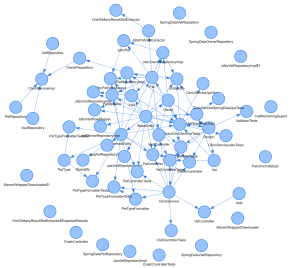
Microservices

Migrating from the Monolith to Microservices: Why?

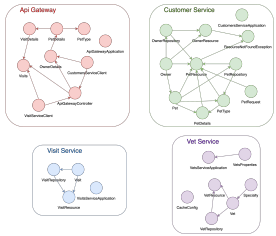
Why migrate from a monolith to microservices

- Scalability
- Development productivity
- Modernizing legacy application

Migrating from the Monolith to Microservices: How?

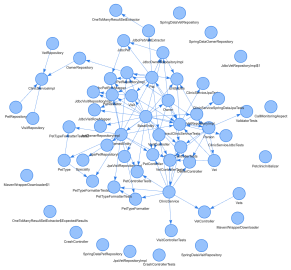


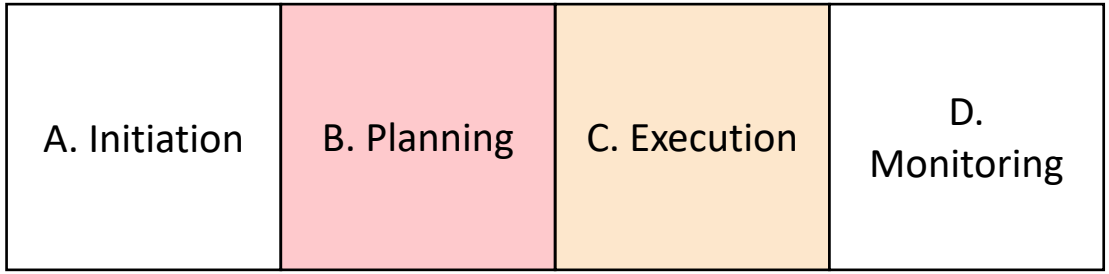
A. Initiation	B. Planning	C. Execution	D. Monitoring

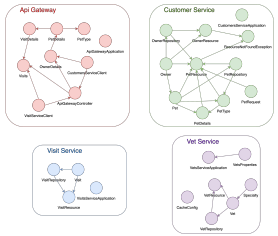


UNIVERSITÉ LAVAL

Migrating from the Monolith to Microservices: Challenges





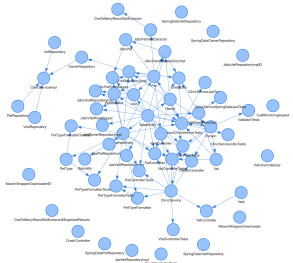


- Expensive
- Lengthy
- Lack of experience

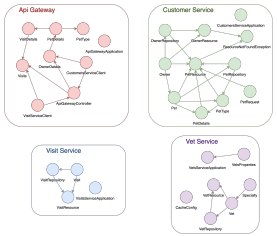
Migrating from the Monolith to Microservices: Who?

Decomposition approaches: Definition

A decomposition approach is a solution that partitions the components of a monolithic application (OOP classes, method, database tables, etc) into a set of potential microservices.



A. Initiation	B. Planning	C. Execution	D. Monitoring



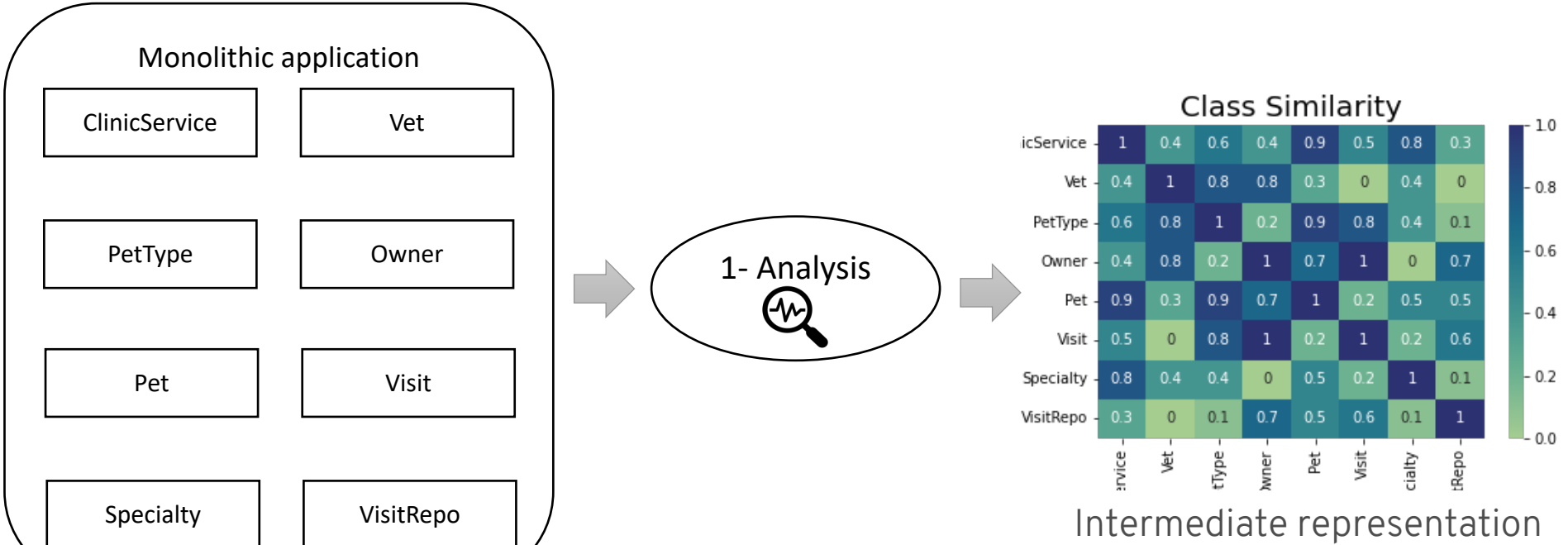
1- Analysis

2- decomposition

Decomposition approaches: Advantages

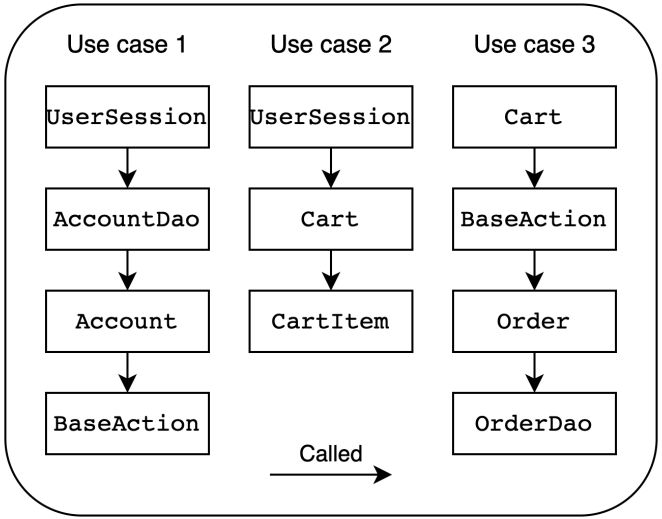
- Improve and evolve instead of refactoring
- Lower migration costs
- Ability to experiment before committing
- A starting point for traditional migration processes
- Unique perspective on the representation of the monolith

Decomposition approaches: Analysis (1/3)



Decomposition approaches: Analysis (2/3)

Execution traces + dynamic analysis



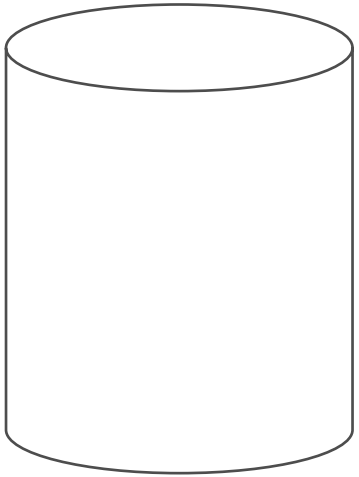
- Mono2micro [3]
- FoSCI [4]
- Process Mining Decomp [5]
- CoGCN [6] + Deeply [18]
- toMicroservices [17]

Source code + static analysis

```
21  public Cart() {
22    this.itemList.setPageSize(4);
23  }
24
25  public Iterator getAllCartItems() { return itemList.getSource().iterator(); }
26  public PagedListHolder getCartItemList() { return itemList; }
27  public int getNumberOfItems() { return itemList.getSource().size(); }
28
29  /* Public Methods */
30
31  public boolean containsItemId(String itemId) {
32    return itemMap.containsKey(itemId);
33  }
34
35  public void addItem(Item item, boolean isInStock) {
36    CartItem cartItem = (CartItem) itemMap.get(item.getItemId());
37    if (cartItem == null) {
38      cartItem = new CartItem();
39      cartItem.setItem(item);
40      cartItem.setQuantity(0);
41      cartItem.setInStock(isInStock);
42      itemMap.put(item.getItemId(), cartItem);
43      itemList.getSource().add(cartItem);
44    }
45    cartItem.incrementQuantity();
46  }
```

- SArF [8]
- Topic Modeling decomp [9]
- MVC decomp [10]

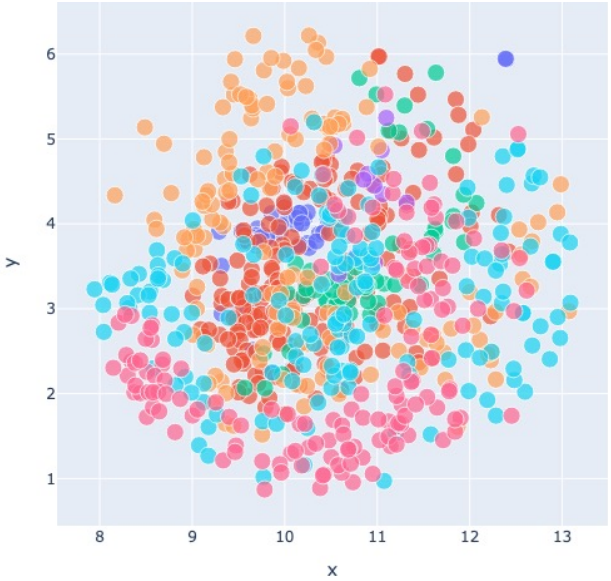
Database + source code



- CARGO [16]
- CHGNN [7]
- DataCentric [15]

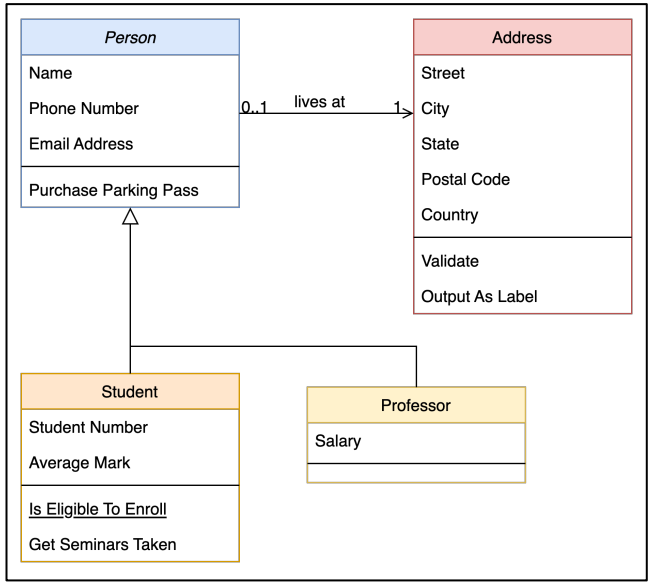
Decomposition approaches: Analysis (3/3)

Feature extraction



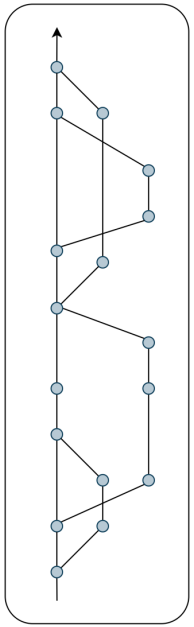
- Code2Vec decomposition [11]

Design artifacts



- Service Cutter [14]
- DataFlow Decomp [12]
- AKF decomp [19]

Commit history

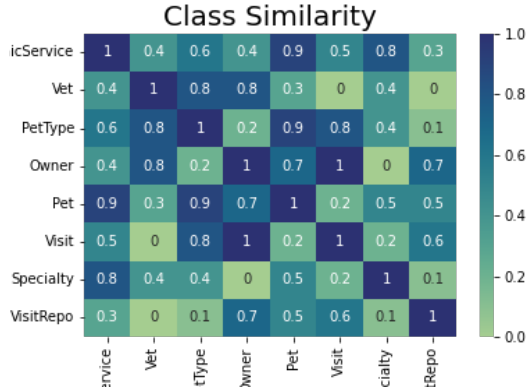


- MEM [13]

Decomposition approaches: Decomposition (1/2)

Monolithic application

- ClinicService
- Vet
- PetType
- Owner
- Pet
- Visit
- Specialty
- VisitRepo



2- decomposition

Visit service

- Visit
- VisitRepo

Vet service

- Vet
- Specialty

Owner service

- PetType
- Owner
- Pet

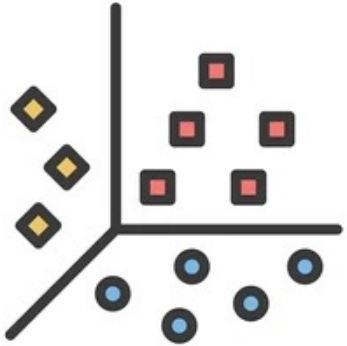
Gateway service

- ClinicService

Decomposition

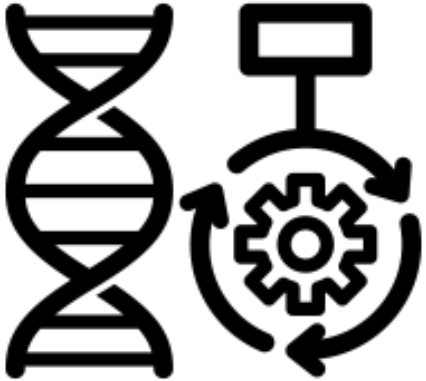
Decomposition approaches: Decomposition (2/2)

