

# A Vision to identify Architectural Smells in Self-Adaptive Systems using Behavioral Maps

Edilton Lima dos Santos  
[edilton.limados@unamur.be](mailto:edilton.limados@unamur.be)

Sophie Fortz  
Gilles Perrouin  
Pierre-Yves Schobbens

CASA@ECSA 2021, Växjö, Sweden, September 13th - 14th, 2021.



# *Agenda*

**Motivation**

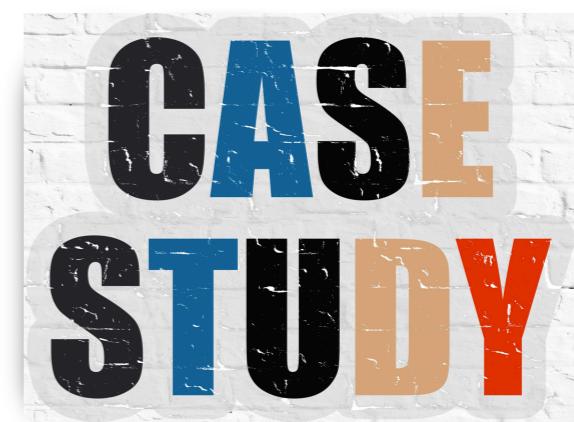
**Proposed  
approach**

*Behavioral Map*

**Processes**

**Algorithm**

*Framework*



# Motivation

- Self-adaptive systems **change** their **behavior depending** on **environmental changes** and **reconfiguration plans**.
- Dynamic Software Product Line (**DSPL**) engineering **implements self-adaptive systems** by dynamically **binding or unbinding features**.
- **Binding or unbinding features** as prescribed by a **feature model**.

# Motivation

- Challenges
  - **Number of possible configuration** goes exponentially with the number of features;
  - The **(re)configuration** process **may add a new architectural solution** in an **inappropriate context** via new features loaded;
  - The **(re)configuration** process **may combine architectural fragments** with **undesirable behaviors**.
- Problem
  - Architectural Smell.

# Motivation

- **Studies about Architectural Smell in SAS:**

- C. Raibulet, F. A. Fontana, S. Caretoni, **A preliminary analysis of self-adaptive systems according to different issues**, Software Quality Journal (2020) 1–31.
- M. A. Serikawa, A. d. S. Landi, B. R. Siqueira, R. S. Costa, F. C. Ferrari, R. Menotti, V. V. De Camargo, **Towards the characterization of monitor smells in adaptive systems**, in: X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), IEEE, 2016, pp. 51–60.

# Proposed approach

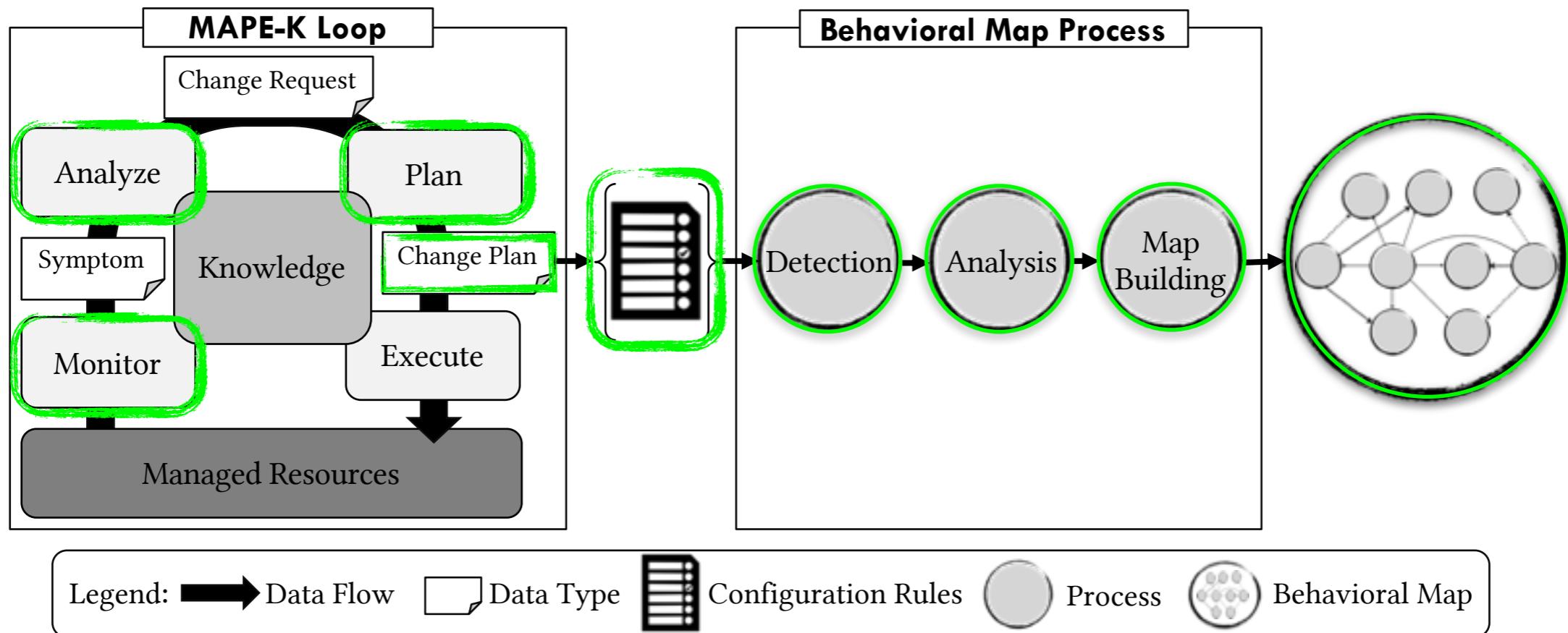
- We introduce **the**

## Behavioral Map

- A Behavioral Map (BM) can be seen as a *hybrid structure*, data, and control information about the self-adaptive system.
- BM maps the interactions and influences that a feature has on other features in a specific configuration for a given runtime context.
- The BM can support architectural bad smell identification at runtime.

