Emergent Software Systems: Theory and Practice

Roberto Rodrigues Filho - INF/UFG
Barry Porter - Lancaster University, UK
Fábio M. Costa - INF/UFG
Iwens Sene Júnior - INF/UFG

***SBRC'21* - Short course**

robertovito@ufg.br

INF
INSTITUTO DE INFORMÁTICA

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

Course Overview

➔ Theory

- ➔ **Motivation:** Contemporary systems complexity;
- ➔ **Background:** Autonomic Computing and Self-adaptive Systems;
- ➔ **Emergent Software Systems**
  - Introducing the concept
  - Implementation (PAL framework)

Course Overview

➔ Practice

- ➔ **Case studies (practical application of the concept):**
  - Emergent Web Server
  - Emergent Microservice
- ➔ **Tool:** Dana language basic syntax and examples;
- ➔ **Demo:**
  - Interactive Emergent System tool;
  - Emergent Web Server;
- ➔ **Practical:** Hands-on experience with the Emergent Web Server;

Theory

robertovito@ufg.br

Motivation

→ Contemporary Systems Complexity

Motivation

➔ Contemporary Systems Complexity

- ➔ **Volatility:** Constant changes in the operating environment;
- ➔ **Heterogeneity:** Plethora of different devices, OSes, protocols, programming models, and so on;

Gordon S. Blair, "**Complex Distributed Systems: The Need for Fresh Perspectives**". *ICDCS* 2018: 1410-1421

Motivation

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

➔ Contemporary Systems Complexity

- ➔ **Volatility:** Constant changes in the operating environment;
- ➔ **Heterogeneity:** Plethora of different devices, OSes, protocols, programming models, and so on;

Gordon S. Blair, "**Complex Distributed Systems: The Need for Fresh Perspectives**". *ICDCS* 2018: 1410-1421

Motivation

➔ Contemporary Systems Complexity

- ➔ **Volatility:** Constant changes in the operating environment;
- ➔ **Heterogeneity:** Plethora of different devices, OSes, protocols, programming models, and so on;

Spoiler Alert: Emergent Software Systems is an attempt at a fresh perspective.

Gordon S. Blair, "**Complex Distributed Systems: The Need for Fresh Perspectives**". *ICDCS* 2018: 1410-1421

Motivation

INF
INSTITUTO DE INFORMÁTICA
UFG
UNIVERSIDADE FEDERAL DE GOIÁS

→ Contemporary Systems Complexity

Push more responsibility for managing the system to the software itself.

Background

➔ Autonomic and Self-adaptive Systems

- ➔ **Autonomic Computing (IBM vision):** Self-managing systems that take the necessary actions to achieve high-level systems goals;

Jeffrey Kephart, and David Chess. "**The vision of autonomic computing.**" Computer 36.1 (2003): 41-50.

Background

➔ Autonomic and Self-adaptive Systems

➔ **Self-* properties:**

- Self-optimising;
- Self-healing;
- Self-protecting;
- Self-configuring;

Background

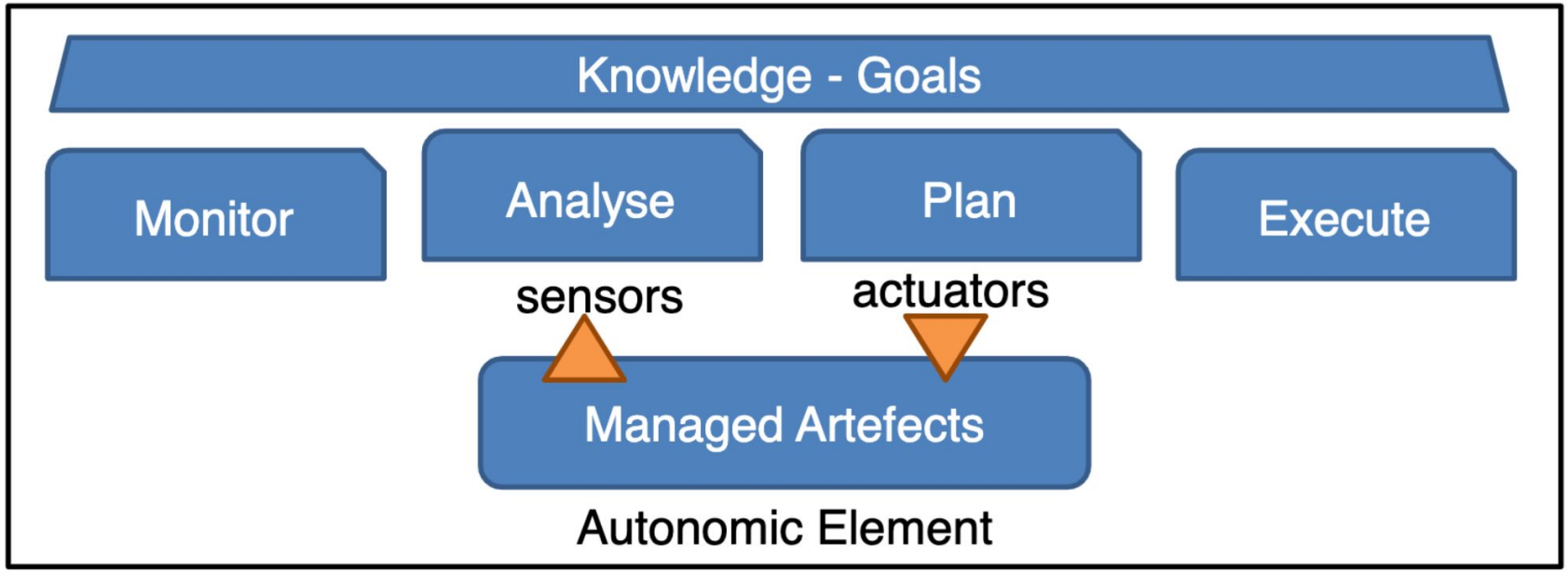
➔ Autonomic and Self-adaptive Systems

- ➔ **Self-adaptive Systems:** Systems that are able to change their structure or behaviour as a response to changes or to maintain systems goals.

Rogério de Lemos, et al. "**Software engineering for self-adaptive systems: A second research roadmap.**" Software Engineering for Self-Adaptive Systems II. Springer, Berlin, Heidelberg, 2013.

Background

→ Autonomic and Self-adaptive Systems: MAPE-K



Background

➔ Autonomic and Self-adaptive Systems

➔ **Developing Autonomic and Self-adaptive Systems is HARD:**

- Develop the system;
- Identify potential autonomic elements;
- Create adaptation points (e.g., feature-models);
- Create adaptation mechanism;
- Create adaptation logic:

Background

➔ Autonomic and Self-adaptive Systems

➔ **Developing Autonomic and Self-adaptive Systems is HARD:**

- Create adaptation logic:
  - Rule-based approach;
  - Logical expressions;
  - Training machine learning and other mathematical models;
  - Reinforcement learning applied to SAS;

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

➔ Introducing the concept

➔ **Emergent software system as an approach to facilitate SAS development.**

Software systems able to build their own understanding of their structure and operating environment and build their adaptation logic accordingly.

Emergent Software Systems

➔ Introducing the concept

- ➔ Emergent software system as an approach to facilitate SAS development.