Clustering



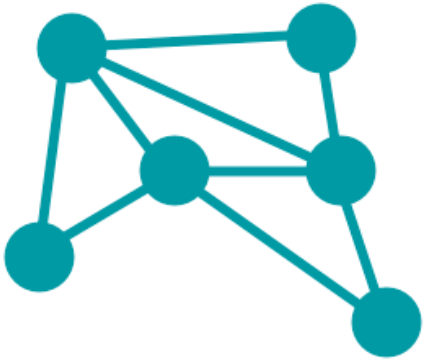
- MEM [13]
- Mono2micro [3]
- Service Cutter [14]
- Topic Modeling [9]
- SArF [8]
- MVC decomp [10]
- Code2Vec decomp [11]
- DataFlow decomp [12]
- FoSCI [4]

Genetic Algorithms



- FoSCI [4]

Graph Neural Networks



- CHGNN [7]
- CO-GCN [6] + Deeply [18]

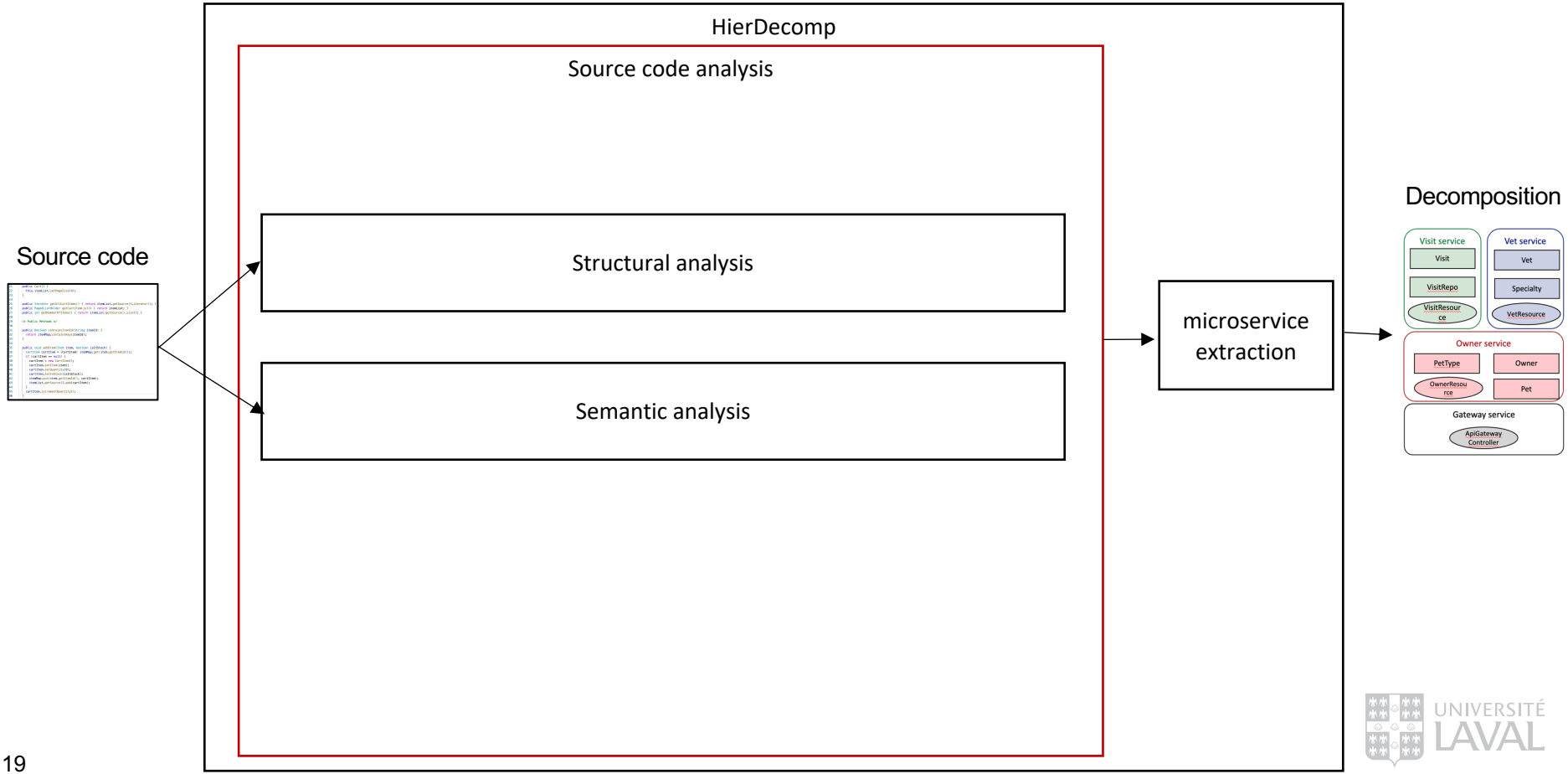
HierDecomp: Introduction

A Hierarchical DBSCAN Method for Extracting Microservices from Monolithic Applications
The International Conference on Evaluation and Assessment in Software Engineering 2022 (EASE2022)

Main contributions:

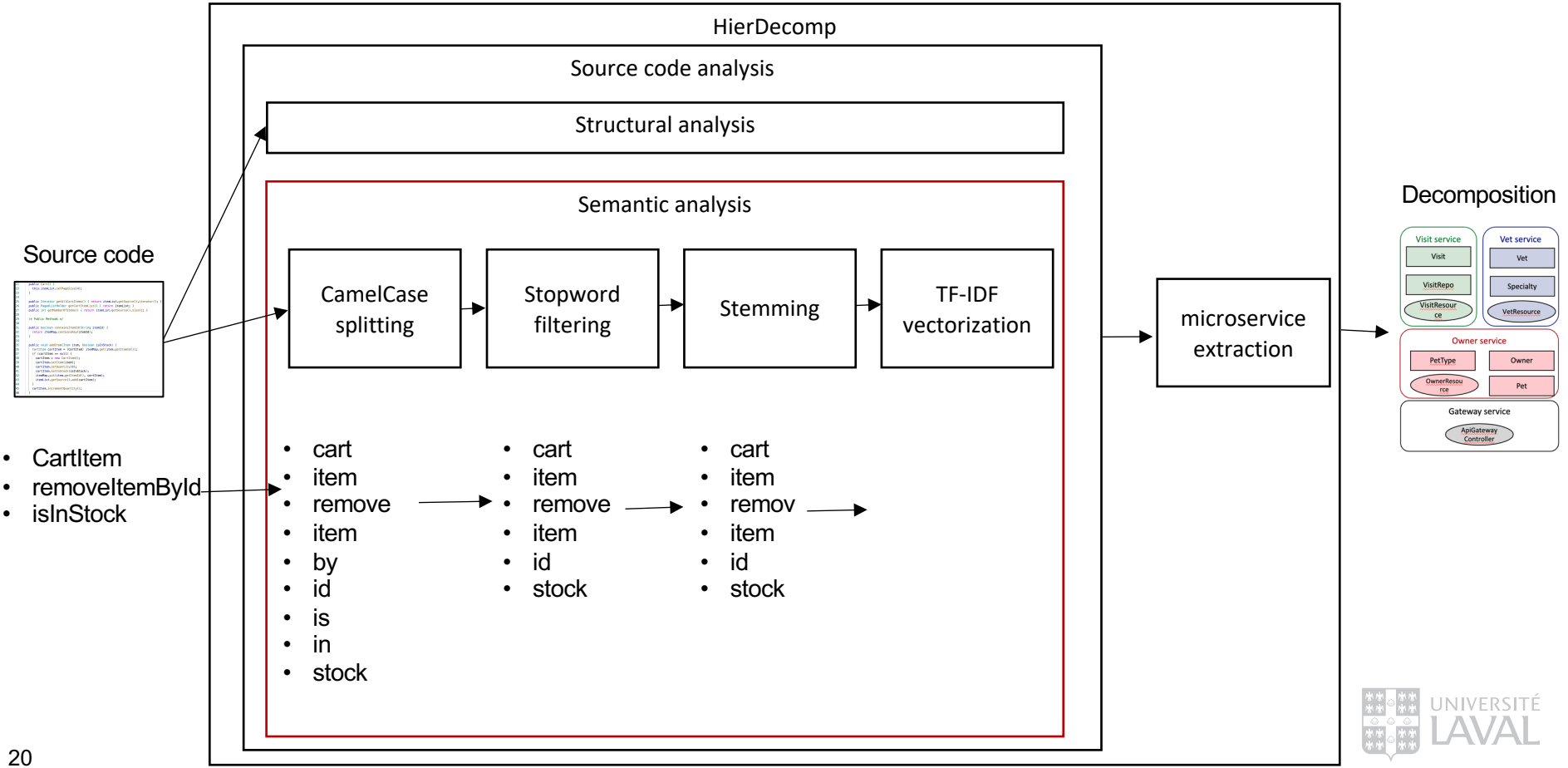
- A hierarchical decomposition suggestion for result explainability and user choice flexibility.
- Number of target microservices is inferred.
- Introduce a new evaluation approach for microservices decomposition.