# Behavioral Map

# Processes

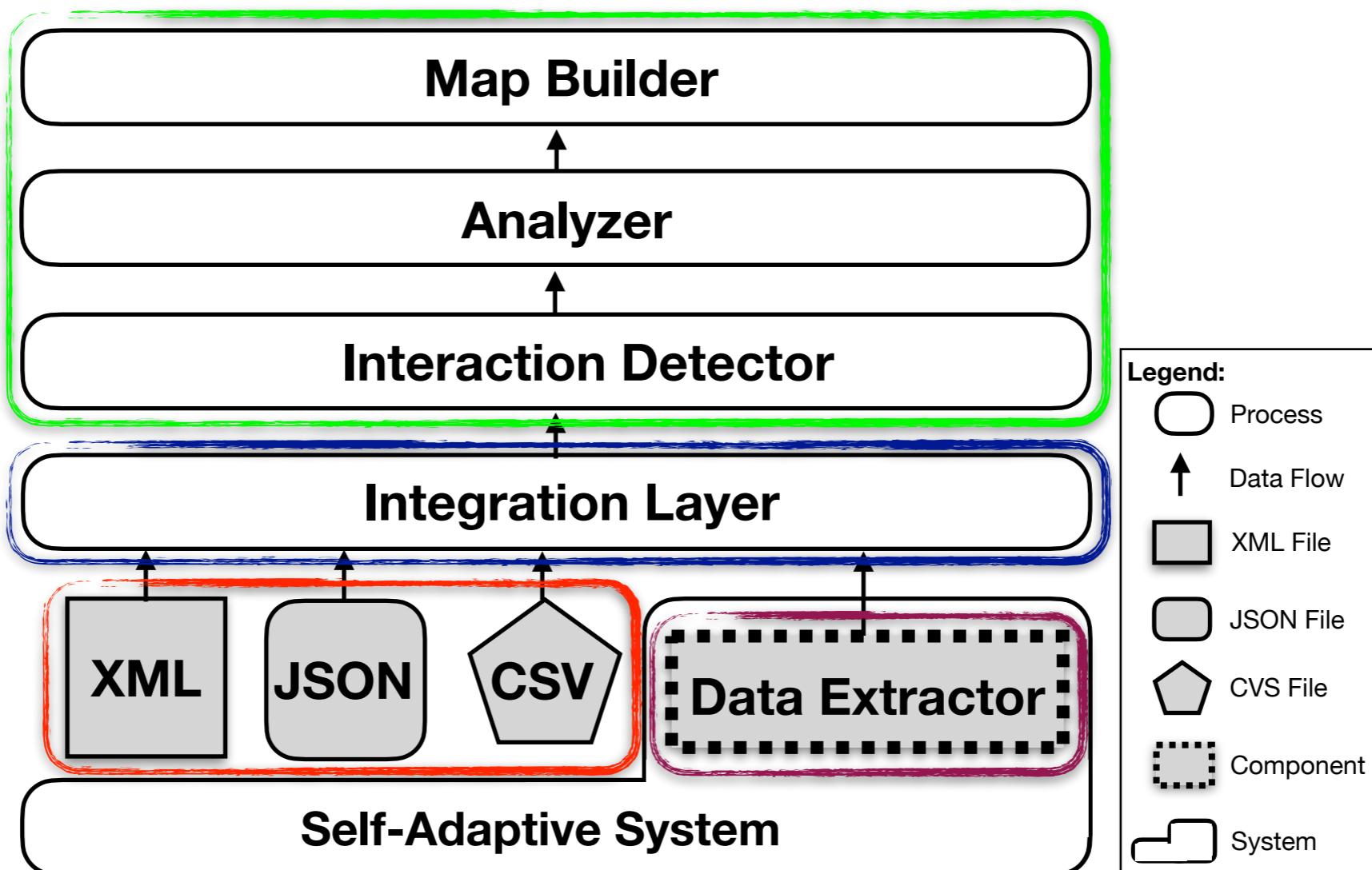


# Algorithm

```
1 table ← loadConfigurationRulesFile(CRfile);
2 verticesOnMap ← createVerticesOnMap(table);
3 foreach vertex in verticesOnMap do
4     foreach row in table do
5         if row.name.equals(vertex.name) then
6             foreach relation in row.getAllRelationships() do
7                 if relation.relationship is not null then
8                     createEdge(vertex, relation.relationship_type, relation.featureName);
9                 end
10            end
11        end
12    end
13 end
```