Component-based models + Reinforcement Learning

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

➔ Introducing the concept

➔ Component-based model:

Component model separates **program logic** from **program structure** (i.e, how logic is wired);

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

Introducing the concept

Component-based model:

- Comparison with an object-oriented programming

```
1   import java.util.ArrayList;
2
3   public class Example {
4       public static void main(String[] args) {
5           String str = "First item on the list";
6           ArrayList<String> myList = new ArrayList<String>();
7           myList.add(str);
8       }
9   }
10
```

Introducing the concept

Component-based model:

- Comparison with an object-oriented programming

```
1   import java.util.ArrayList;
2
3   public class Example {
4       public static void main(String[] args) {
5           String str = "First item on the list";
6           ArrayList<String> myList = new ArrayList<String>();
7           myList.add(str);
8       }
9   }
10
```

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

Introducing the concept

- Classes are statically bound;
- Logic is entangled with structure;
- Generates a single binary (monolith);

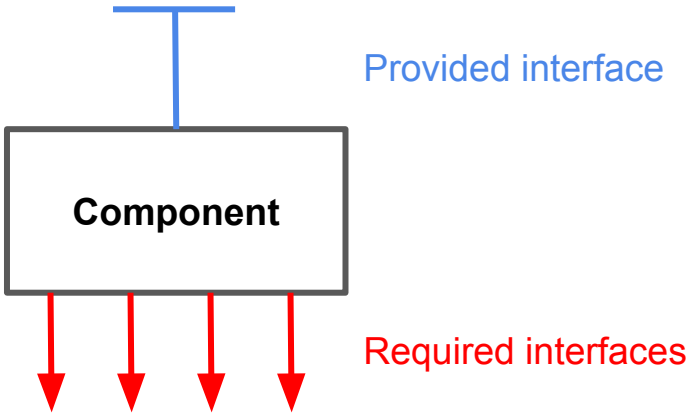
Component-based model:

- Comparison with an object-oriented programming

```
1   import java.util.ArrayList;
2
3   public class Example {
4       public static void main(String[] args) {
5           String str = "First item on the list";
6           ArrayList<String> myList = new ArrayList<String>();
7           myList.add(str);
8       }
9   }
10
```

Introducing the concept

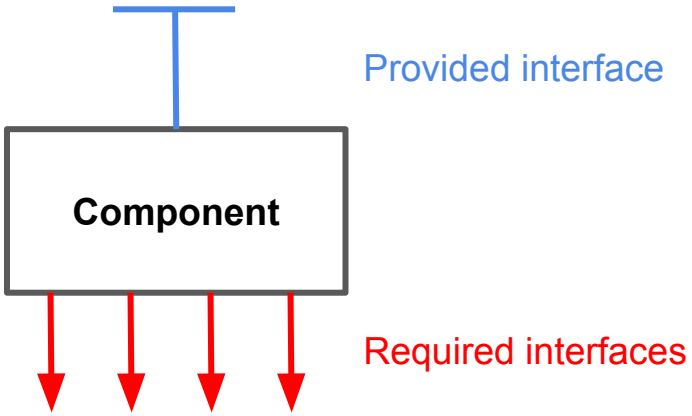
- Component-based model:



Introducing the concept

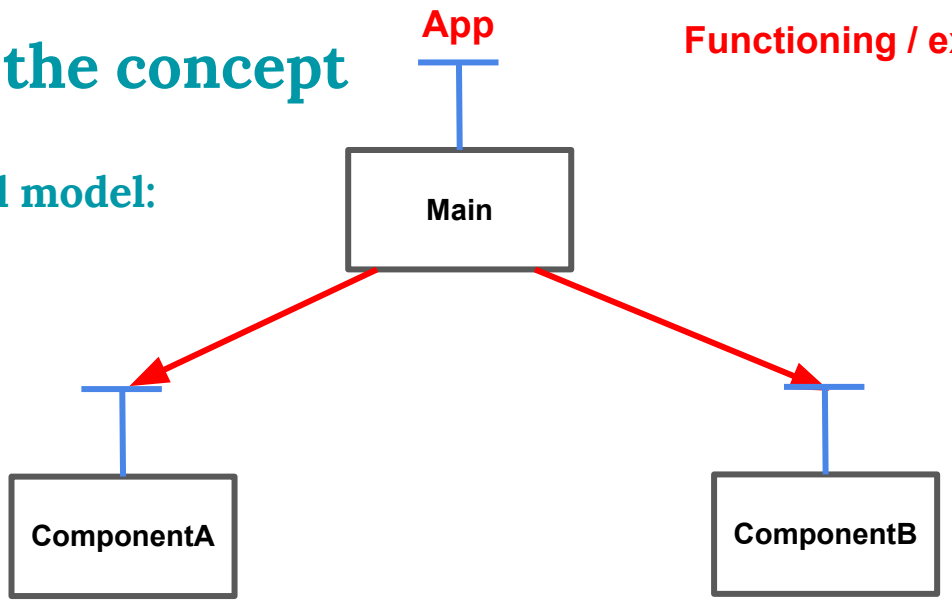
Component-based model:

- **Interface** is a collection of function prototypes;
- **Component** is strongly encapsulated;
- **Composition** happens through a composer separate from the logic of the system;



Emergent Software Systems

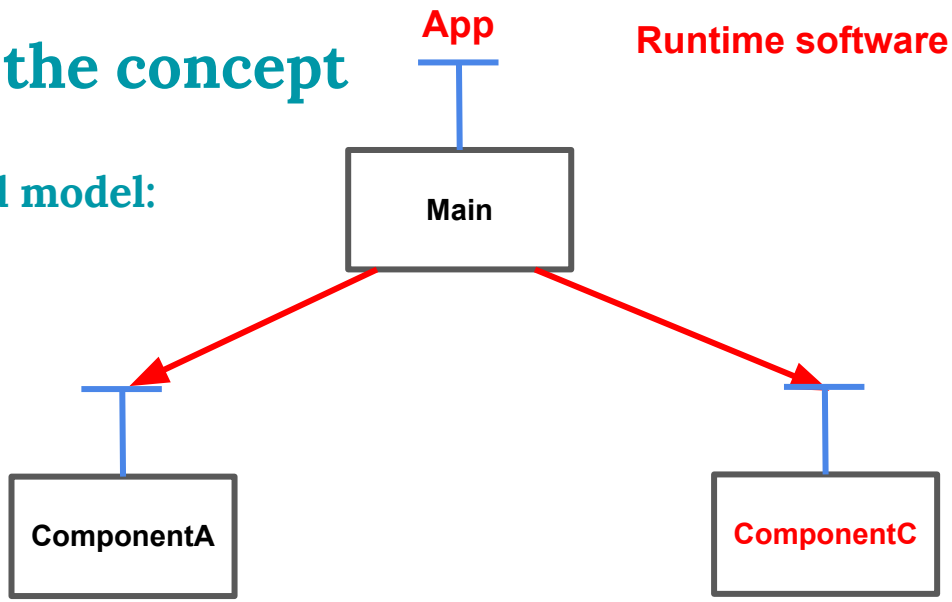
- Introducing the concept
  - Component-based model:



Functioning / executing system

Emergent Software Systems

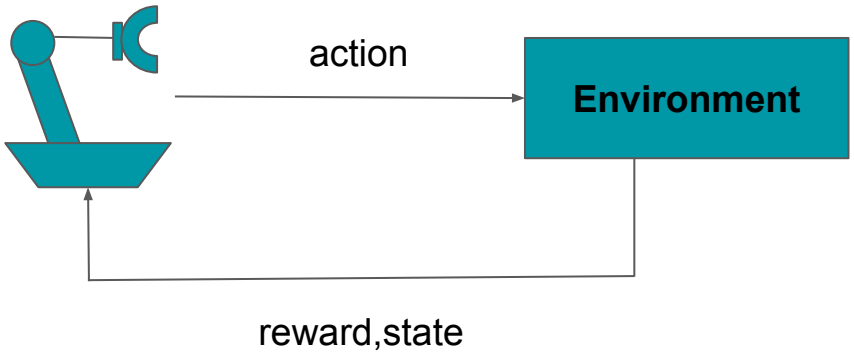
- Introducing the concept
  - Component-based model:



Runtime software adaptation / evolution.

