# Automating microservices orchestration & data-driven evolutionary architectures

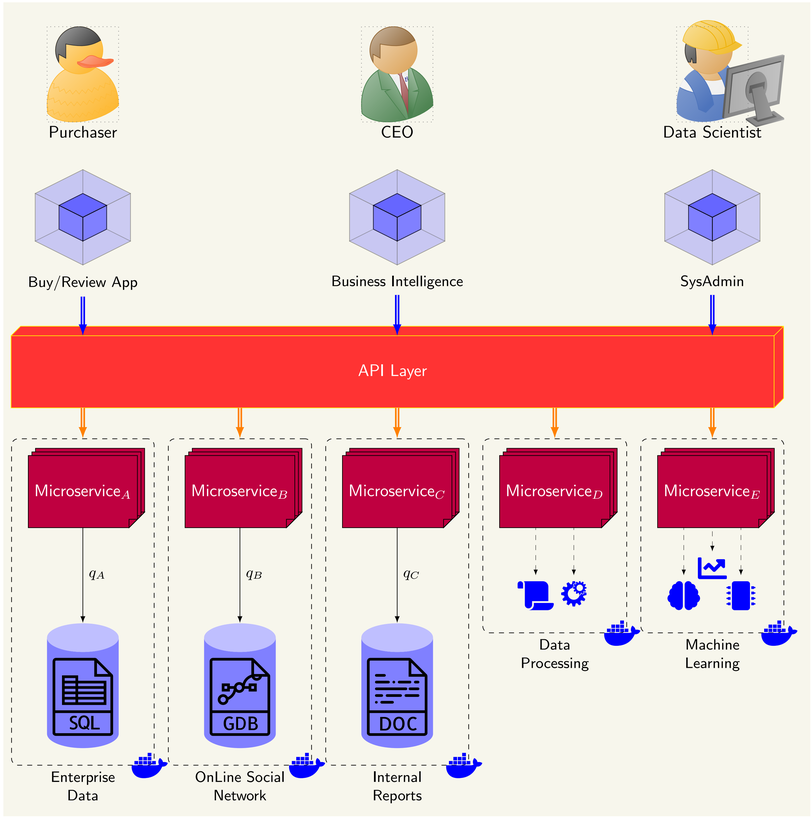
Giacomo Bergami (School of Computing, Newcastle University)

## Implementation Goals

- Using Data Integration as a major driving force for composing microservices and deriving data transformation functions.
- Using Evolutionary Architectures for distributing computations for machine learning pipelines.
- Exploiting Global-As-View methodologies for projecting user queries into distributed nodes while running data integration tasks.
- Delivering trained models as data to be shared across nodes.
- Deriving Certified Fitness Functions for determining the architecture composition fitness before deployment and run-time testing.

## Introduction

The rapidly evolving landscape of computer and data science entails the early adoption of new technologies in pre-existing ecosystems. Evolution is not necessarily gradual, as this might trigger a massive refactoring in pre-existing systems. The major forces driving such changes are the piling-up of additional business functionality required by the new customer [1], the improvement of pre-existing functionalities by delivering alternative solutions [2], or still the disruptive introduction of a new technology requiring a massive restructuring of the entire software ecosystem.Footnote 1 In this scenario, long-term planning is neither possible nor sustainable, as the current trends suggest that any future technology will drastically differ from the currently available ones. This then motivates the adoption of evolutionary architectures, which are usually characterised in terms of traditional software architectures (topology) as well as of (non-)functional requirements and the way to carry out computations through explicitly wired or "linkable" components (governance). [1].




## Important Result

Notwithstanding the existence of current automated pipelines to integrate data for AI and Machine Learning purposes, none of the ones available from current literature allow data integration tasks, which should always be a pre-processing step. Furthermore, none of the available microservice composition services exploit data integration tools for composing dynamically available microservices exposing different data interfaces.

## Materials

The following materials were required to complete the research:

- Relational Learning
- Data-Aware Data Mining
- Near-Data Processing
- Logical-Driven AI
- Ontology Alignments
- Explainable AI
- Decidable Programming Lanugages (non Turing Complete)
- Microservices Orchestration

## Methods

Rather than exploiting training-based AI which might provide unexpected results in highly voliatile environment, we postulate for the need of using decidable computations using the Rule of Least Power. This will allow us to completely control the overall pipeline, while extracting important properties from the programs being run within the architecture (e.g., *correctness*, *running time*).

## Generalised Semistructured Model

The possibility of sharing data across disparate microservices also requires defining a brand new data model encompassing all the currently-available data models (Graphs, JSON, XML, RBDMS) as well as supporting all the currently-available query languages. To fullfill this aim, we then postulate the need for a generalised data model and query language supporting all of the above [2].

## References

[1] Giacomo Bergami.
Towards automating microservices orchestration through data-driven evolutionary architectures.
*Service Oriented Computing and Applications*, 18(1):1–12, Mar 2024.

[2] Giacomo Bergami and Wiktor Zegadło.
Towards a generalised semistructured data model and query language.
*SIGWEB Newsl.*, 2023(Summer), aug 2023.

## Acknowledgements

We thank Michael Fronistas (UG) and Tauras Stasiulionis (UG) for the preliminary implementation of MADS. We thank Wiktor Zegadło for the first implementation of the Generalised Semistructured Model (GSM).

## Contact Information

- Web: https://jackbergus.github.io
- Email: giacomo.bergami@newcastle.ac.uk

AiRLInE
Paraconsistent AI

MADS