Behavioral  
Map

# Framework Architecture

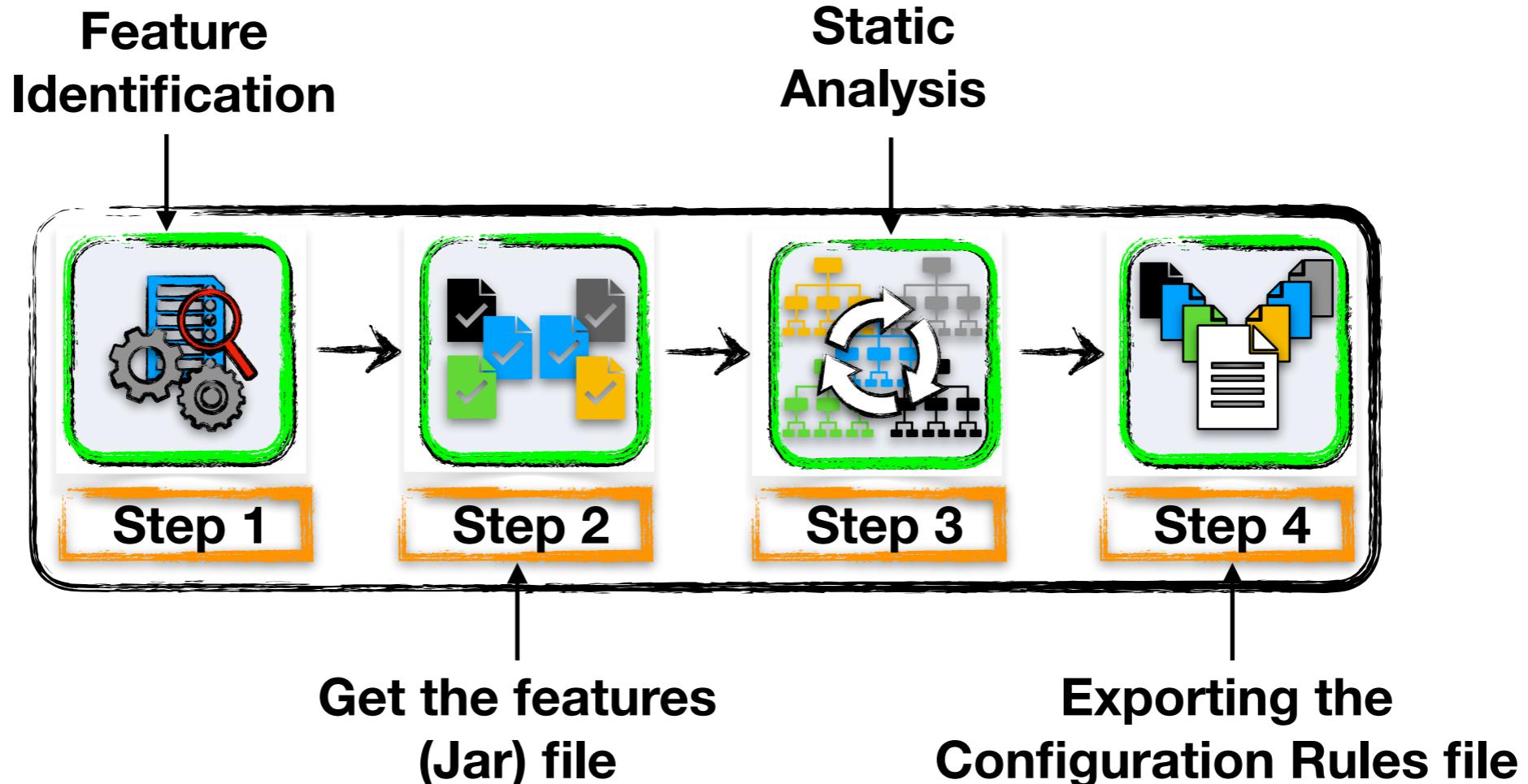


*Behavioral  
Map*

# Data Extractor Process



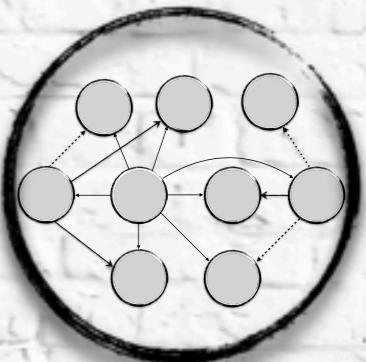
**WALA**  
T. J. WATSON LIBRARIES FOR ANALYSIS



The Behavioral Map was implemented

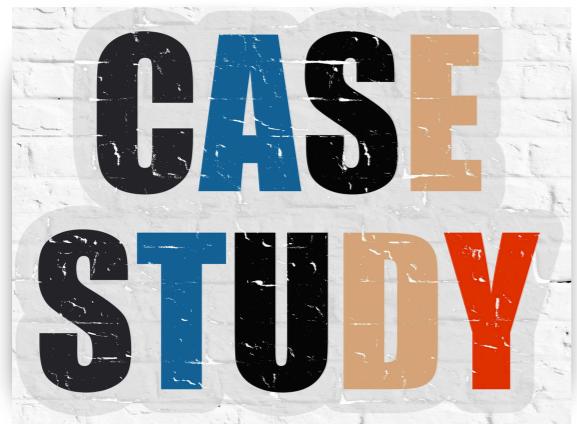
using

# Behavioral Map



## CASE STUDY



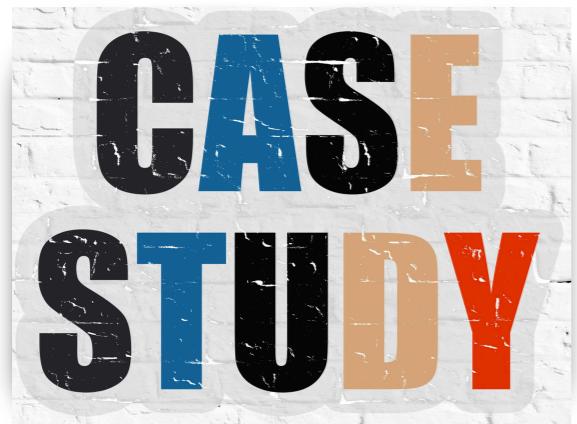


# Architectural Smell List

Smell Name	Detection
Cyclic Dependency (CD) [16]	Full
Extraneous Connector (EC) [8]	Full
Hub-Like Dependency (HL) [16, 10]	Full
Oppressed Monitors (OM)[11]	Partial