HierDecomp: Analysis



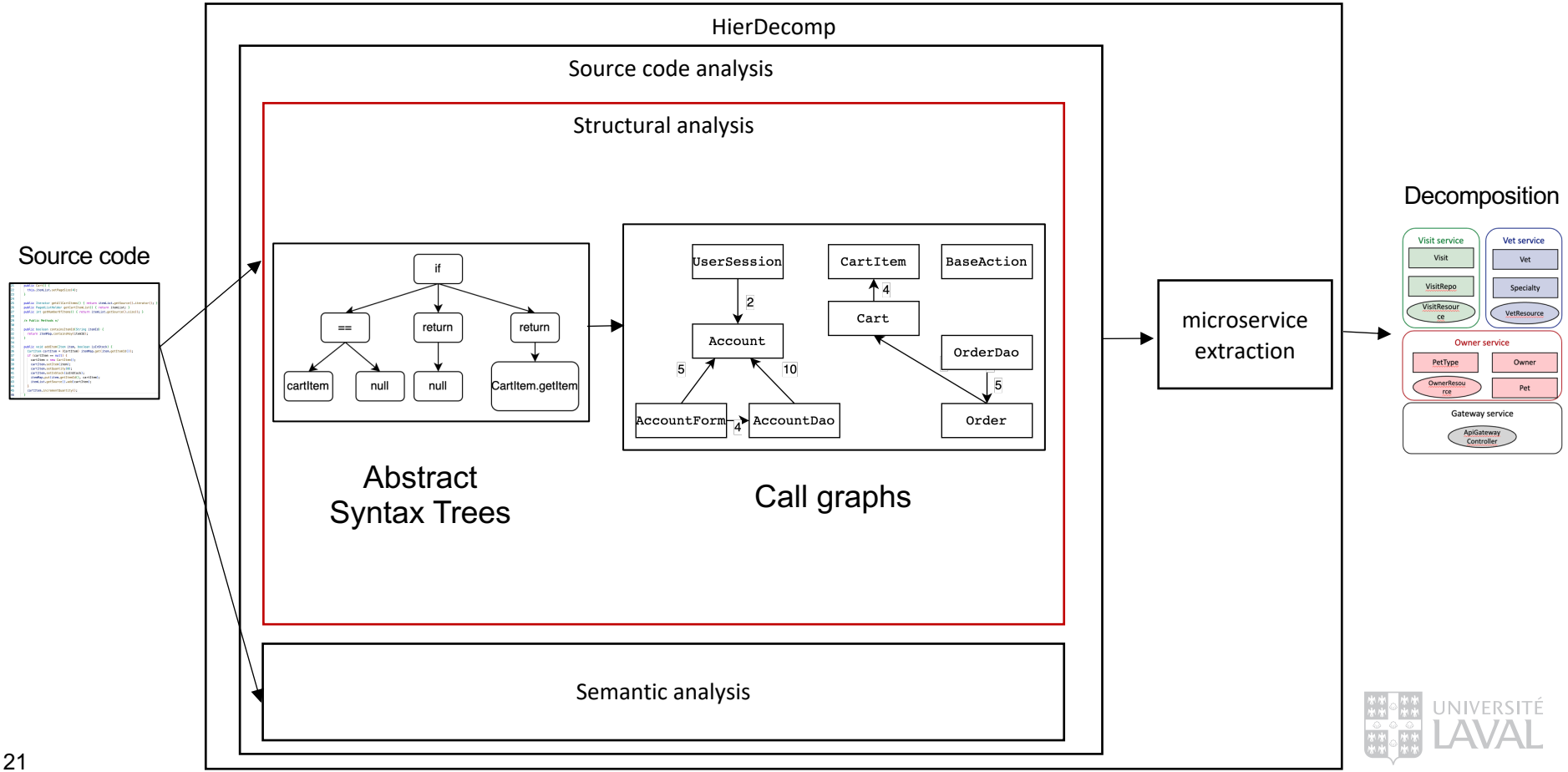
UNIVERSITÉ LAVAL

HierDecomp: Semantic Analysis

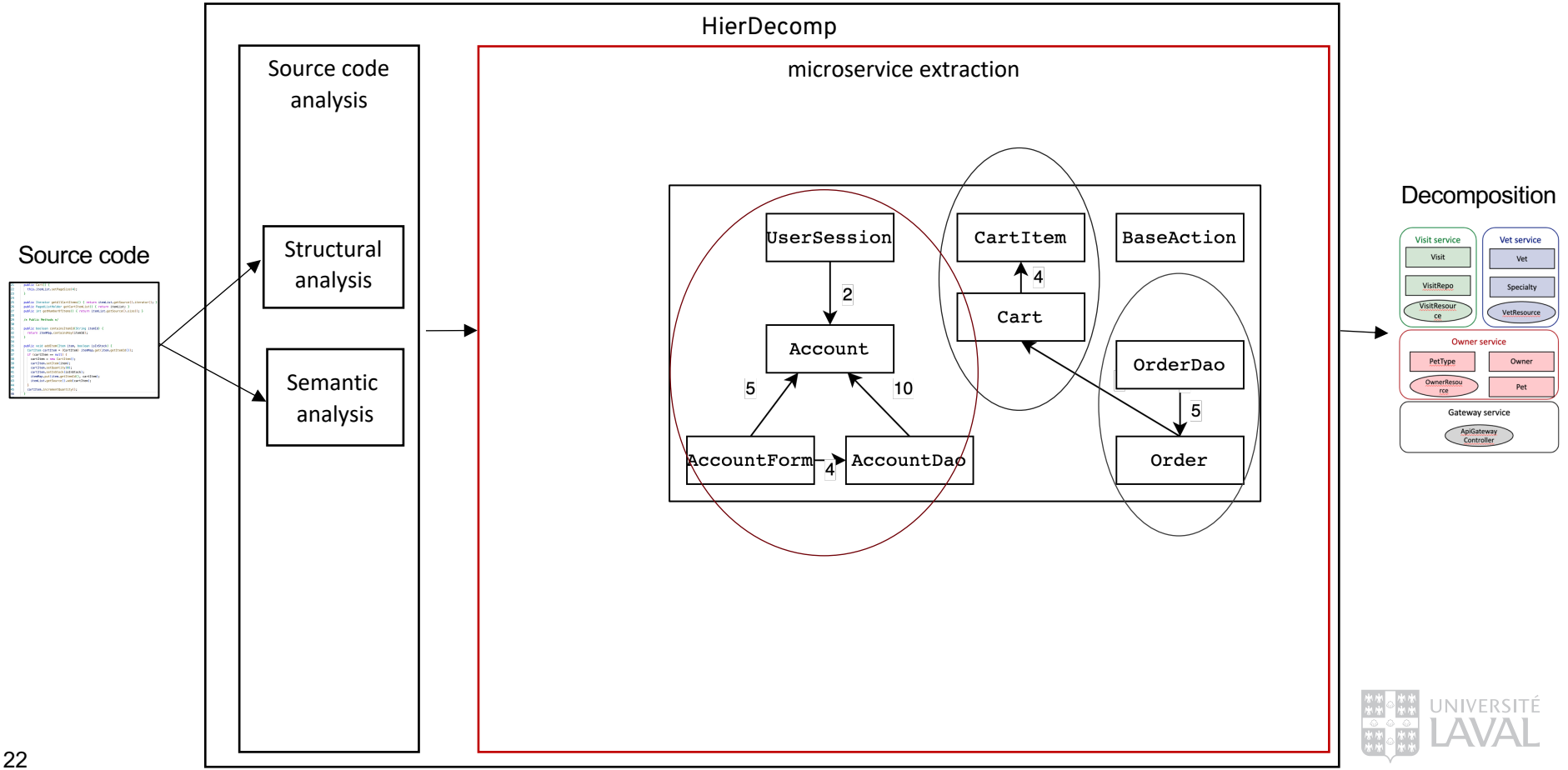


UNIVERSITÉ LAVAL

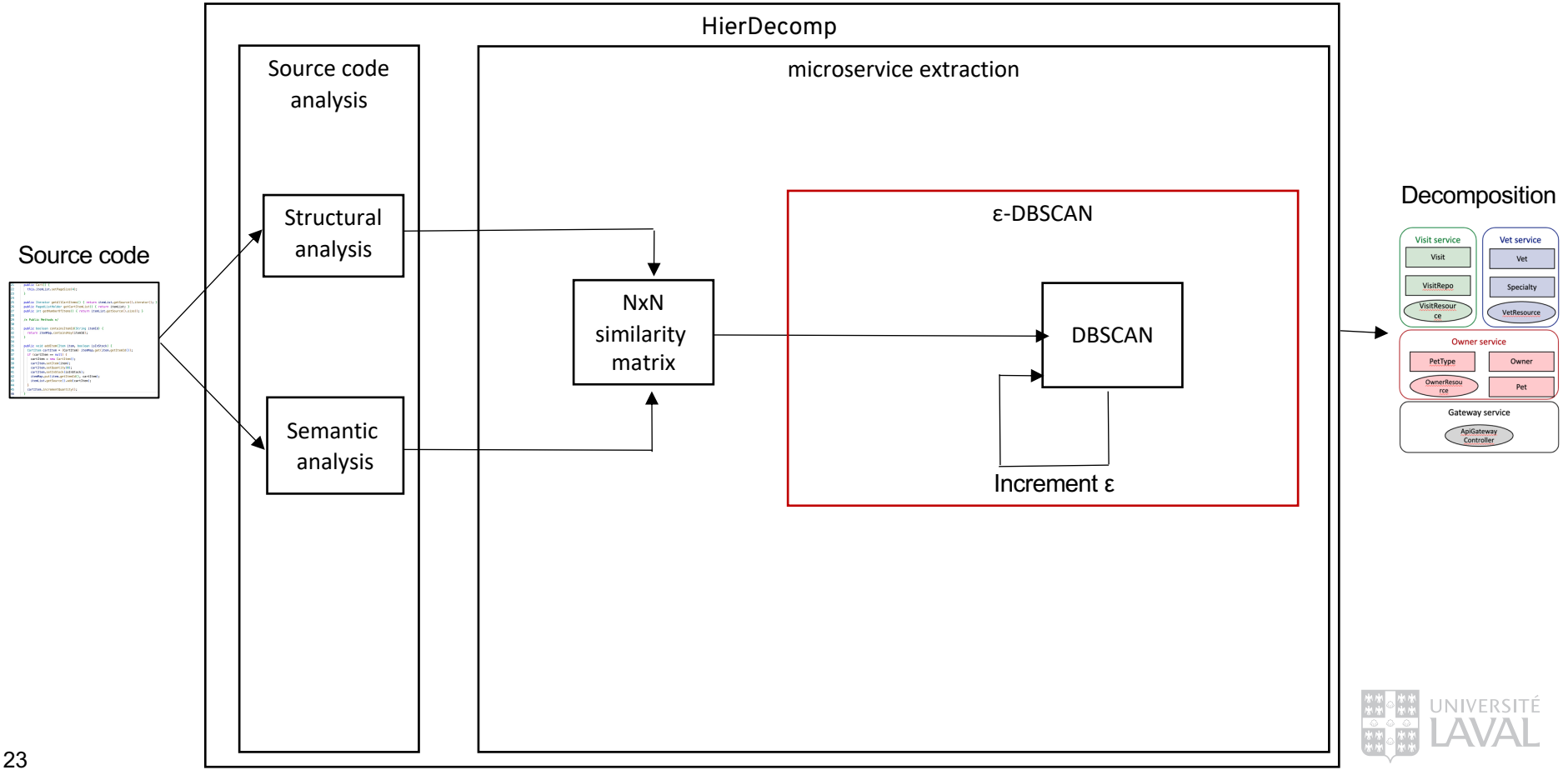
HierDecomp: Structural Analysis



HierDecomp: epsilon-DBSCAN (1/2)



HierDecomp : epsilon-DBSCAN (1/2)



UNIVERSITÉ LAVAL

HyDec: introduction

Combining Static and Dynamic Analysis to Decompose Monolithic Application into Microservices

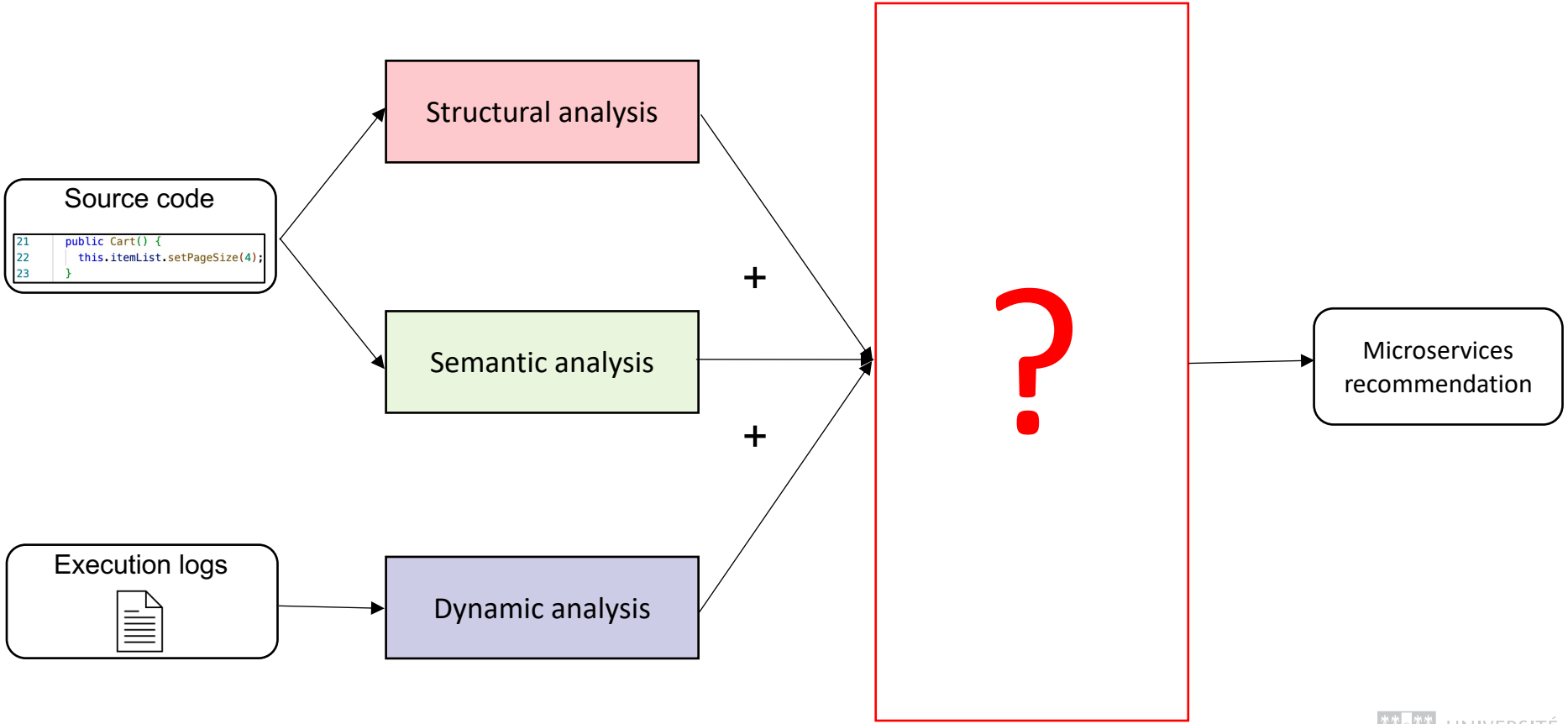
The 20th International Conference on Service-Oriented Computing 2022 (ICSOC2022)

Main contributions:

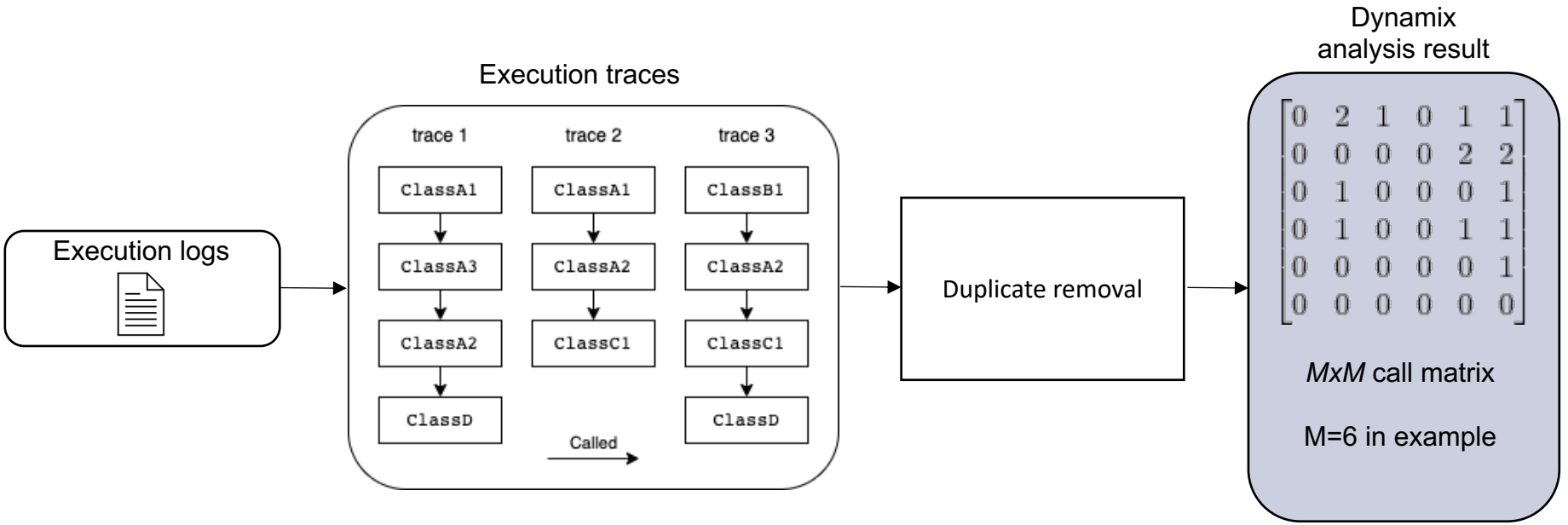
- A general approach to combine multiple analysis sources in order to generate hierarchical decompositions.
- Multiple combination approaches.
- A decomposition approach that improves the coverage while maintaining a performance similar to state-of-the-art approaches.