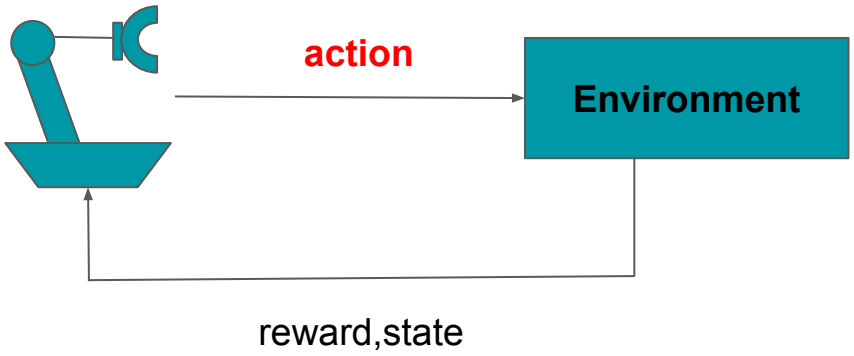
Introducing the concept

- Reinforcement learning:



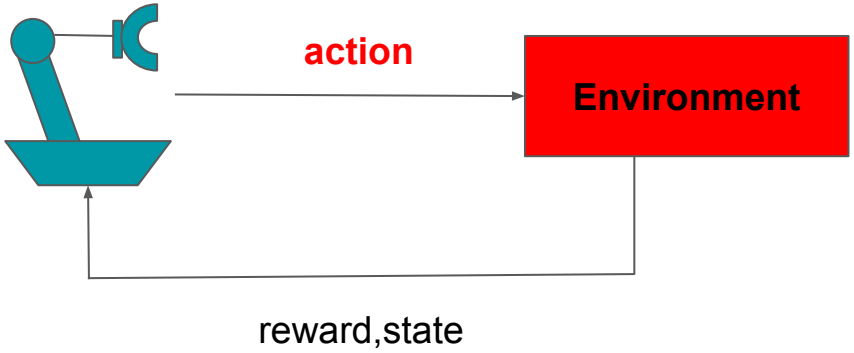
Introducing the concept

- Reinforcement learning:



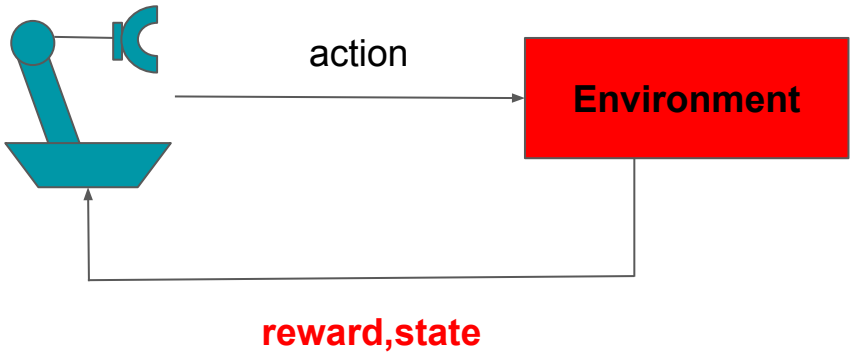
Introducing the concept

- Reinforcement learning:



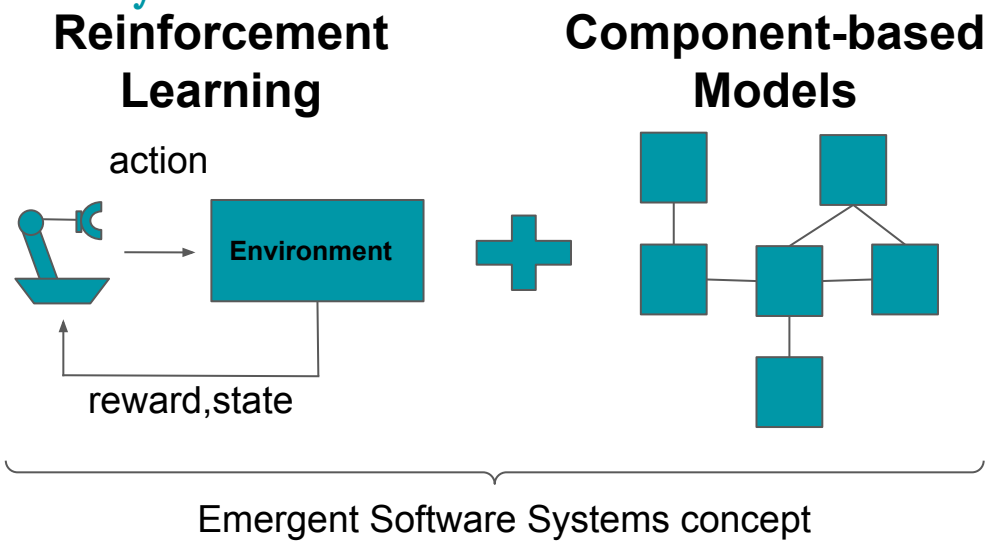
Introducing the concept

- Reinforcement learning:



Introducing the concept

- Emergent Software Systems:



INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

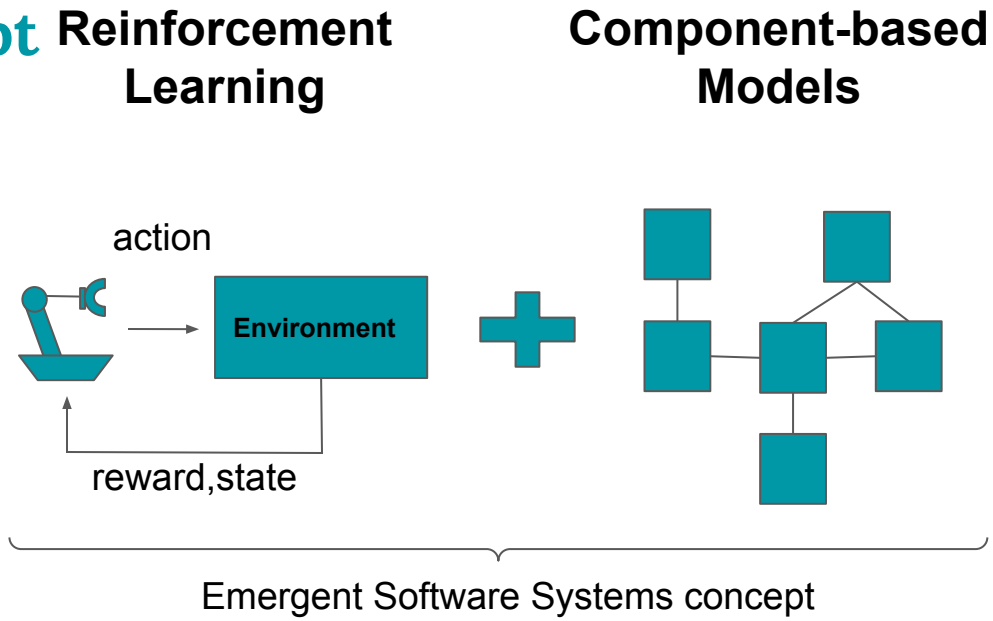
→ Introducing the concept

→ Emergent Software Systems requires:

- A set of component variants (multiple components implementing the same interface);
- A set of metrics are extracted from the live system to monitor systems health and classify its operating environment;
- Zero-training reinforcement learning algorithm to find the most suitable architectural composition for the identified operating environment;