C. Raiblet, F. A. Fontana, S. Caretoni, **A preliminary analysis of self-adaptive systems according to different issues**, Software Quality Journal (2020) 1–31.

M. A. Serikawa, A. d. S. Landi, B. R. Siqueira, R. S. Costa, F. C. Ferrari, R. Menotti, V. V. De Camargo, **Towards the characterization of monitor smells in adaptive systems**, in: X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), IEEE, 2016, pp. 51–60.



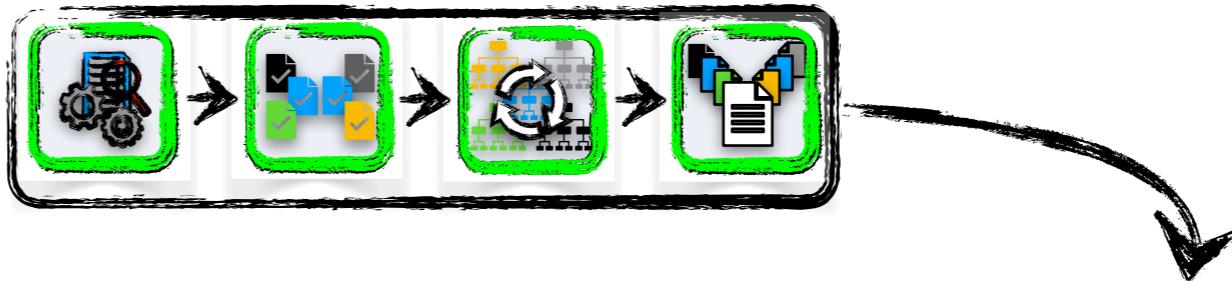
# The SHE Smart Home System

- We applied the **BM** framework on SHE smart home system based on MAPE-K loop and implemented using the publish-subscribe architecture.
- The SHE is composed by:
  - **Core features:** Manager, Listener, Loader, Installer, and Presentation Layer.
  - We included **four optional features** as follows:
    - **Luminosity:** used to read data from the luminosity sensor;
    - **Presence:** used to read data from the presence sensor;
    - **lampController:** responsible for controlling Lamp feature's behavior using the information read from Luminosity and Presence features;
    - **Lamp:** an actuator used to switch on and off lights based on the lampController feature's data.

Case study available in: <https://github.com/edilton-santos/BehavioralMapExample>

# CASE STUDY

## Running the Data Extractor

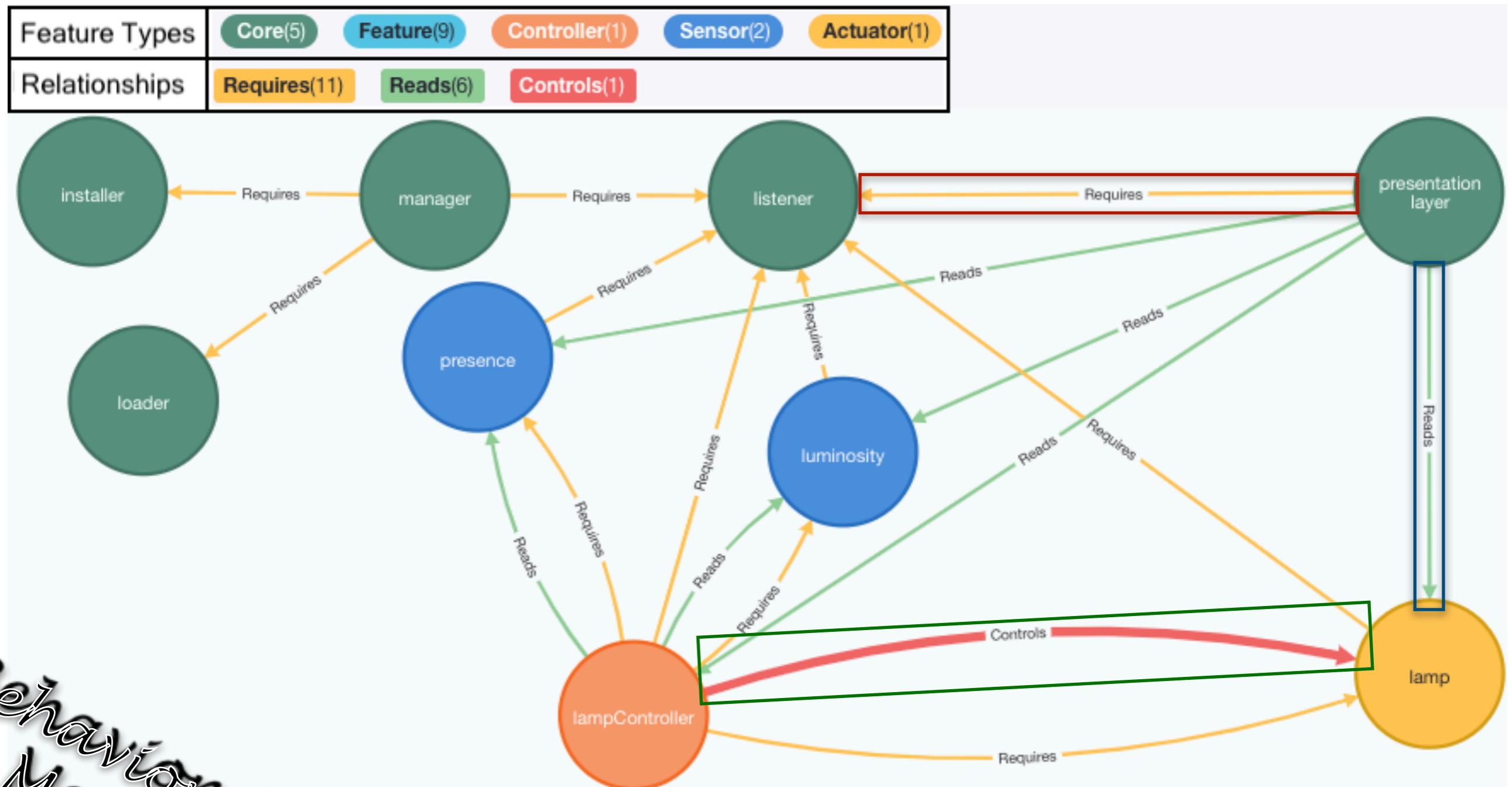


```
{  
    "imported_packages": ["com.she.core.listener"],  
    "relationships": [  
        {"relationship_type": "Requires", "feature_name": "listener"},  
        {"relationship_type": "Reads", "feature_name": "lampController"},  
        {"relationship_type": "Reads", "feature_name": "luminosity"},  
        {"relationship_type": "Reads", "feature_name": "presence"},  
        {"relationship_type": "Reads", "feature_name": "lamp"}  
],  
    "name": "presentation layer",  
    "type": "Core",  
    "version": "1.0.0",  
    "status": "Ativo",  