UNIVERSITÉ LAVAL

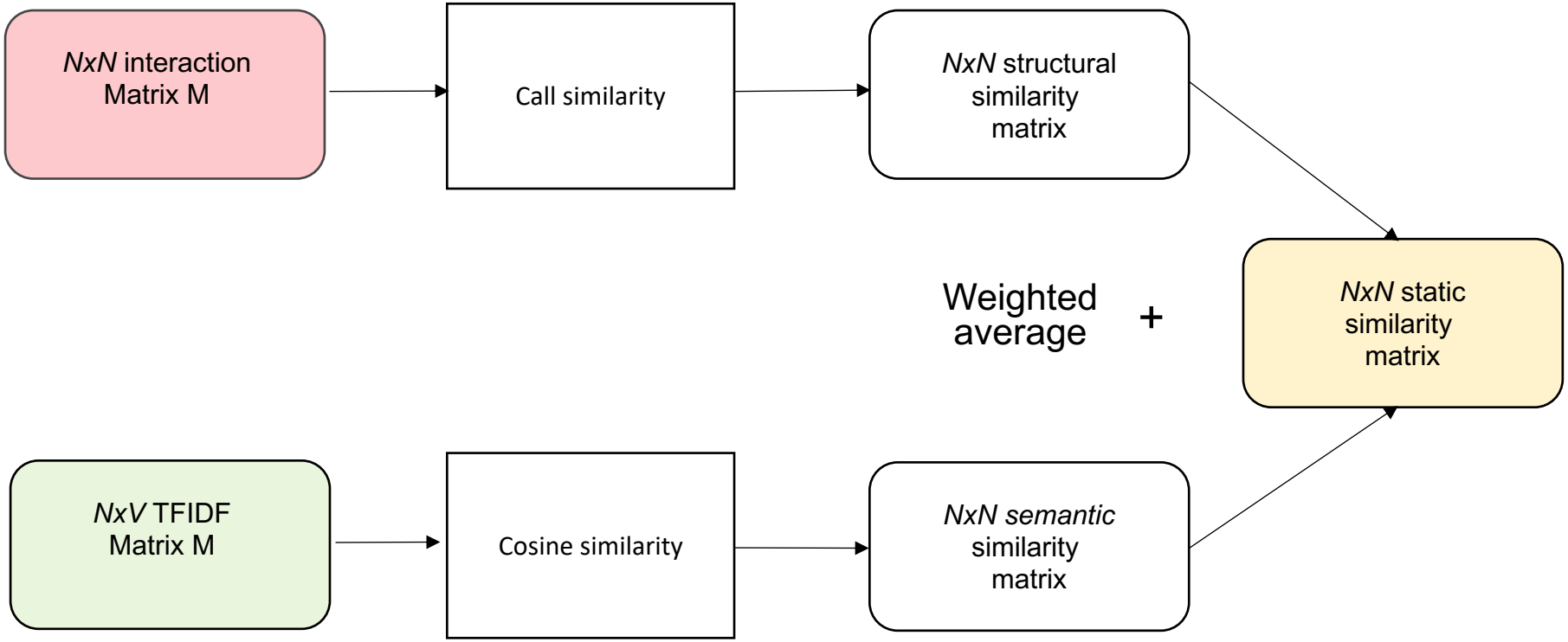
HyDec: Overview



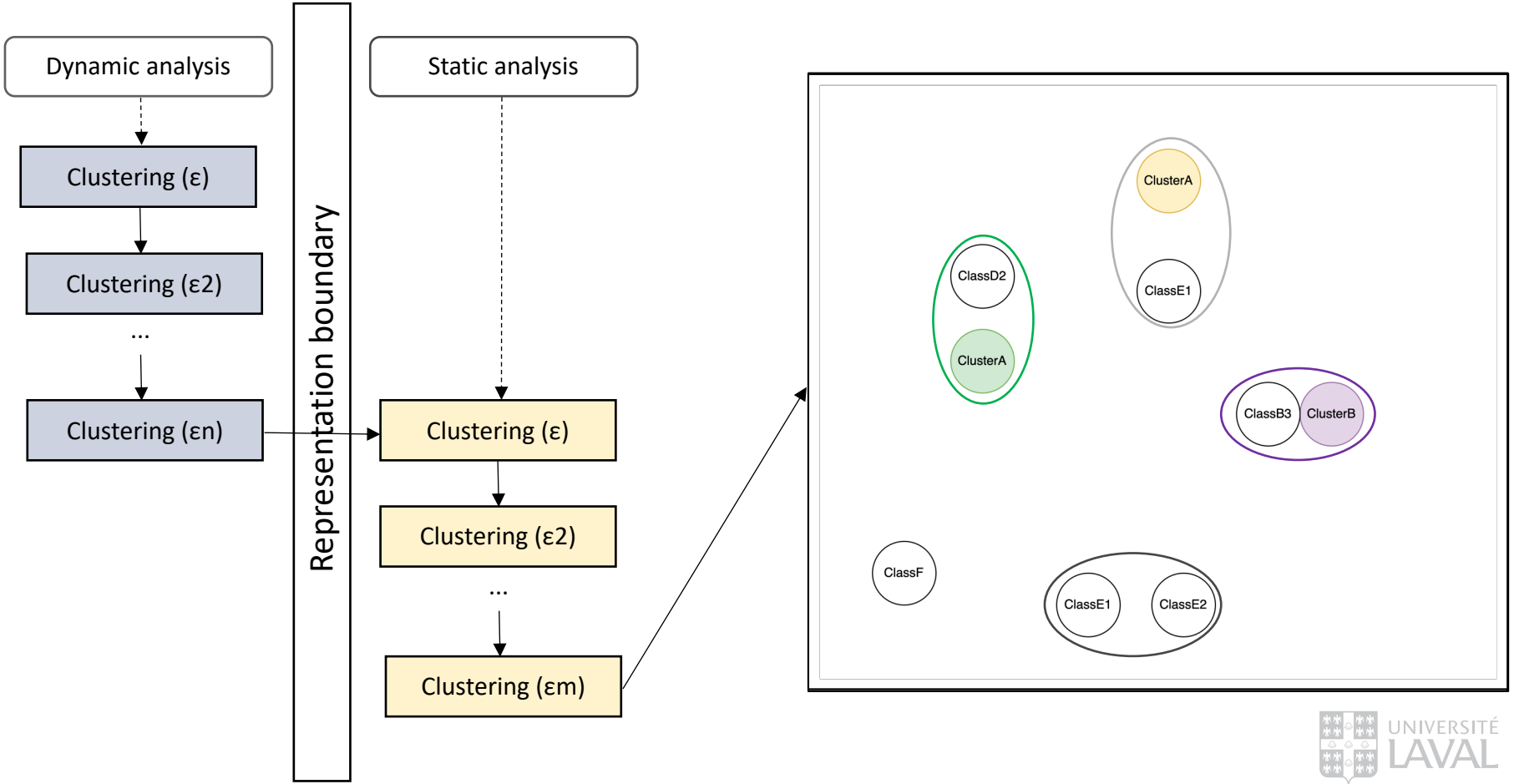
HyDec: Dynamic analysis



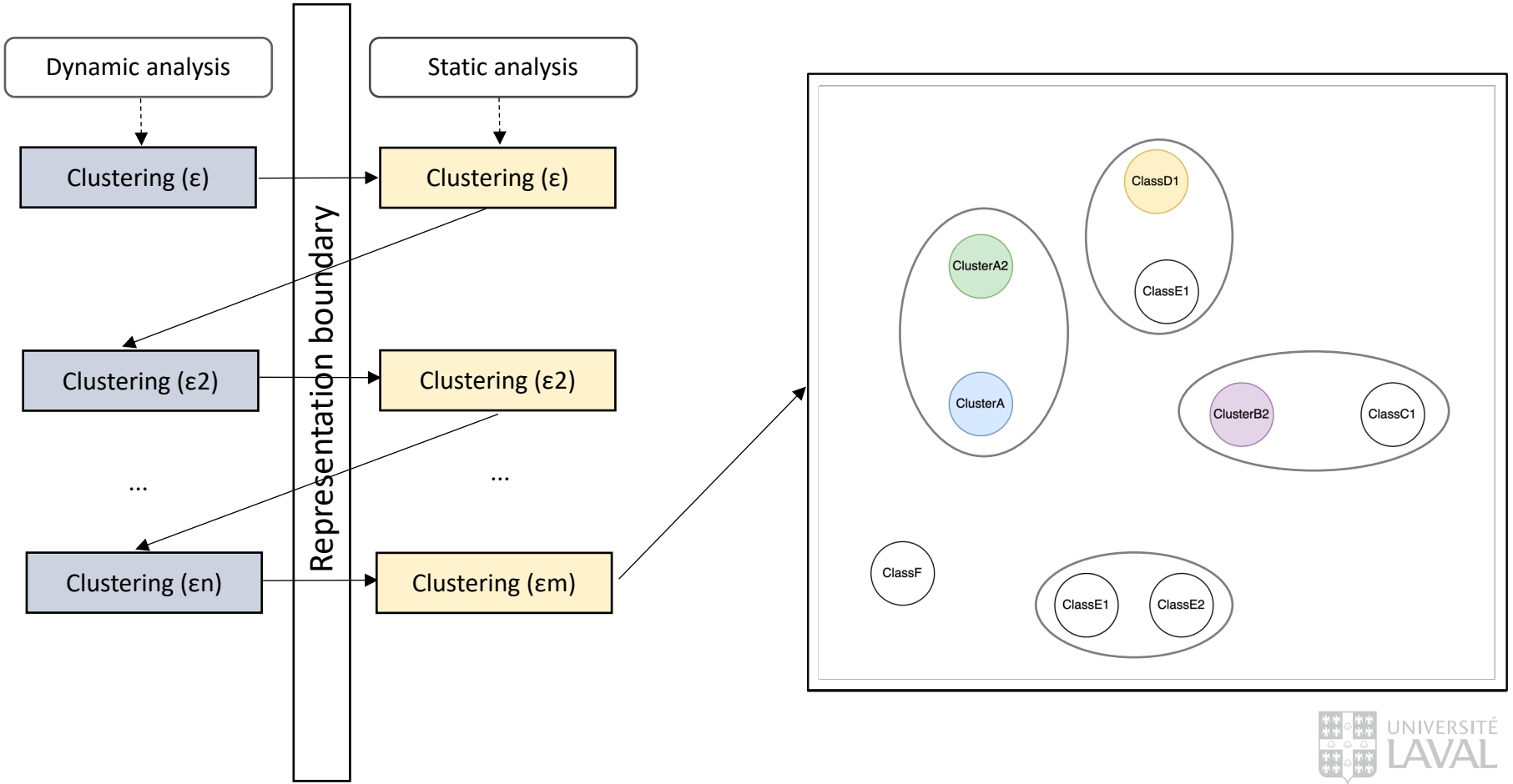
HyDec: Static similarity



HyDec: Sequential epsilon-DBSCAN



HyDec: Alternating epsilon-DBSCAN



MSExtractor: Introduction

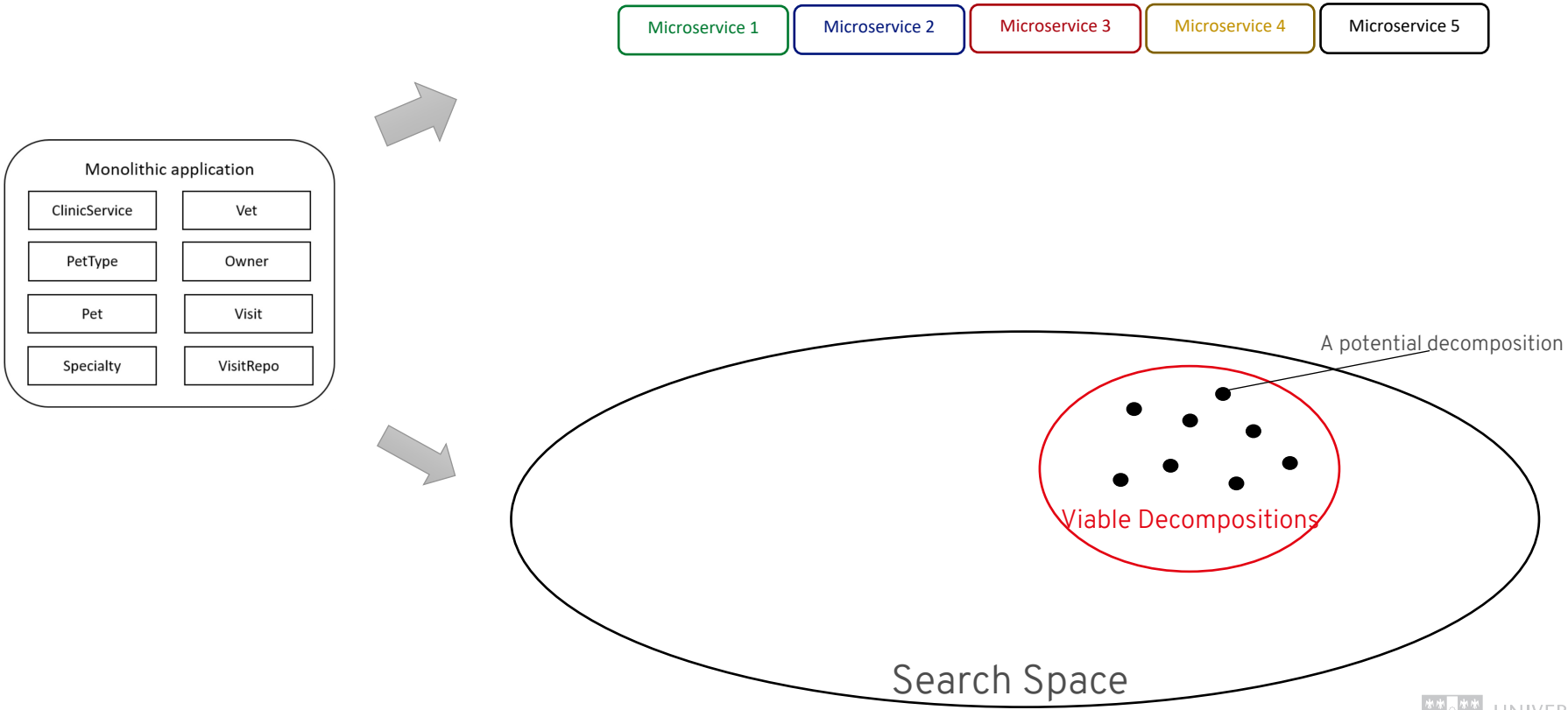
Improving microservices extraction using evolutionary search
The Journal of Information and Software Technology Volume 151

Main contributions:

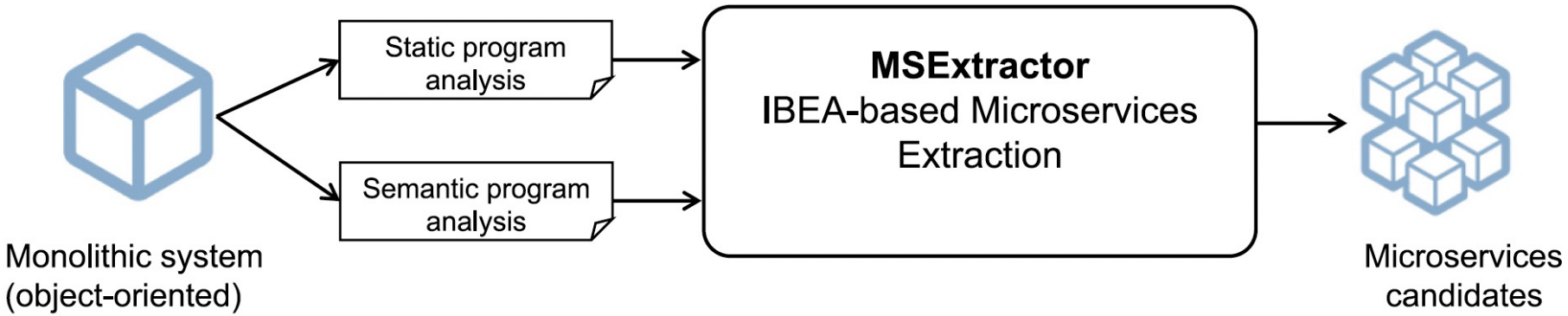
- Formulating the microservices decomposition task as a search problem with an evolutionary algorithm.
- Using a multi-objective evolutionary algorithm in order to encapsulate the different aspects within a decomposition.
- Differentiating between interface and inner classes within a decomposition.