Emergent Software Systems

- Introducing the concept
  - Emergent Software Systems:



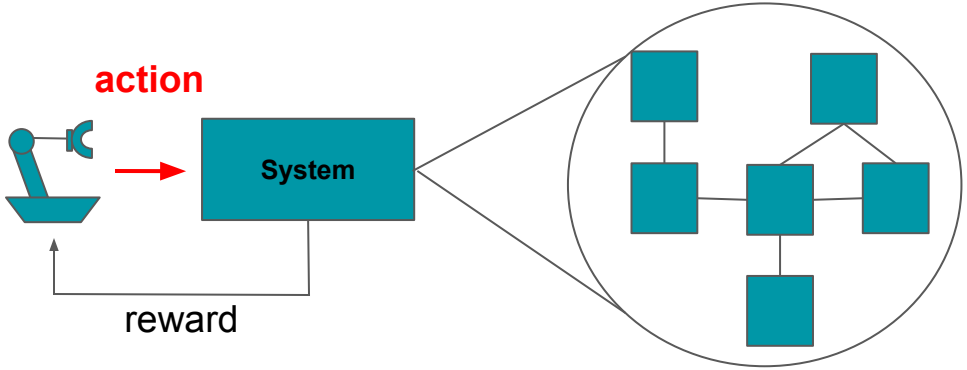
Emergent Software Systems

- Introducing the concept

Reinforcement Learning

Component-based Models

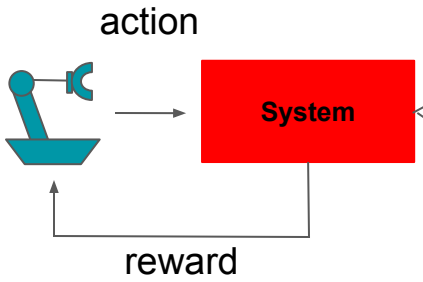
- Emergent Software Systems:



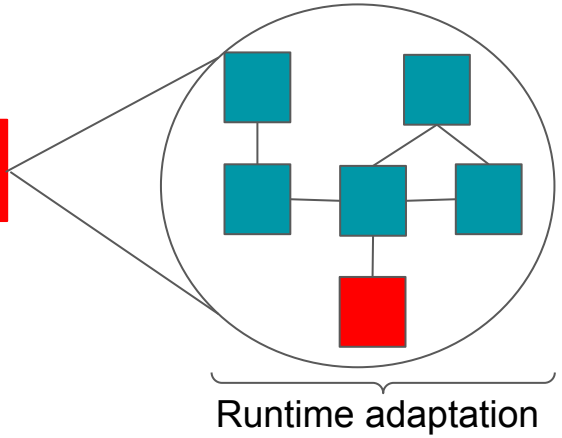
Emergent Software Systems

- Introducing the concept
  - Emergent Software Systems:

Reinforcement Learning



Component-based Models

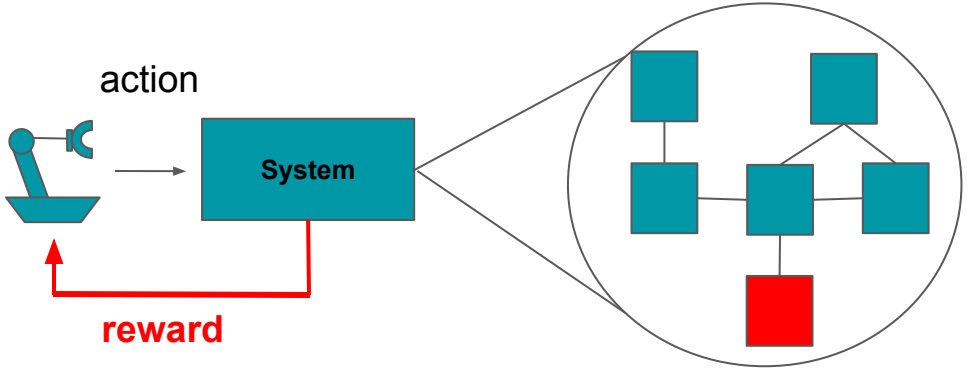


Emergent Software Systems

- Introducing the concept
  - Emergent Software Systems:

Reinforcement Learning

Component-based Models

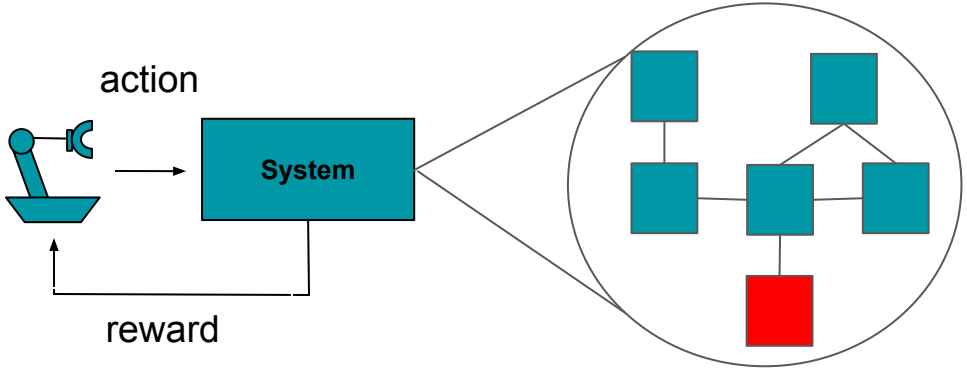


Emergent Software Systems

- Introducing the concept
  - Emergent Software Systems:

Reinforcement Learning

Component-based Models



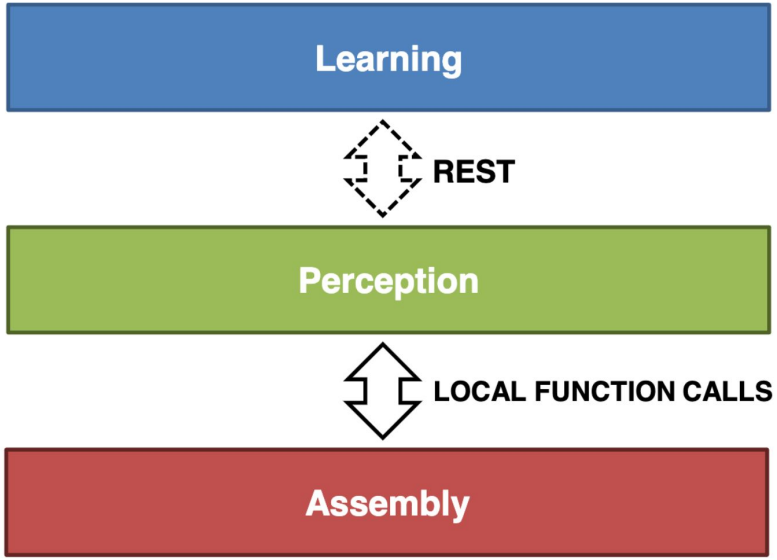
INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

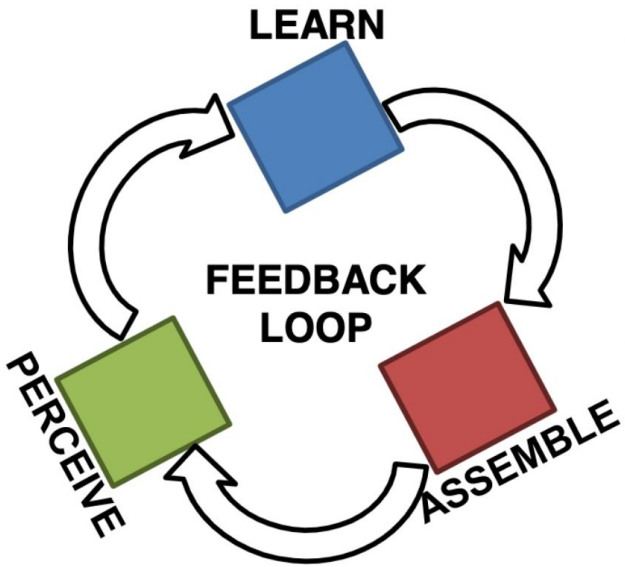
➔ Implementation (PAL framework)