```

# CASE STUDY

## Identifying Architectural Bad Smells



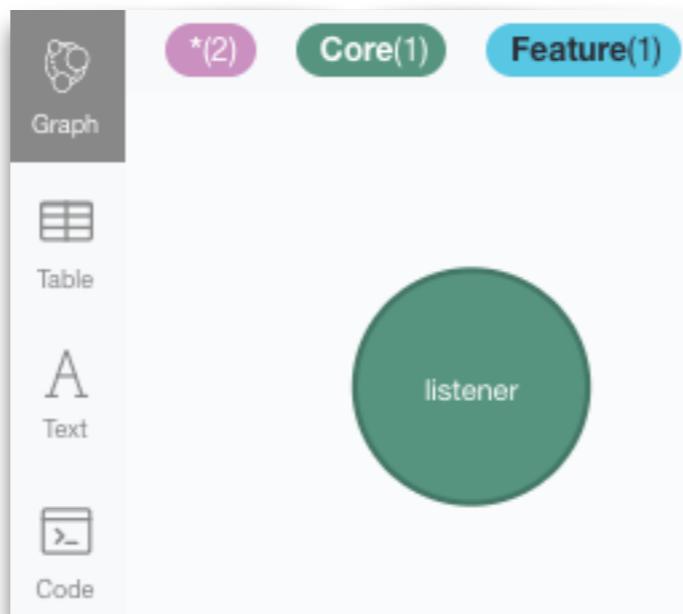
# CASE STUDY

## Identifying Architectural Bad Smells

**Cypher query:**

```
// Hub-Like Dependency
MATCH (f:Feature)-[r:Requires]→(f2:Feature)
OPTIONAL MATCH (f2)-[r2:Requires]→(:Feature)
WITH f2, count(r) AS Rtotal, count(r2) AS Rtotal2
WHERE Rtotal ≥ 5 AND Rtotal > Rtotal2
RETURN f2, Rtotal, Rtotal2
```