UNIVERSITÉ LAVAL

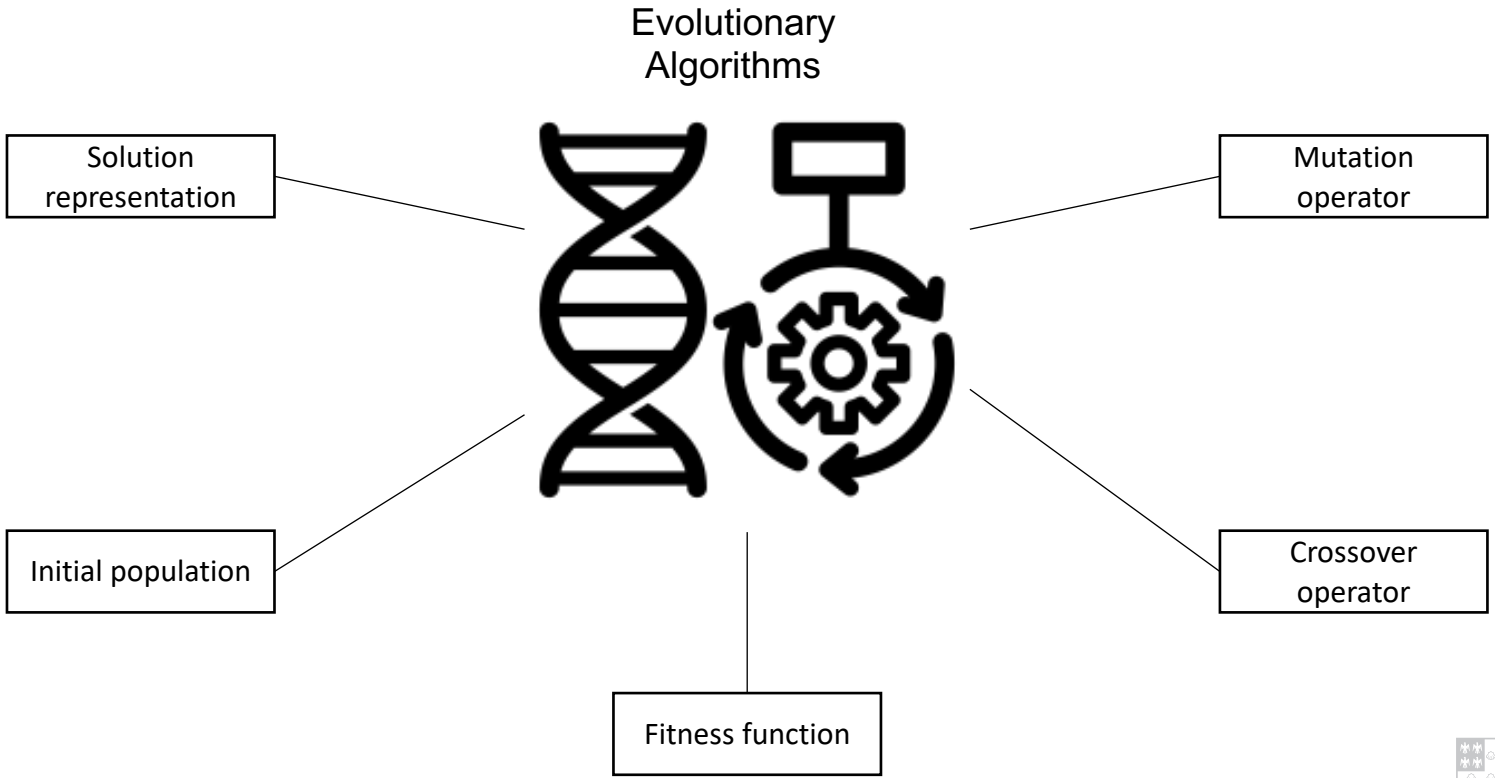
MSExtractor: Clustering vs Optimization



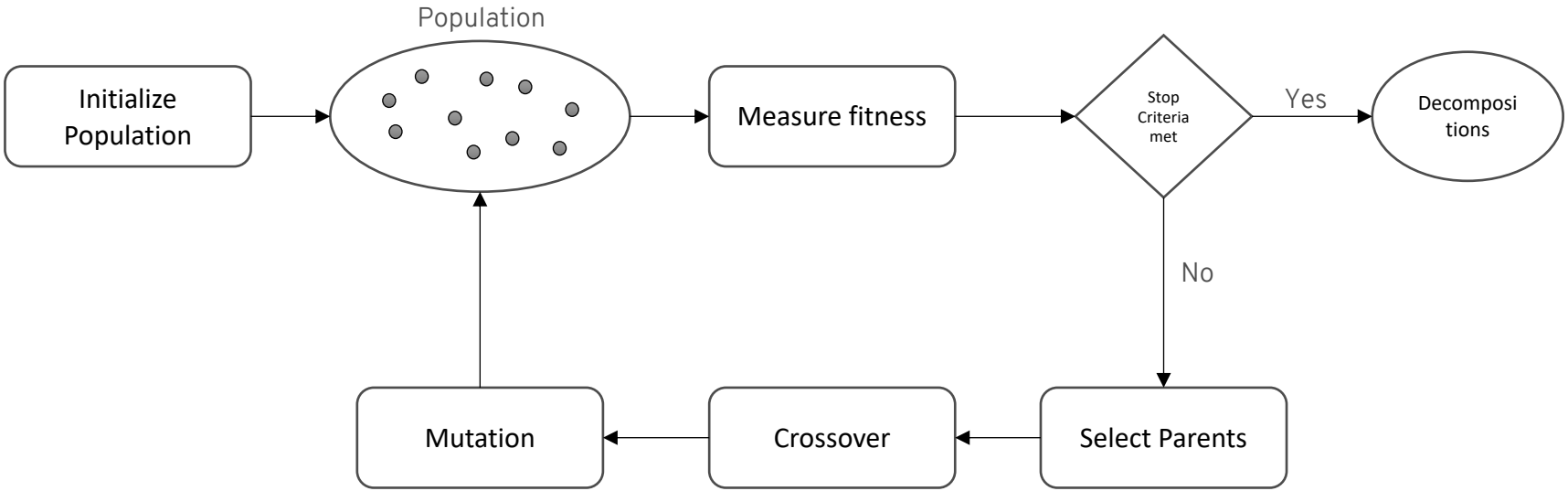
MSExtractor: Overview



MSExtractor: Evolutionary algorithms

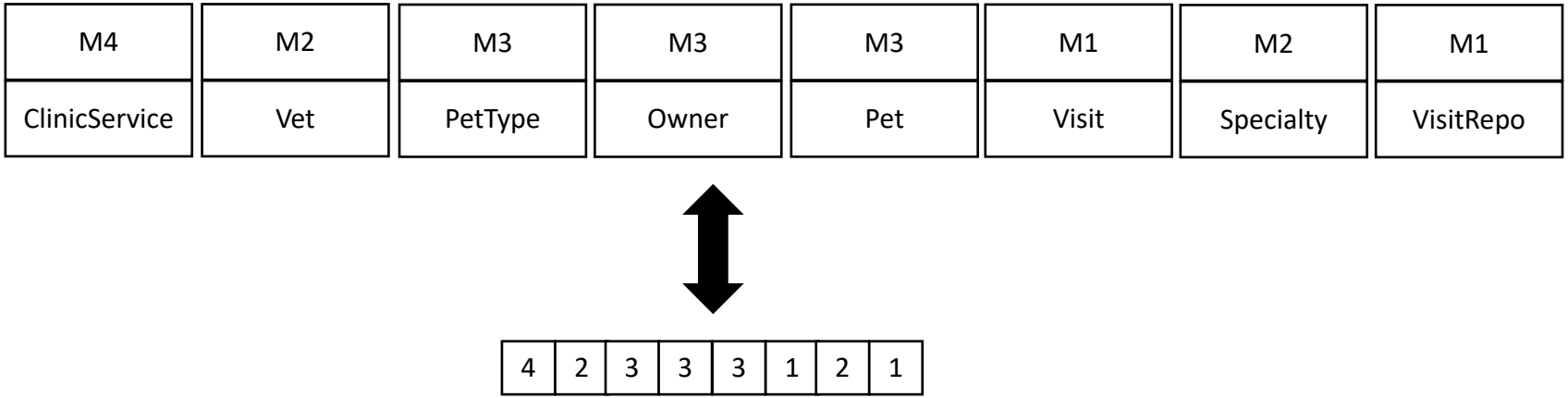


MSExtractor: Workflow



MSExtractor: Solution representation

label-based integer encoding:

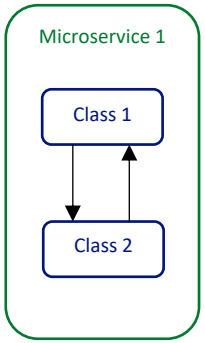


Initial population: random sampling

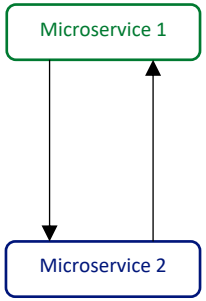
MSExtractor: MOEA

Multi-Objective Evolutionary Algorithm (MOEA)

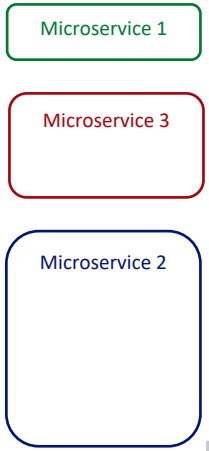
Cohesion



Coupling

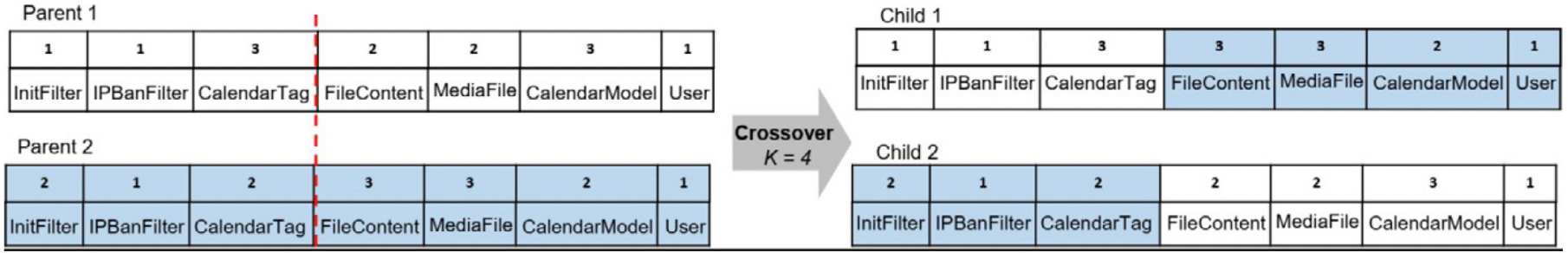


Granularity



MSExtractor: Operators

Crossover operator



Mutation operator

RLDec: Introduction

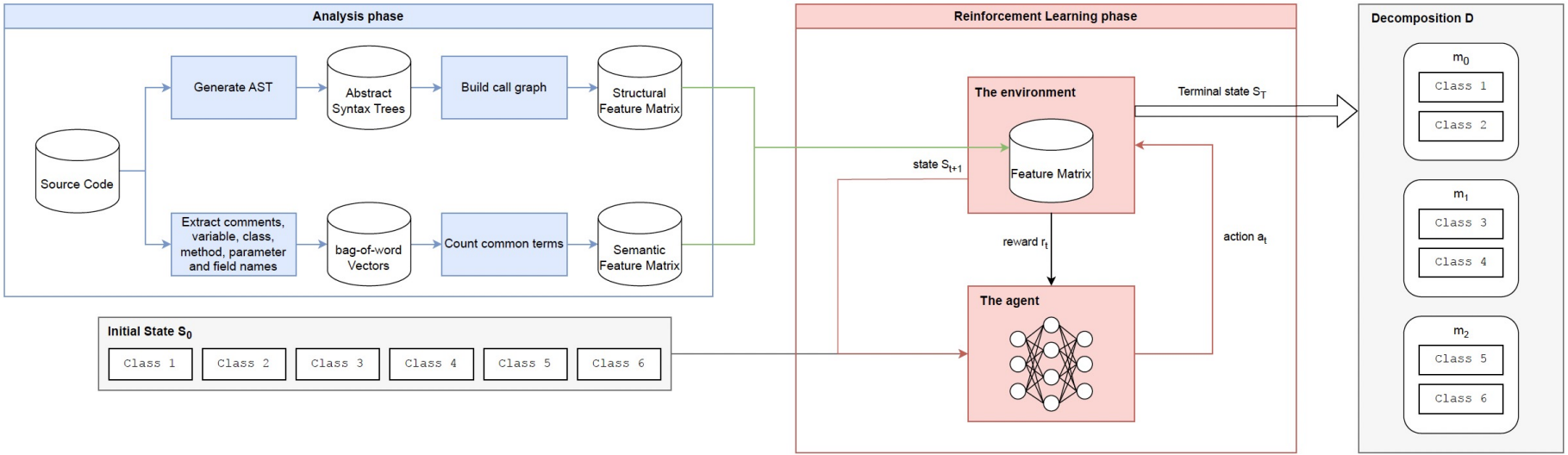
Extracting Microservices from Monolithic Systems using Deep Reinforcement Learning
In Review *for The Empirical Software Engineering*

Main contributions:

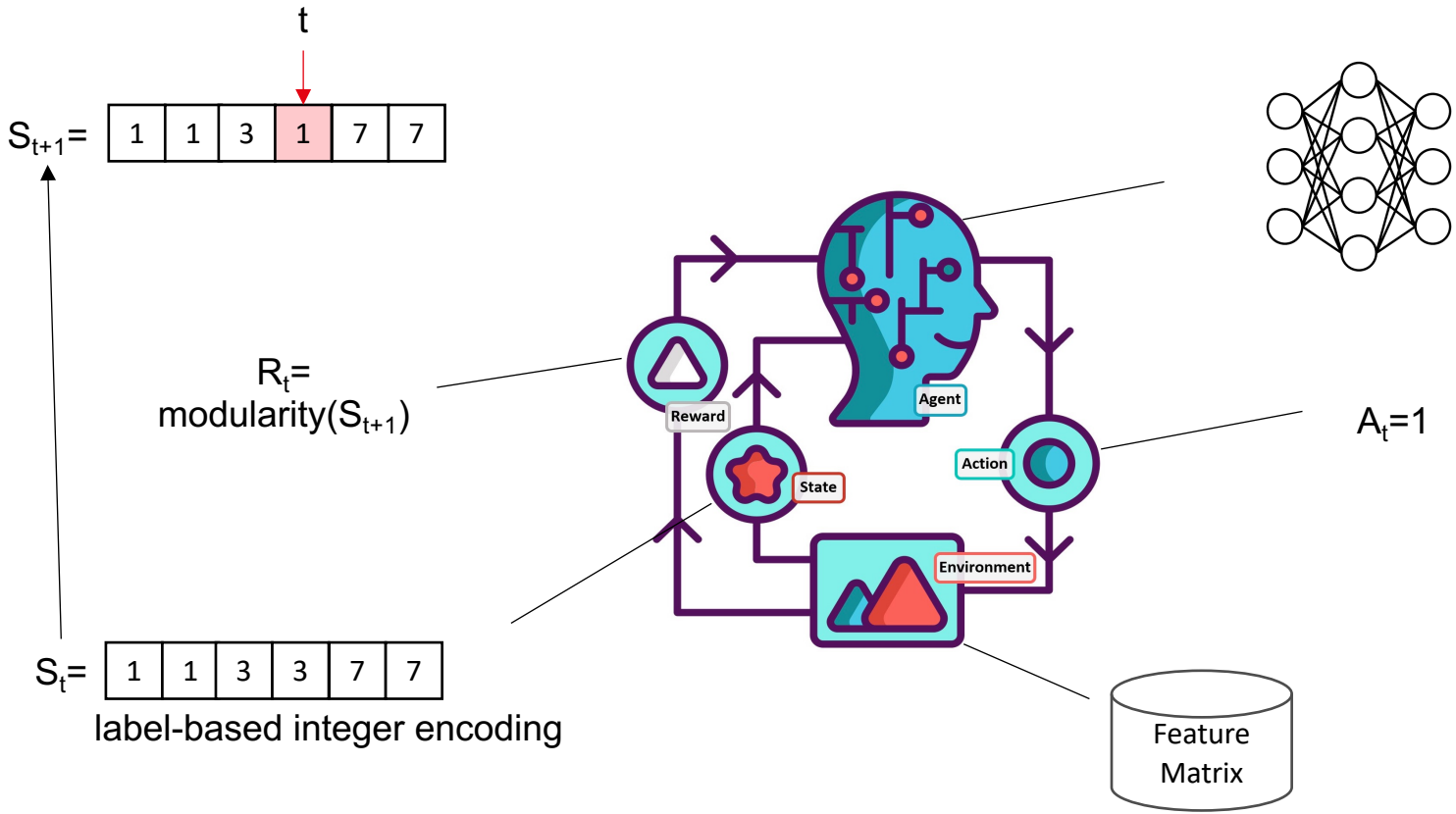
- Formulate the microservices decomposition problem as a reinforcement learning task.
- Improving the evaluation process by introducing novel metrics that can encapsulate multiple aspects and that can compare with existing decompositions.

UNIVERSITÉ LAVAL

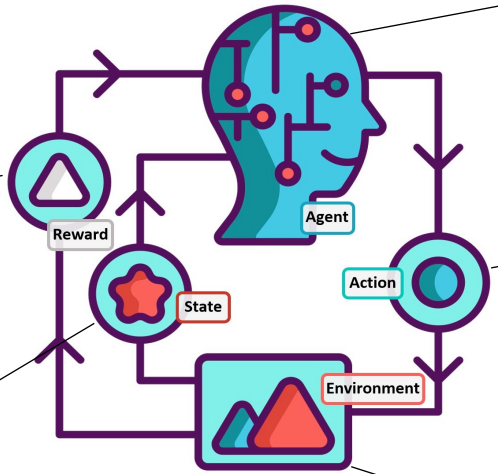
RLDec: Overview

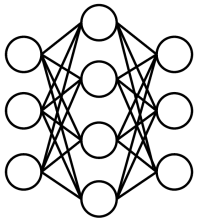


RLDec: The sequential approach



RLDec: The combined sequential approach





R_t= (SMQ(S_{t+1})+ CMQ(S_{t+1}))/2

A_t=1

S_t=

1	1	3	3	7	7

label-based integer encoding

Semantic Feature Matrix

Structural Feature Matrix

UNIVERSITÉ LAVAL

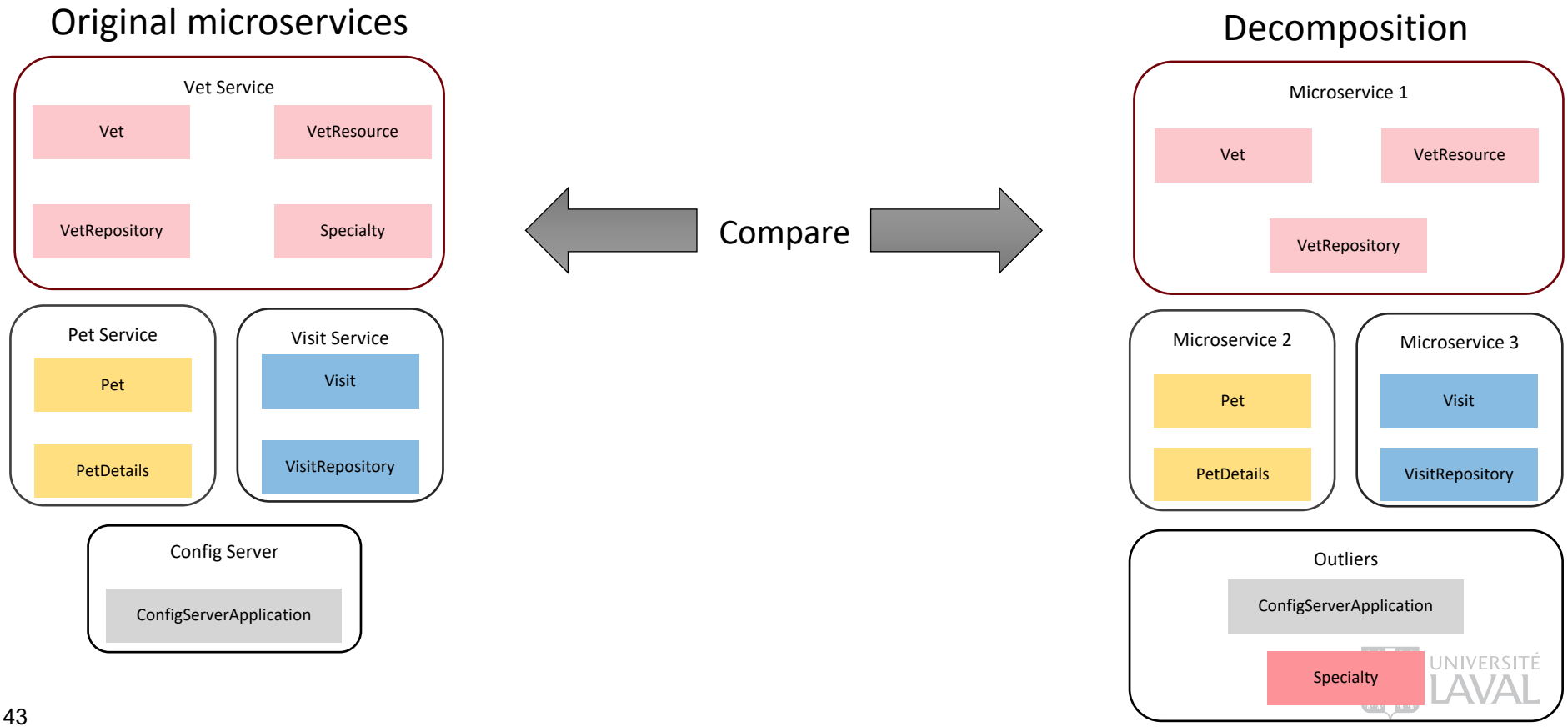
RLDec: RQ2

How does our approach perform when compared with state-of-the-art decomposition baselines?

baseline	CHM	CHD	ICP	BCP	NED	DSCORE
CoGCN	*33*	*30*	31	16*	31	34
HyDec	**15**	27*	21*	**15**	47	22*
Mono2micro	34	35	*25*	*26*	21*	*32*
MSExtractor	34	*30*	40	36	29	37
RLDec	16*	**10**	**20**	44	*24*	**9**
TopicDecomp	36	35	30	31	**15**	34

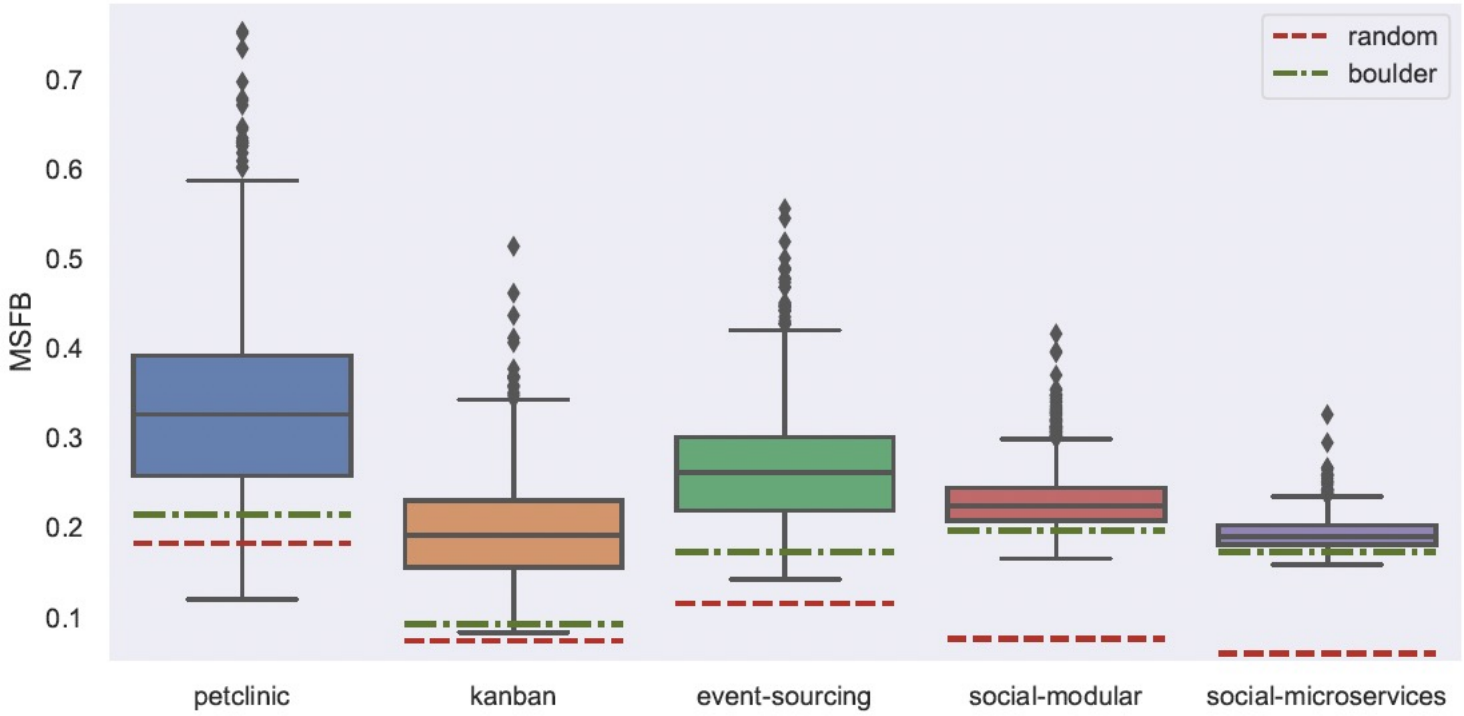
RLDec: RQ3 (1/2)

Is our approach able to recapture the components of microservices that were created by human experts?

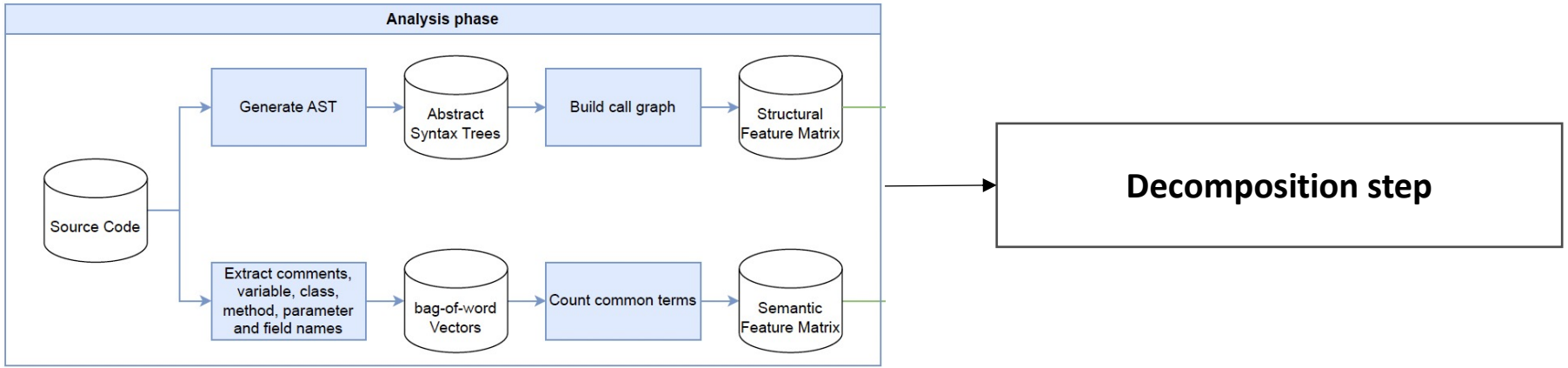


RLDec: RQ3 (2/2)

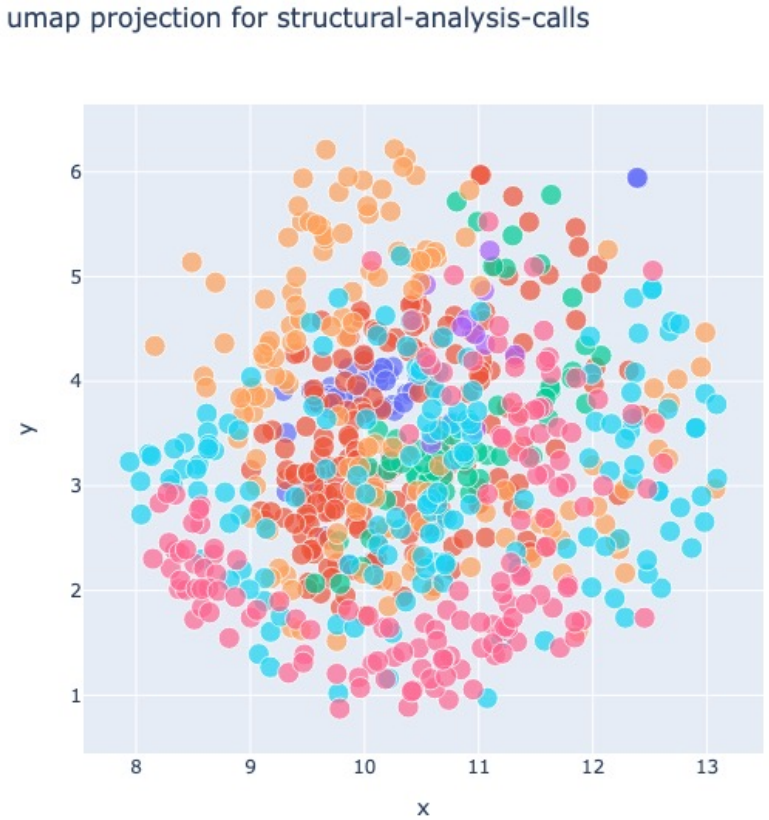
Is our approach able to recapture the components of microservices that were created by human experts?