➔ Overview:

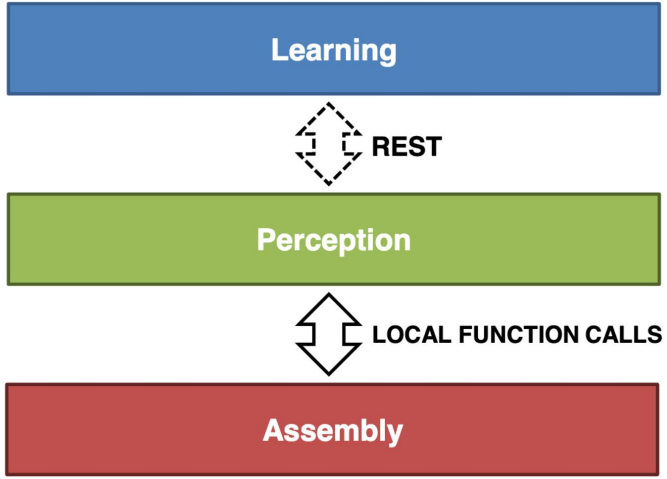
- PAL stands for Perception, Assembly, Learning;
- Dana programming language (component-based model);
- Learning algorithm: Baseline and Multi-armed Bandits;

→ Implementation (PAL framework)

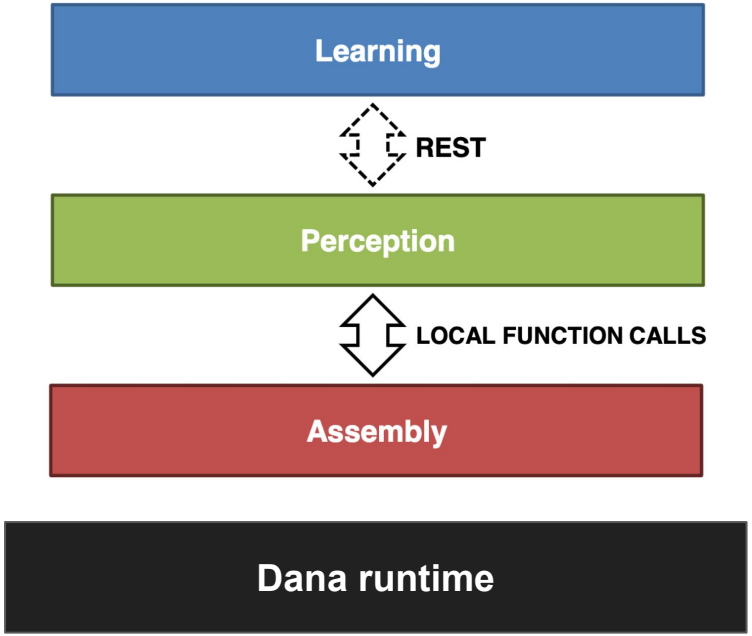




→ Implementation (PAL framework)

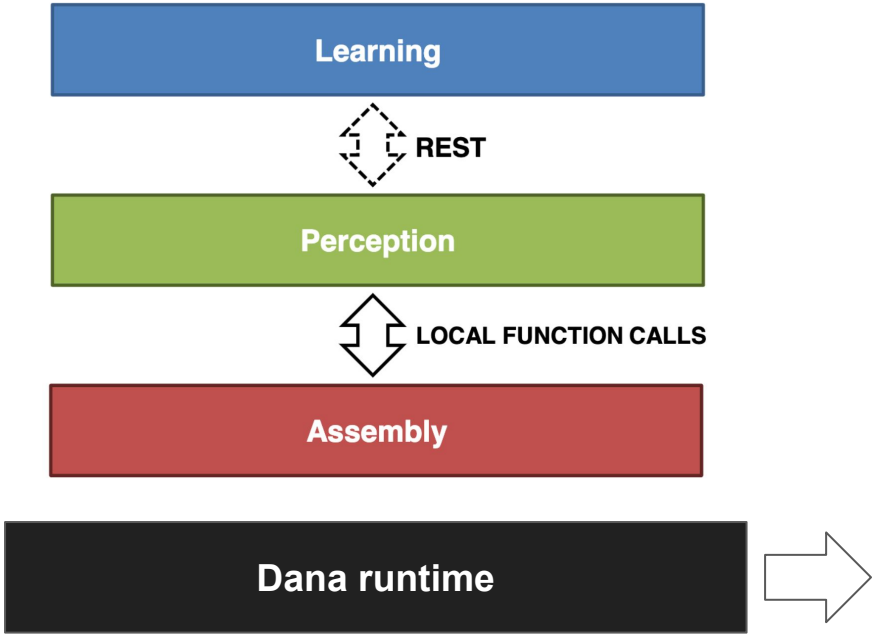


→ Implementation (PAL framework)



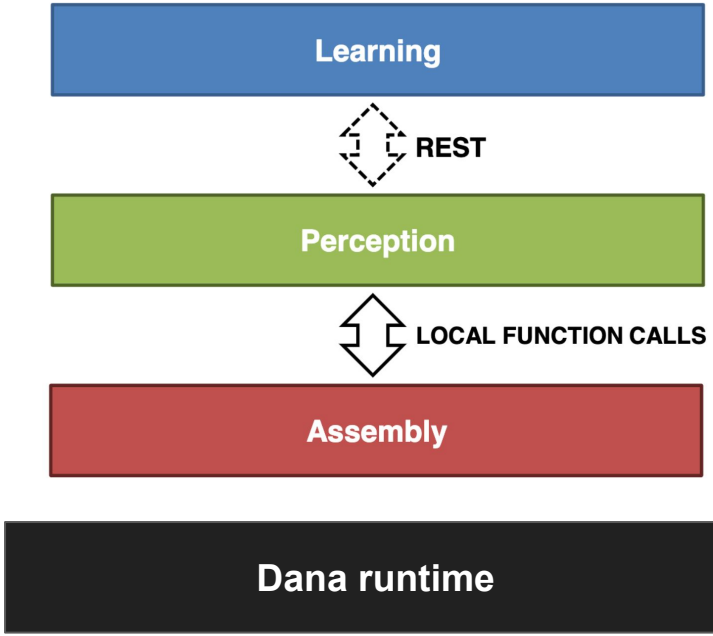
robertovito@ufg.br

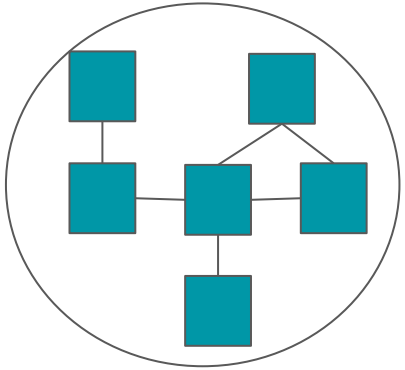
→ Implementation (PAL framework)