**Result:**



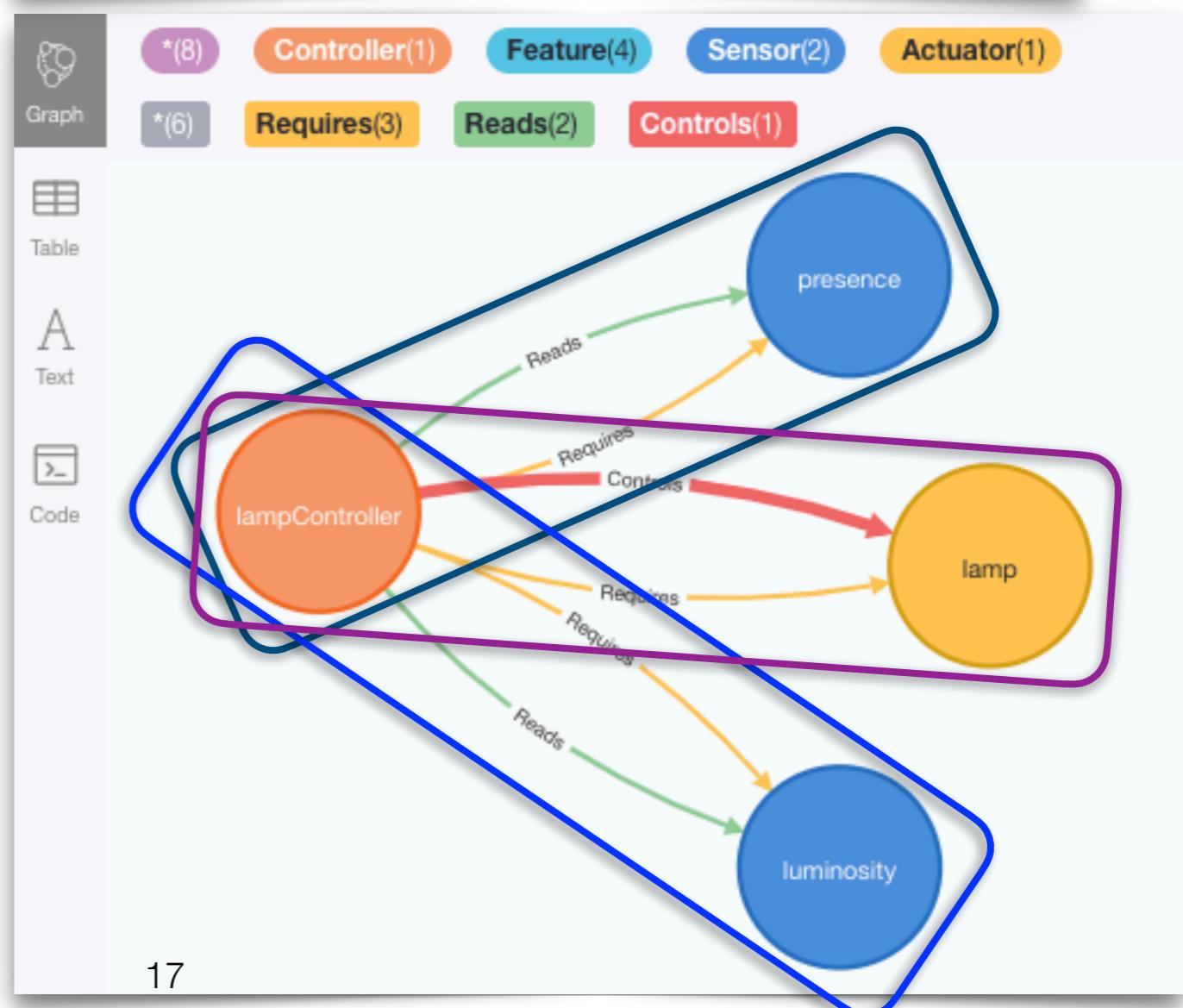
# CASE STUDY

## Identifying Architectural Bad Smells

**Cypher query:**

```
// Extraneous Connector
MATCH (f:Feature)-[r:Requires]→(f2:Feature)
WHERE exists((f)-[:Reads|Controls]→(f2))
RETURN f, r, f2
```

**Result:**



Behavioral Map

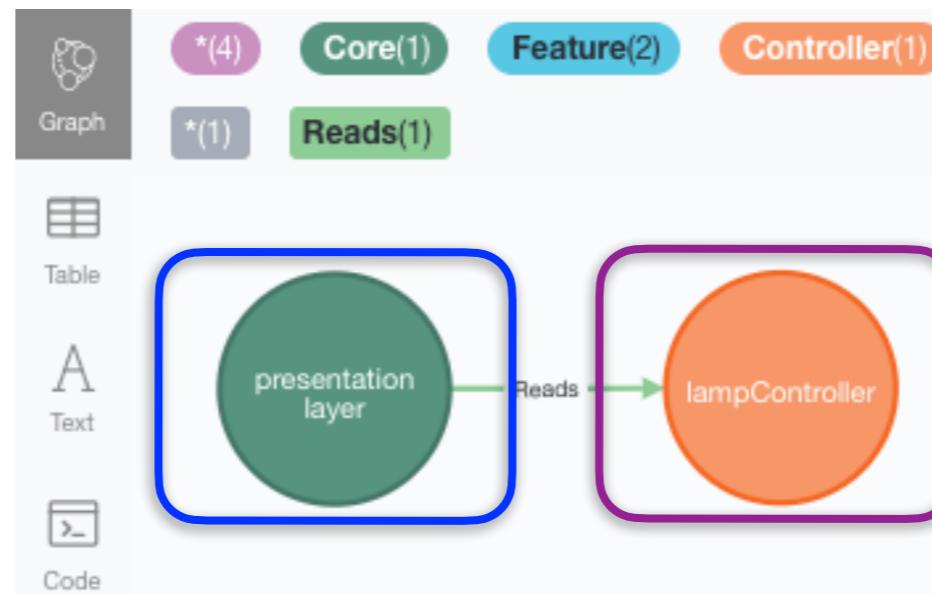
# CASE STUDY

## Identifying Architectural Bad Smells

Cypher query:

```
1 // Look for Oppressed Monitors
2 MATCH (f1:Feature)-[r:Reads]→(:Feature)
3 WITH f1, count(r) As Rtotal
4 WHERE Rtotal ≥ 2
5 RETURN f1, Rtotal
```

Result:



Behavioral  
Map

# CASE STUDY

## Identifying Architectural Bad Smells

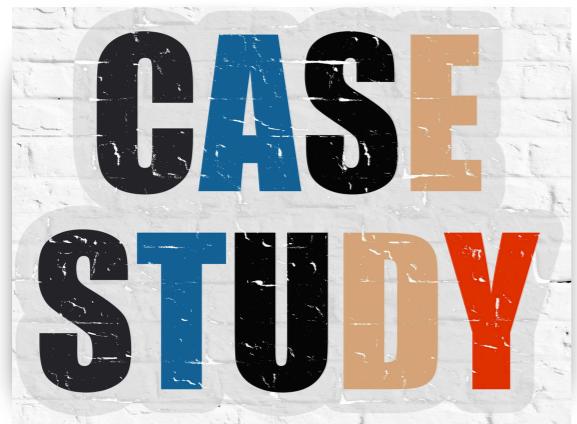
**Cypher query:**

```
1 // Cyclic Dependency
2 MATCH (f:Feature)-[:Requires]→(f2:Feature)-[:Requires]→(f)
3 OPTIONAL MATCH (f2)-[:Requires]→(f3:Feature)-[:Requires]→(f)
4 RETURN f, f2, f3
```

**Result:**

The screenshot shows the Neo4j browser interface. On the left, there are two tabs: 'Table' (selected) and 'Code'. The main area displays the result of a Cypher query: '(no changes, no records)'.

Behavioral  
Map



# Identifying Architectural Bad Smells

<https://github.com/edilton-santos/BehavioralMapExample>

edilton-santos / BehavioralMapExample

< Code Issues Pull requests Actions Projects Wiki Security Insights

main · 1 branch · 0 tags Go to file Code

edilton-santos	Update extraneous-connector.cypher	0be1faf 2 hours ago	16 commits
Behavioral Map scripts	Update extraneous-connector.cypher	2 hours ago	
Neo4J APOC Configuration	Add files via upload	2 months ago	
Neo4J and APOC configuration.pdf	Add files via upload	2 months ago	
README.md	Update README.md	last month	
crFileSHEstudy1.json	Add files via upload	2 months ago	
crFileSHEstudy2.json	Add files via upload	2 months ago	

README.md

## Behavioral Map Example

Before running the examples, you need to install Neo4J Desktop. You can do the [download](#) the Neo4J Desktop on this website <https://neo4j.com/download/?ref=try-neo4j-ip>.

The folder **Neo4J APOC Configuration** stores the APOC API configuration file. **Neo4J and APOC configuration.pdf** file describes how to create a local database and APOC configuration process.

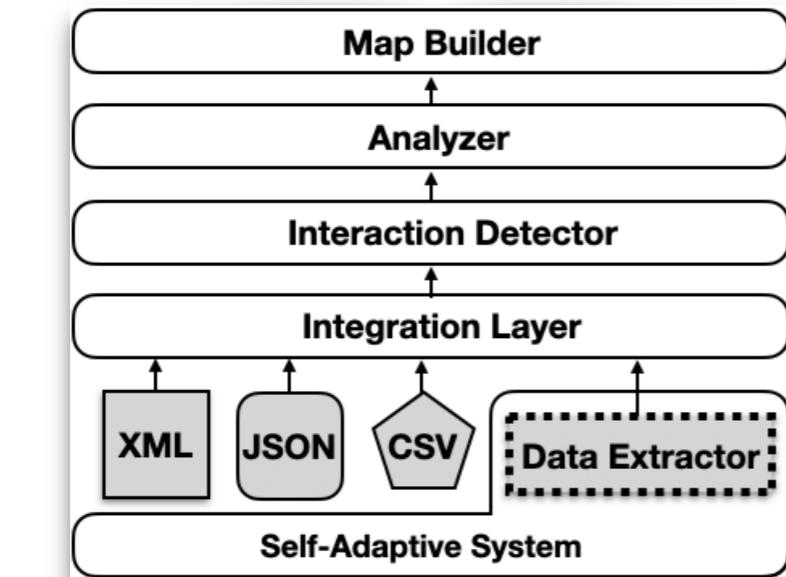
We provide two Configuration Rules (CR) files (**crFileSHEstudy1.json**, and **crFileSHEstudy2.json**) that can be used to create the Behavioral Map and look for Architectural Smell using the scripts are available in the folder **Behavioral Map scripts**.

Scripts available in the folder **Behavioral Map scripts**:

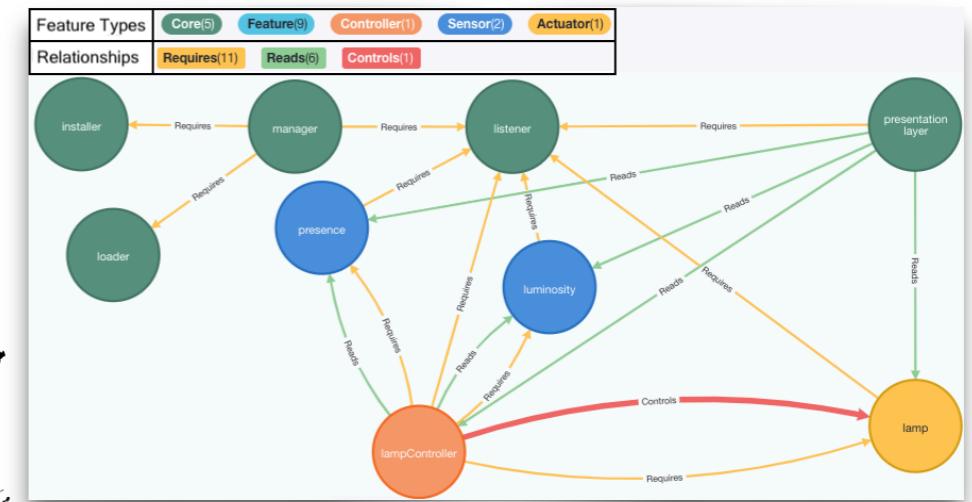
- **import-cr-file-and-create-the-behavioral-map.cypher**: Script used to import CR file and create the Behavioral Map.