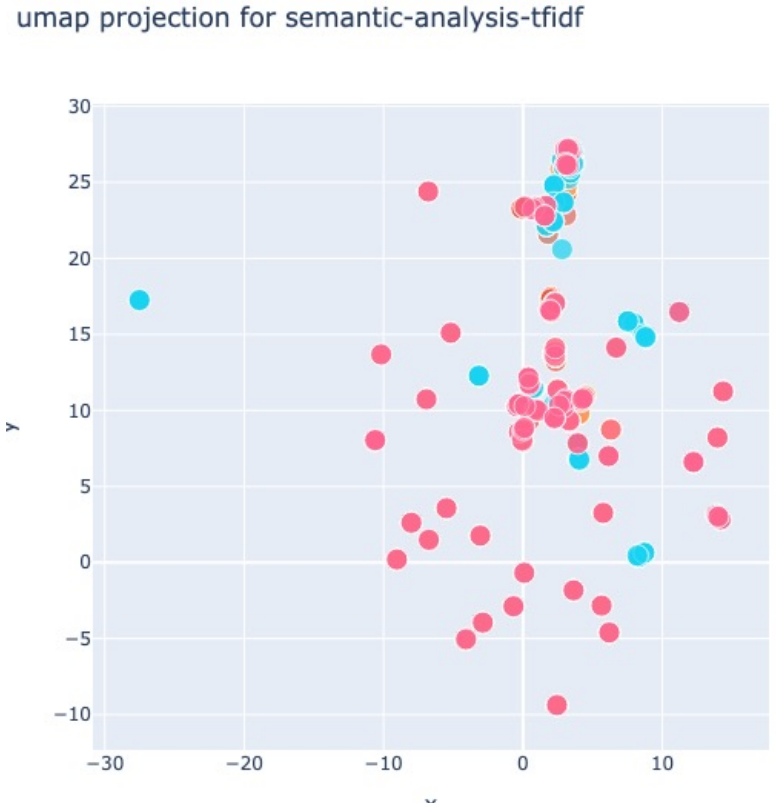


Representation learning: Motivation



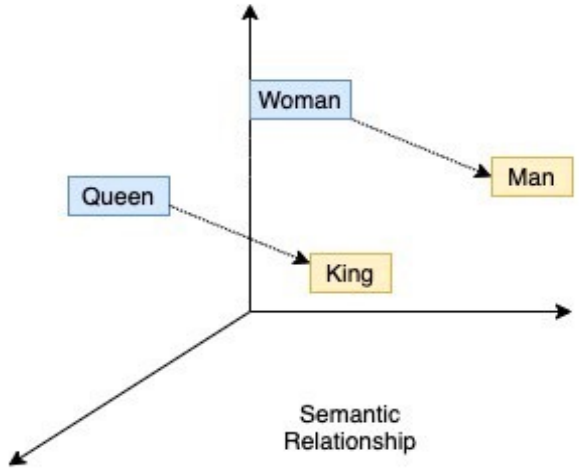
Representation learning: Visualization

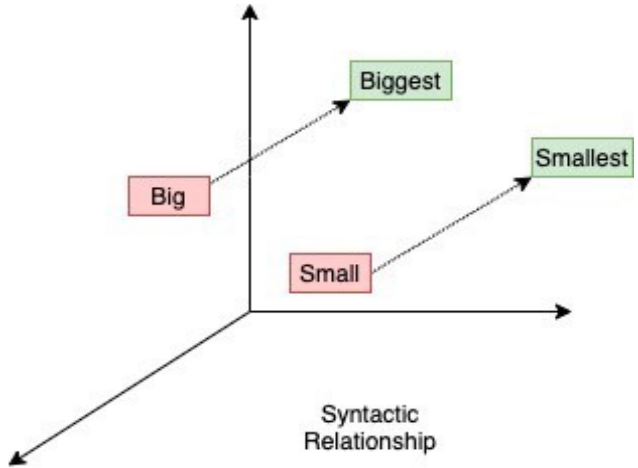




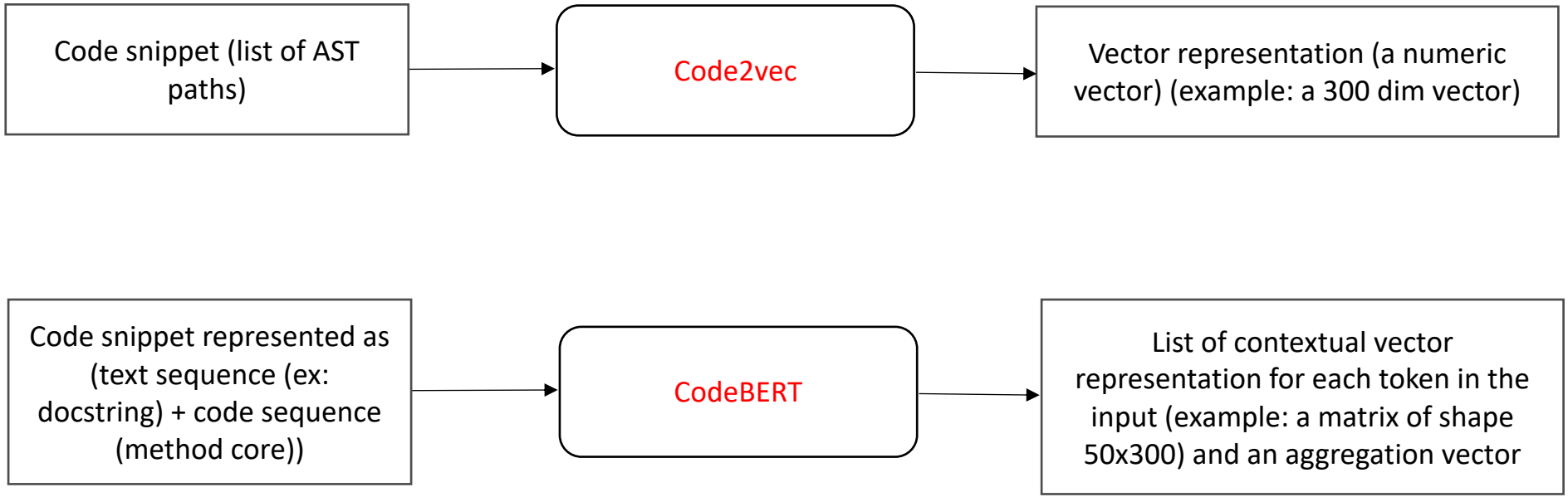
Representation learning: alternatives

word (set of characters) (ex: example) → Word Embedding model (ex: word2vec) → Vector representation (a numeric vector) (example: a 300 dim vector)



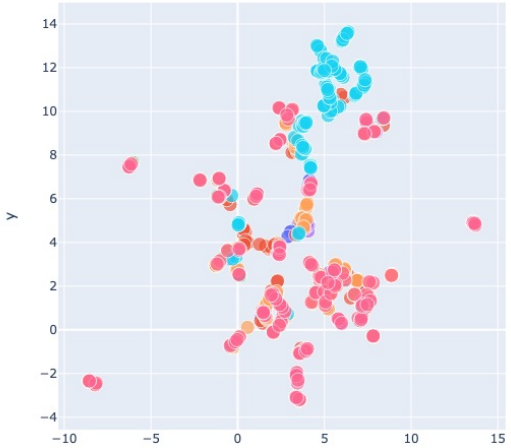


Representation learning: code embeddings

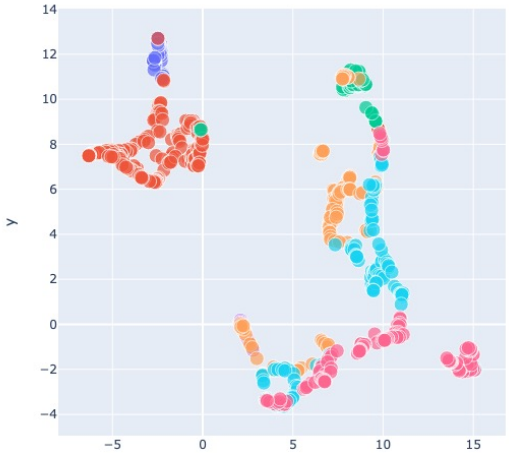


Representation learning: Large Language Models

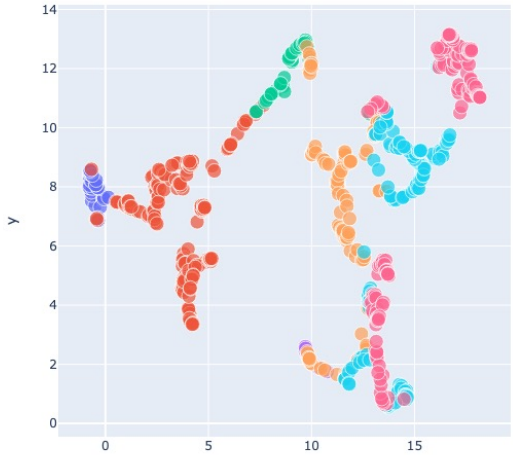
OpenAI-OpenAI3-Large



deepseek-coder-6.7B-instruct-GGUF-deepseek-coder-6.7B



umap projection for SFR-Embedding-Mistral-GGUF-SFR-Er



UNIVERSITÉ LAVAL

Representation learning: Data Collection

- Curated list of microservices applications in various programming languages used in research
- A list of “special” applications that can be considered due to factors such as having a monolithic version, the scale of the application and the structure of the repository
- Keyword query: [regex=“micro(|-)?services?(|-)(architecture|system|application)”]
- Collected 154 Java microservices applications and 91 C# microservices applications

Representation learning: Evaluation (1/2)

Evaluating the distribution of embeddings in the context Monolith to Microservices:

1. *Generate the embeddings for each class or method in an application*
2. *Measure the similarity between each couple of classes/methods (for example cosine similarity)*
3. *Measure the binary cross entropy loss based on the actual decomposition.*
4. *Evaluate the models based on the mean score across all applications.*

Model Name	SFR-Embedding-Mistral	deepseek-coder-6.7B-instruct	OpenAI	Code2Vec	CodeBERT	semantic-analysis	structural-analysis
Mean score	0.704	0.736	0.757	0.816	0.873	0.964	19.700

Representation learning: Evaluation (2/2)

Evaluating the distance between the generated decomposition and the actual decomposition:

1. *Generate the embeddings for each class or method in an application*
2. *Generate a decomposition for each algorithm (k-means, hierarchical clustering, dbscan, etc)*
3. *Measure the MSFB score defined in the RLDec approach*
4. *Evaluate the models based on the mean score across all applications and algorithms.*

Model Name	SFR-Embedding-Mistral	deepseek-coder-6.7B-instruct	OpenAI	CodeBERT	Code2Vec	semantic-analysis	structural-analysis
MSFB	0.300	0.290	0.283	0.250	0.247	0.187	0.182

The decomposition platform: Current challenges

- Increasing number of decomposition approaches

- Difficulty of reproducing existing approaches

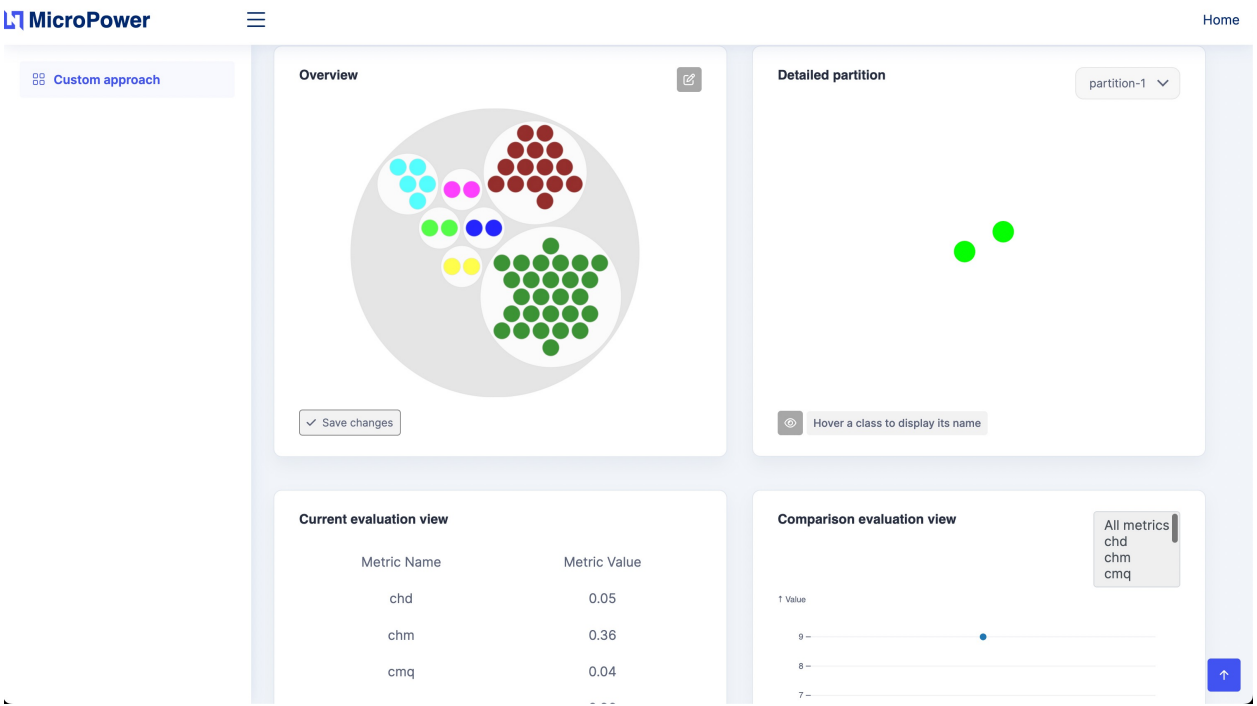
- Varied number of evaluation metrics and their implementations