Seamless runtime adaptation

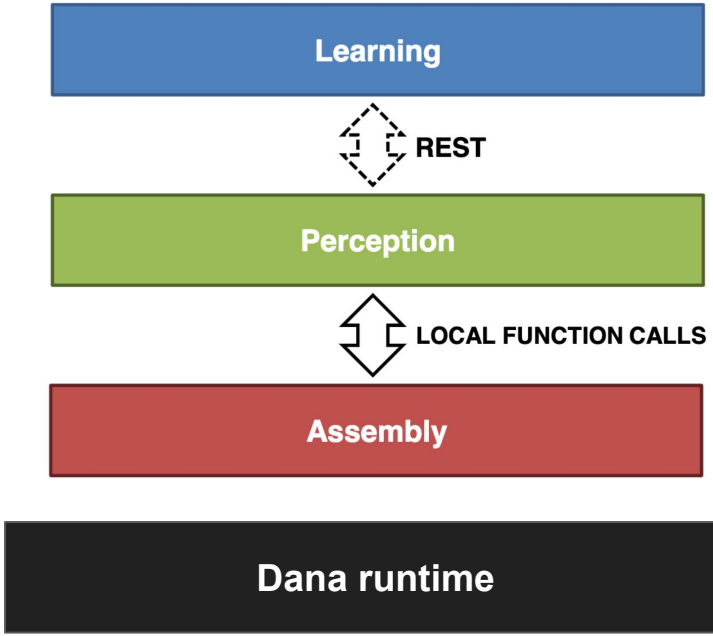
→ Implementation (PAL framework)

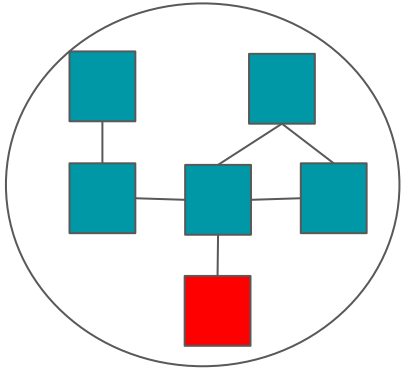




Seamless runtime adaptation

→ Implementation (PAL framework)

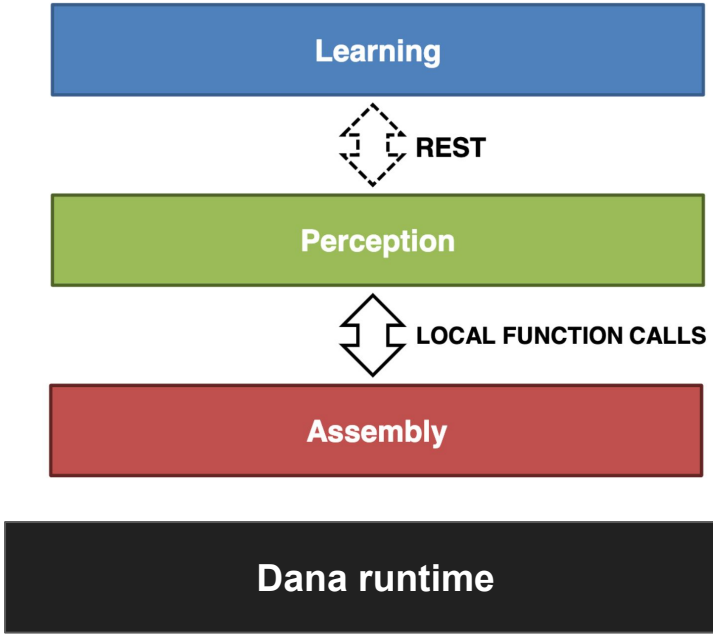




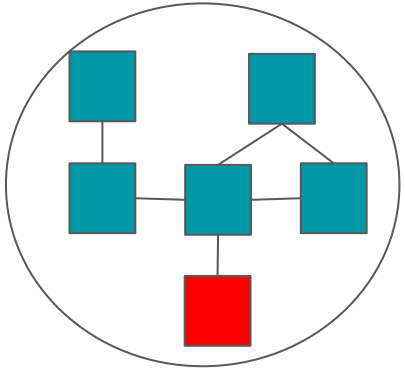
Seamless runtime adaptation

Emergent Software Systems

→ Implementation (PAL framework)



Seamless runtime adaptation

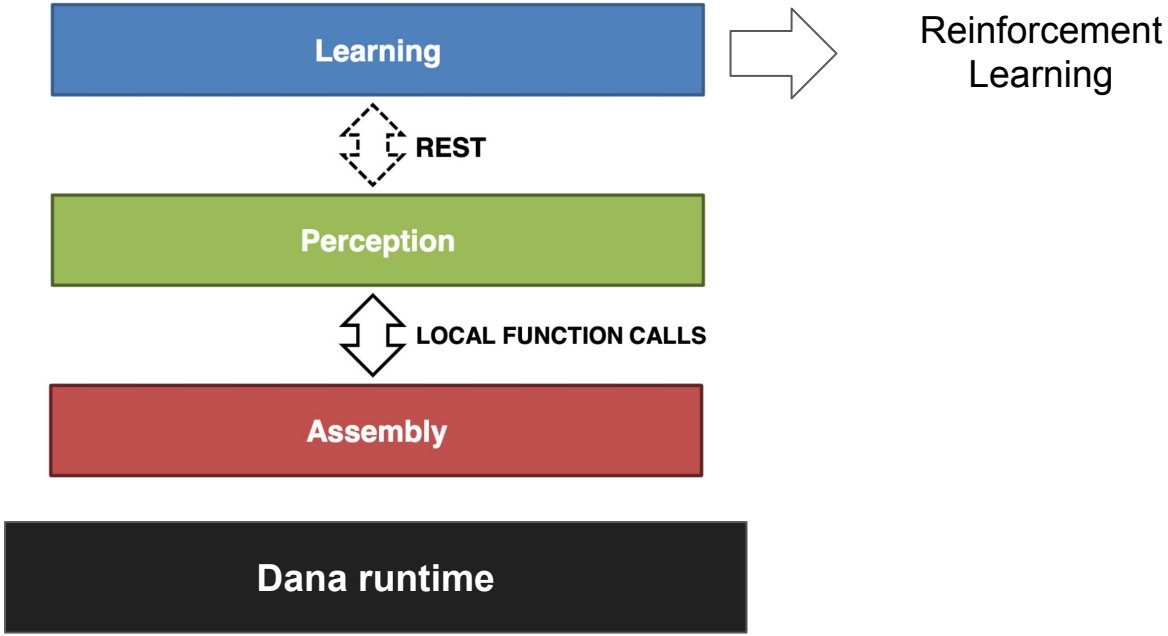


Porter, B., Rodrigues Filho, R., 2021. **"A programming language for sound self-adaptive systems"**, *In: 2021 IEEE ACSOS*