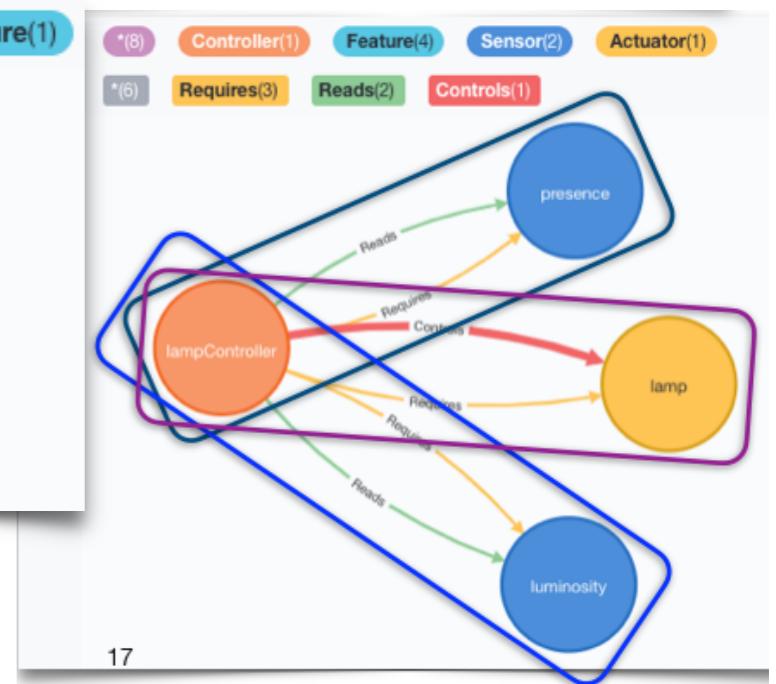
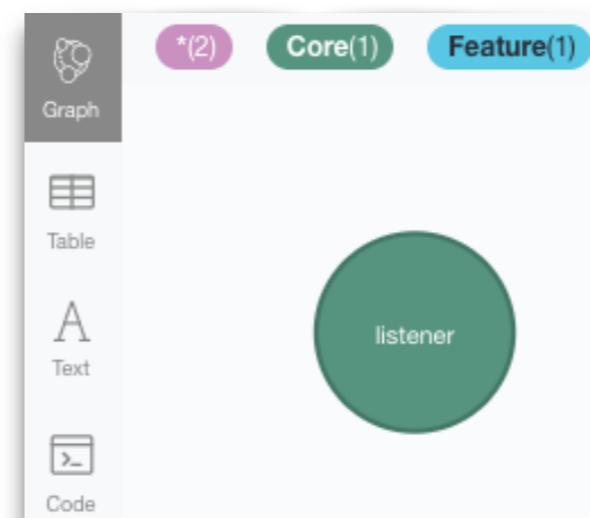
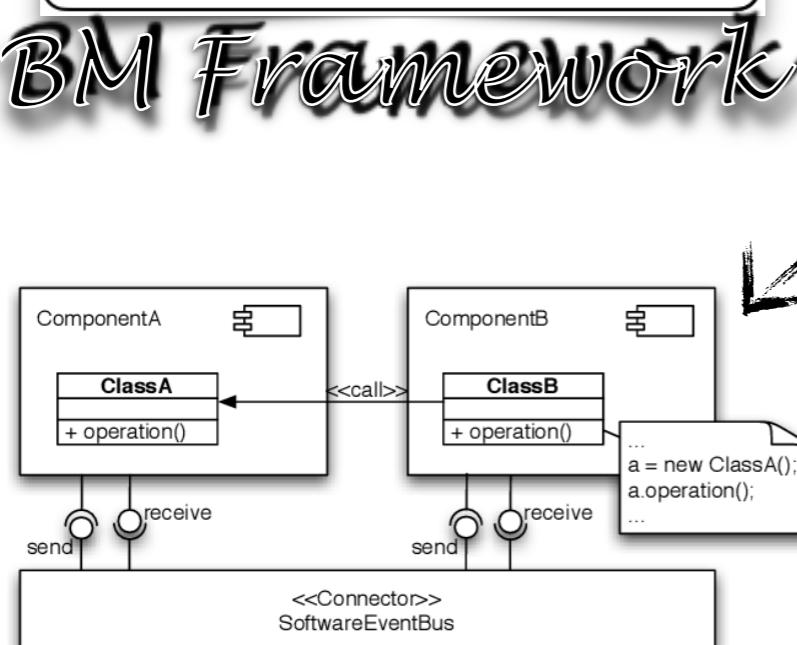
# CASE STUDY



# Conclusion



**Behavioral Map**



A large, dense collage of the words "thank you" in many different languages, arranged in a roughly circular pattern. The languages include English, Spanish, Portuguese, French, German, Italian, Russian, Chinese, Japanese, Korean, Vietnamese, Polish, and others. The text is in a bold, black, sans-serif font.

# A Vision to identify Architectural Smells in Self-Adaptive Systems using Behavioral Maps

Edilton Lima dos Santos  
[edilton.limados@unamur.be](mailto:edilton.limados@unamur.be)

Sophie Fortz  
Gilles Perrouin  
Pierre-Yves Schobbens

CASA@ECSA 2021, Växjö, Sweden, September 13th - 14th, 2021.