- Multiple benchmark monolithic applications

- Lack of visualization tools for decompositions

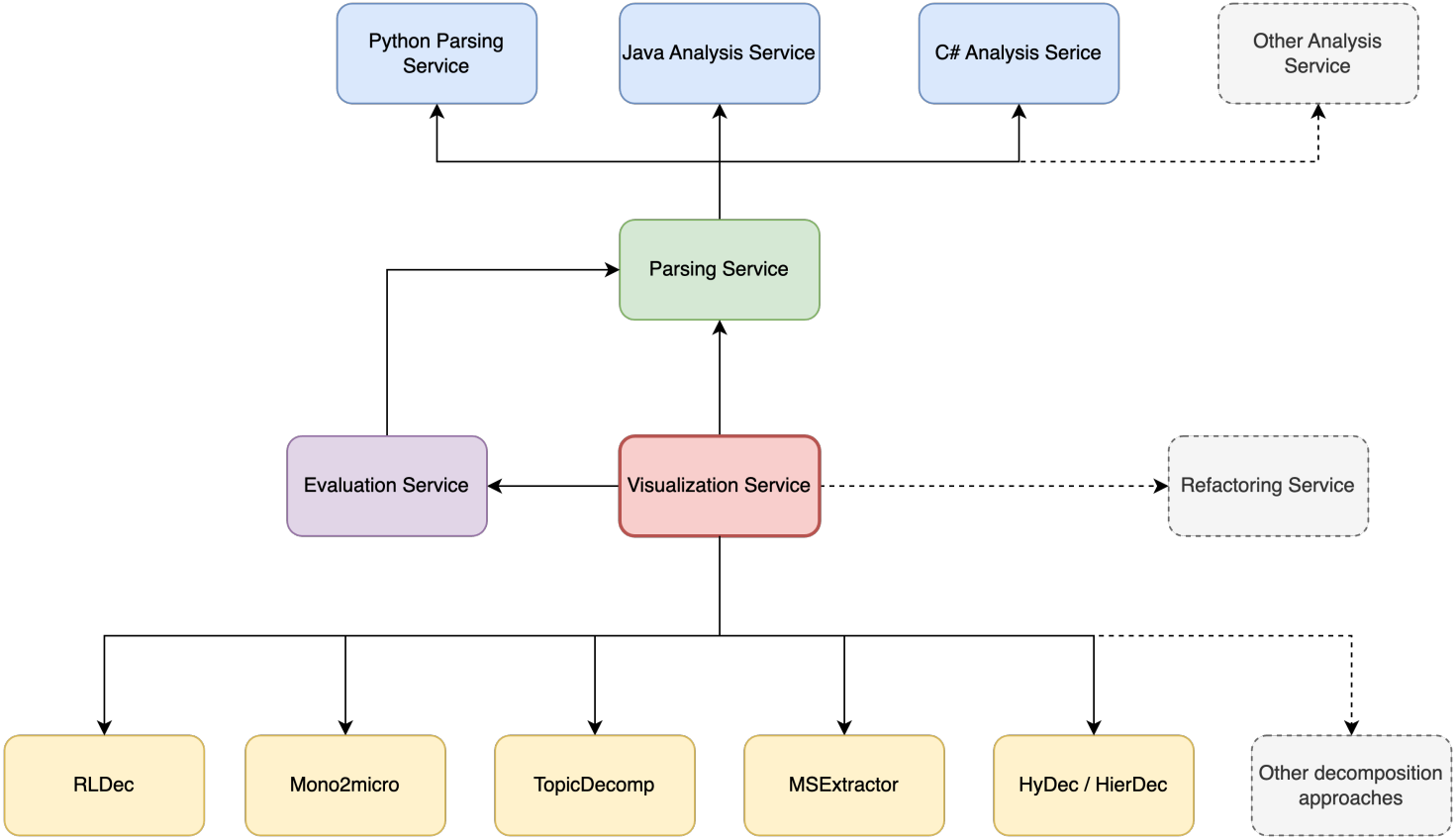
The decomposition platform: Objective

Objective: Share a standardized platform for applying and evaluating decomposition approaches and visualizing decompositions



UNIVERSITÉ LAVAL

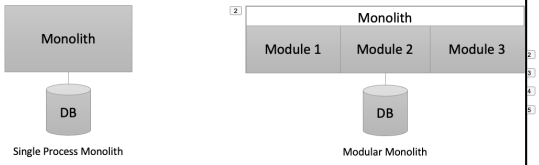
The decomposition platform: Architecture



Conclusion

The Monolithic Architecture (1/2)

- **Definition**: A Monolith is a single unit of deployment.

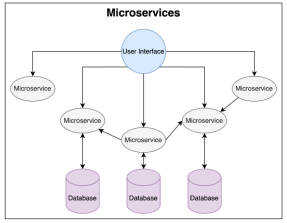


- Simple development process

- Lack of scalability
- High coupling

The Microservices Architecture (1/2)

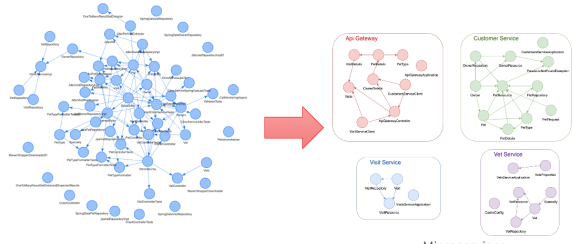
- Independent deployability
- Business Domain Driven
- Small microservices
- Modularity



- Horizontal Scalability
- Robustness
- Technology diversity
- Clear isolation
- Convient for cloud and devops

- Implementation complexity
- Deployment overhead
- Adapting to new workflows

Migrating from the Monolith to Microservices: what is it?



Monolith

Microservices

HierDecomp: Introduction

A Hierarchical DBSCAN Method for Extracting Microservices from Monolithic Applications

The International Conference on Evaluation and Assessment in Software Engineering 2022 (EASE2022)

Main contributions:

- A hierarchical decomposition suggestion for result explainability and user choice flexibility.
- Number of target microservices is inferred.
- Introduce a new evaluation approach for microservices decomposition.

HyDec: introduction

Combining Static and Dynamic Analysis to Decompose Monolithic Application into Microservices

The 20th International Conference on Service-Oriented Computing 2022 (ICSOC2022)

Main contributions:

- A general approach to combine multiple analysis sources in order to generate hierarchical decompositions.
- Multiple combination approaches.
- A decomposition approach that improves the coverage while maintaining a performance similar to state-of-the-art approaches.

MSExtractor: Introduction

Improving microservices extraction using evolutionary search

The Journal of Information and Software Technology Volume 151

Main contributions:

- Formulating the microservices decomposition task as a search problem with an evolutionary algorithm.
- Using a multi-objective evolutionary algorithm in order to encapsulate the different aspects within a decomposition.
- Differentiating between interface and inner classes within a decomposition.

RLDec: Introduction

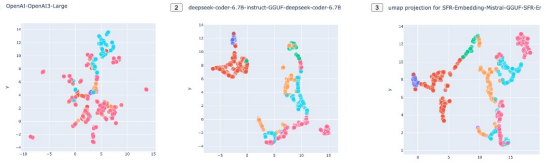
Extracting Microservices from Monolithic Systems using Deep Reinforcement Learning

***In Review** for The Empirical Software Engineering*

Main contributions:

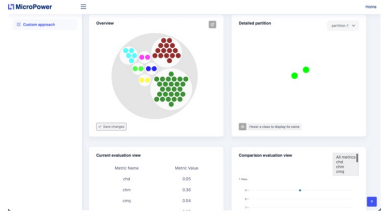
- Formulate the microservices decomposition problem as a reinforcement learning task.
- Improving the evaluation process by introducing novel metrics that can encapsulate multiple aspects and that can compare with existing decompositions.

Representation learning: Large Language Models



The decomposition platform: Objective

Objective: Share a standardized platform for applying and evaluating decomposition approaches and visualizing decompositions



UNIVERSITÉ LAVAL

References

[1] Sean Treadway: https://developers.soundcloud.com/blog/evolution-of-soundclouds-architecture

[2] Stephen Sun: https://www.fullstackexpress.io/p/evolution-soundcloud-architecture-final

[3] Anup K. Kalia, Jin Xiao, Rahul Krishna, Saurabh Sinha, Maja Vukovic, and Debasish Banerjee. Mono2micro: A practical and effective tool for decomposing monolithic java applications to microservices. pages 1214–1224. Association for Computing Machinery, Inc, 8 2021. ISBN 9781450385626. doi: 10.1145/3468264.3473915. URL https://dl.acm.org/doi/10.1145/3468264.3473915.

[4] Wuxia Jin, Ting Liu, Yuanfang Cai, Rick Kazman, Ran Mo, and Qinghua Zheng. Service candidate identification from monolithic systems based on execution traces. IEEE Transactions on Software Engineering, 47:987–1007, 5 2021. ISSN 19393520. doi: 10.1109/TSE.2019.2910531. URL https://ieeexplore.ieee.org/document/8686152.

[5] Davide Taibi and Kari Systä. From monolithic systems to microservices: A decomposition framework based on process mining. pages 153–164. SciTePress, 2019. ISBN 9789897583650. doi: 10.5220/0007755901530164. URL https://www.scitepress.org/Link.aspx?doi=10.5220/0007755901530164.

[6] Utkarsh Desai, Sambaran Bandyopadhyay, and Srikanth Tamilselvam. Graph neural network to dilute outliers for refactoring monolith application. In AAAI Conference on Artificial Intelligence, 2 2021. URL http://arxiv.org/abs/2102.03827.

[7] Alex Mathai, Sambaran Bandyopadhyay, Utkarsh Desai, and Srikanth Tamilselvam. Monolith to microservices: Representing application software through heterogeneous graph neural network. 2022. URL https://arxiv.org/abs/2112.01317https://www.ijcai.org/proceedings/2022/542.

[8] Manabu Kamimura, Keisuke Yano, Tomomi Hatano, and Akihiko Matsuo. Extracting candidates of microservices from monolithic application code. volume 2018-December, pages 571–580. IEEE Computer Society, 7 2018. ISBN 9781728119700. doi: 10.1109/APSEC.2018.00072.

[9] Miguel Brito, Jácome Cunha, and João Saraiva. Identification of microservices from monolithic applications through topic modelling. 2021. doi: 10.1145/3412841.3442016.

[10] Luís Nunes, Nuno Santos, and António Rito Silva. From a monolith to a microservices architecture: An approach based on transactional contexts. volume 11681 LNCS, pages 37–52. Springer Verlag, 2019. ISBN 9783030299828. doi: 10.1007/978-3-030-29983-5_3.

[11] Omar Al-Debagy and Péter Martinek. A microservice decomposition method through using distributed representation of source code. Scalable Computing, 22:39–52, 2021. ISSN 18951767. doi: 10.12694:/scpe.v22i1.1836. URL https://www.scpe.org/index.php/scpe/article/view/1836.

[12] Shanshan Li, He Zhang, Zijia Jia, Zheng Li, Cheng Zhang, Jiaqi Li, Qiuya Gao, Jidong Ge, and Zhihao Shan. A dataflow-driven approach to identifying microservices from monolithic applications. Journal of Systems and Software, 157:110380, 2019. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2019.07.008. URL https://www.sciencedirect.com/science/article/pii/S0164121219301475.

[13] Genc Mazlami, Jurgen Cito, and Philipp Leitner. Extraction of microservices from monolithic software architectures. pages 524–531. Institute of Electrical and Electronics Engineers Inc., 9 2017. ISBN 9781538607527. doi: 10.1109/ICWS.2017.61. URL https://ieeexplore.ieee.org/document/8029803.

[14] Michael Gysel, Lukas Kölbener, Wolfgang Giersche, and Olaf Zimmermann. Service cutter: A systematic approach to service decomposition. volume 9846 LNCS, 2016. doi: 10.1007/978-3-319-44482-6_12.

[15] Yamina Romani, Okba Tibermacine, and Chouki Tibermacine. 2022. Towards Migrating Legacy Software Systems to Microservice-based Architectures: a Data-Centric Process for Microservice Identification. In 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C). 15–19.

[16] Vikram Nitin, Shubhi Asthana, Baishakhi Ray, and Rahul Krishna. 2023. CARGO: AI-Guided Dependency Analysis for Migrating Monolithic Applications to Microservices Architecture. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering. Article 20, 12 pages.

[17] Wesley K. G. Assunção, Thelma Elita Colanzi, Luiz Carvalho, Juliana Alves Pereira, Alessandro Garcia, Maria Julia de Lima, and Carlos Lucena. 2021. A Multi-Criteria Strategy for Redesigning Legacy Features as Microservices: An Industrial Case Study. In 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). 377–387.

[18] Rahul Yedida, Rahul Krishna, Anup Kalia, Tim Menzies, Jin Xiao, Maja Vukovic, Partitioning cloud-based microservices (via deep learning), 2021, arXiv preprint arXiv:2109.14569.

[19] Zhiding Li, Chenqi Shang, Jianjie Wu, Yuan Li, Microservice extraction based on knowledge graph from monolithic applications, Information and Software Technology, Volume 150, 2022, 106992, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2022.106992.