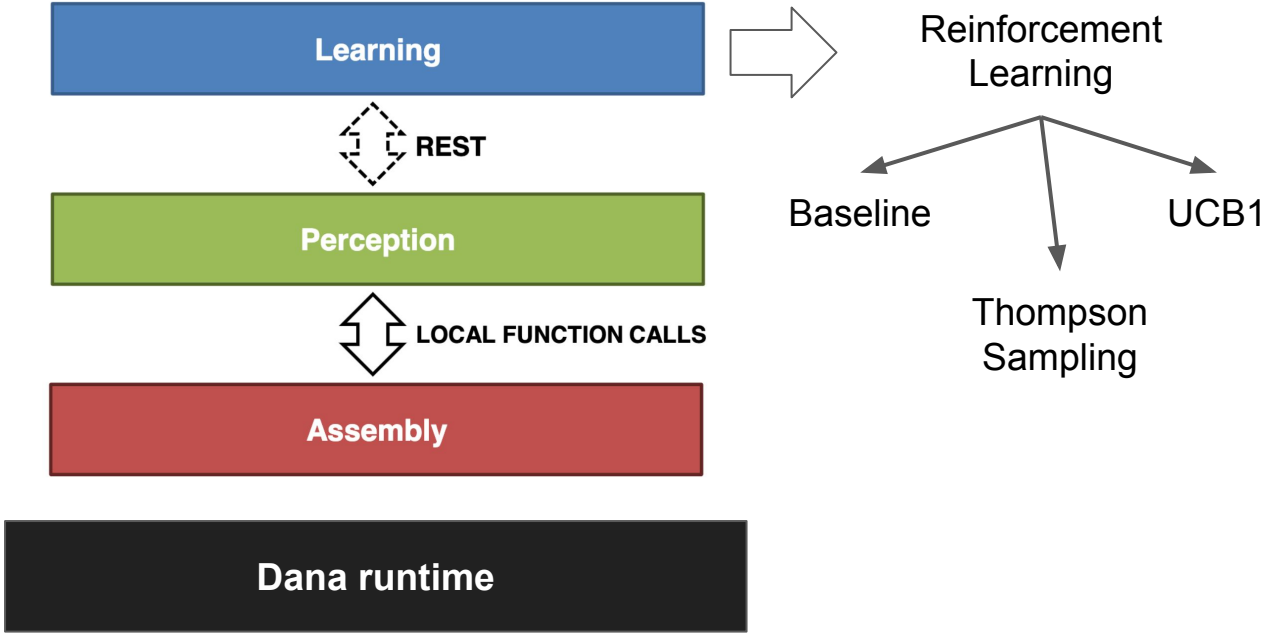
Porter, B., 2014. **"Runtime modularity in complex structures: A component model for fine grained runtime adaptation"**, *In: 2014 ACM CBSE*

robertovito@ufg.br

→ Implementation (PAL framework)

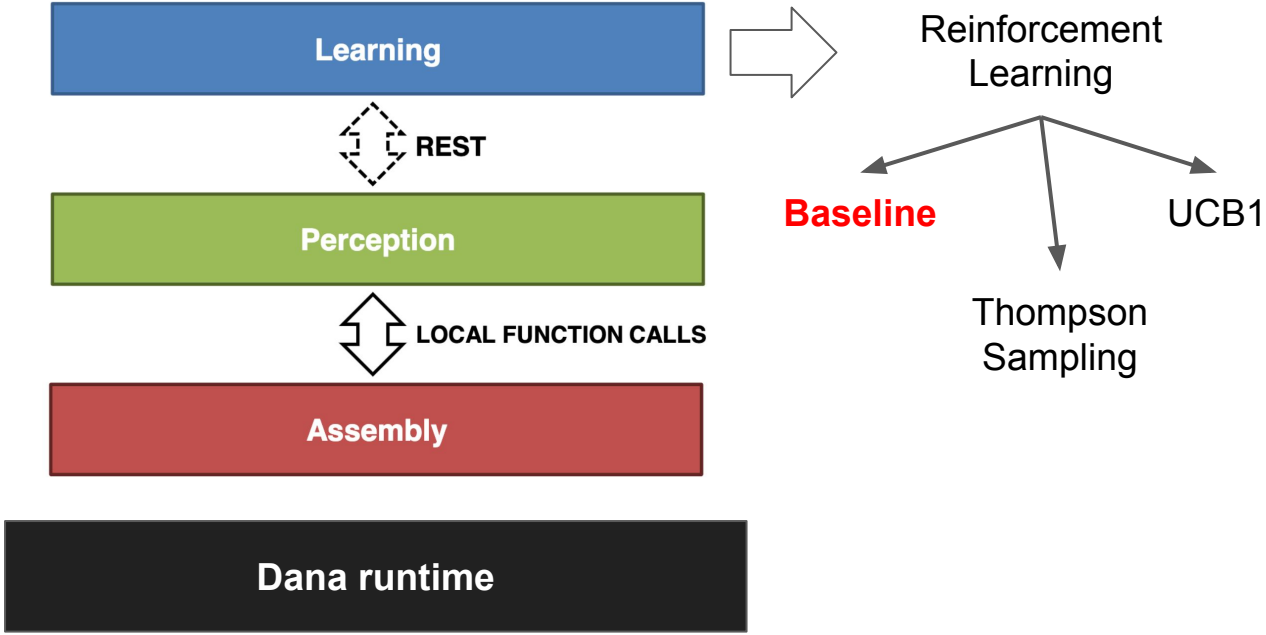


→ Implementation (PAL framework)



robertovito@ufg.br

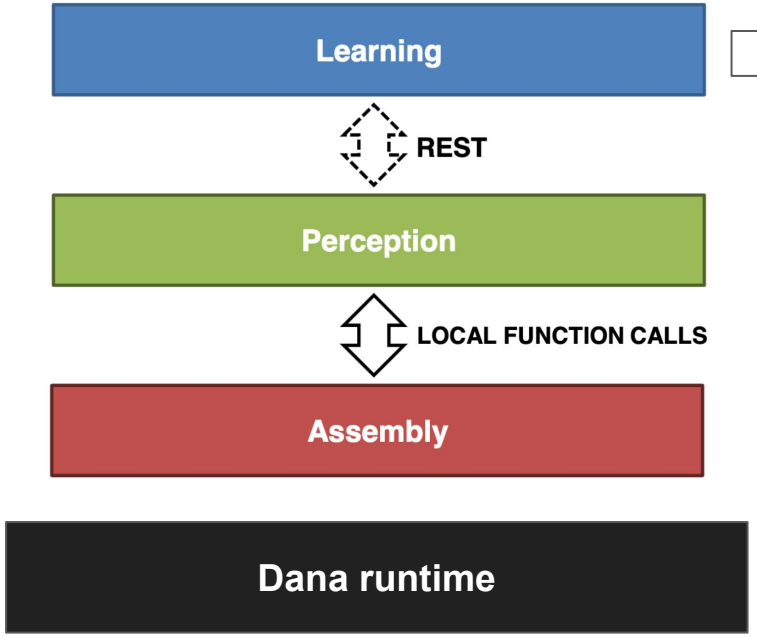
→ Implementation (PAL framework)

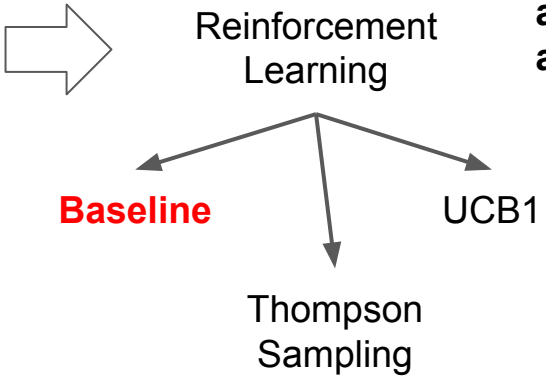


robertovito@ufg.br

Emergent Software Systems

→ Implementation (PAL framework)





Porter, B., Rodrigues Filho, R. 2016 **"Losing control: The case for emergent software systems using autonomous assembly, perception, and learning."** In: 2016 IEEE SASO

Porter, B., Grieves, M., Rodrigues Filho, R., and Leslie, D. 2016. **"REX: A development platform and online learning approach for runtime emergent software systems."** In: 2016 USENIX OSDI

Porter, B., Rodrigues Filho, R. **"Distributed emergent software: Assembling, perceiving and learning systems at scale."** In: 2019 IEEE SASO

robertovito@ufg.br

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Assembly

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Assembly

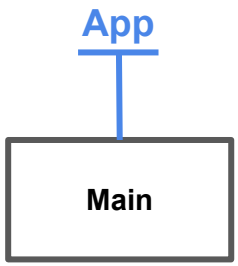
Assembly API

- void setMain(char mainComp[])
- bool setConfig(char config[])
- char[] getConfig()

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Assembly

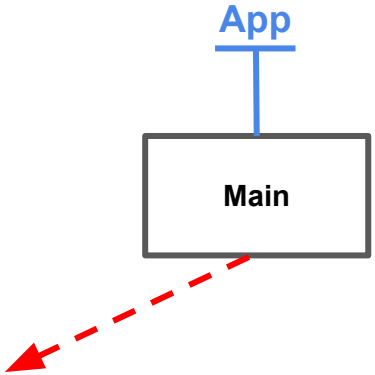


Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()

Data structure

→ Implementation (PAL framework)



Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()

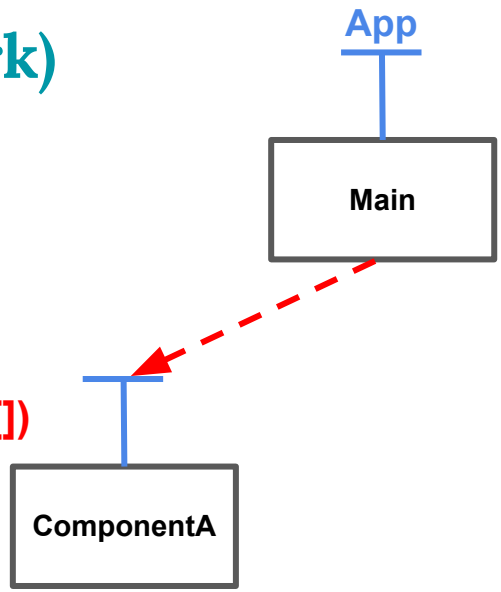
Data structure

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



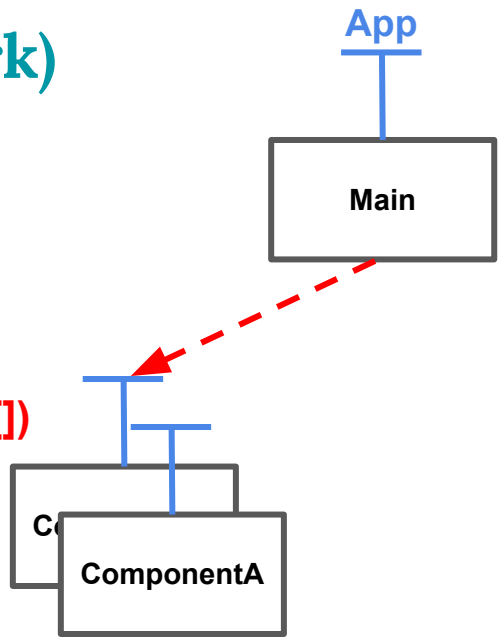
Data structure

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



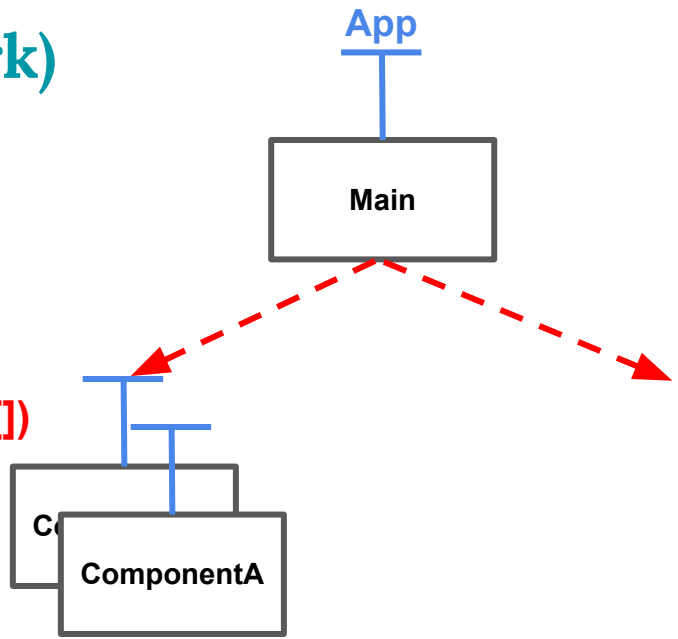
Data structure

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



Data structure

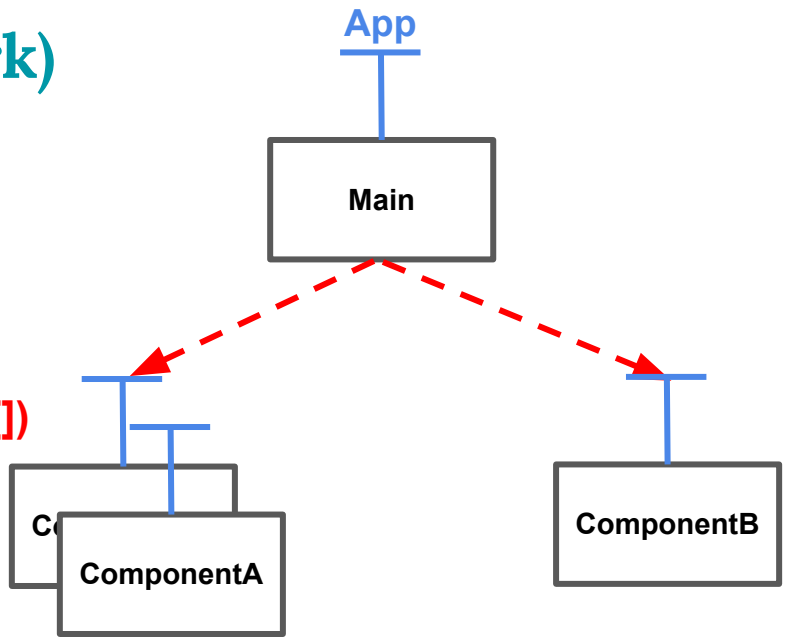
Emergent Software Systems

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



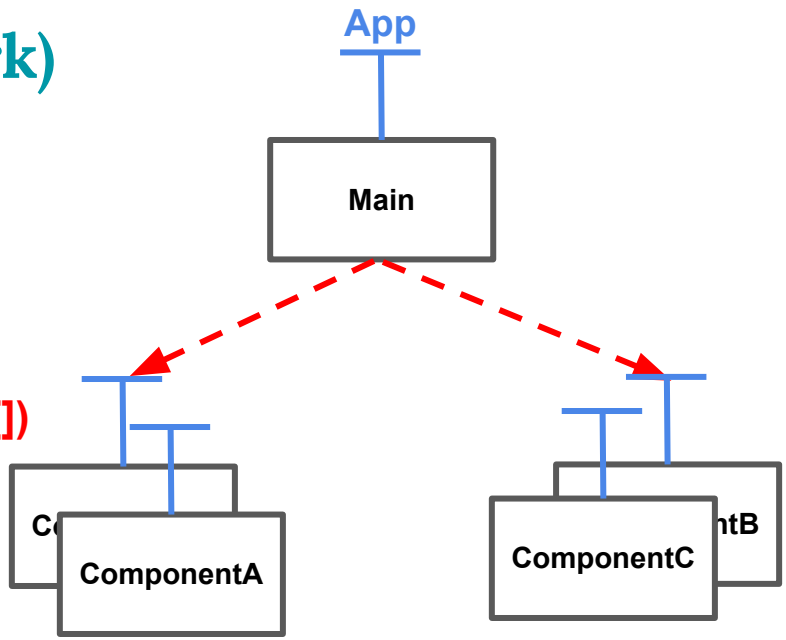
Data structure

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



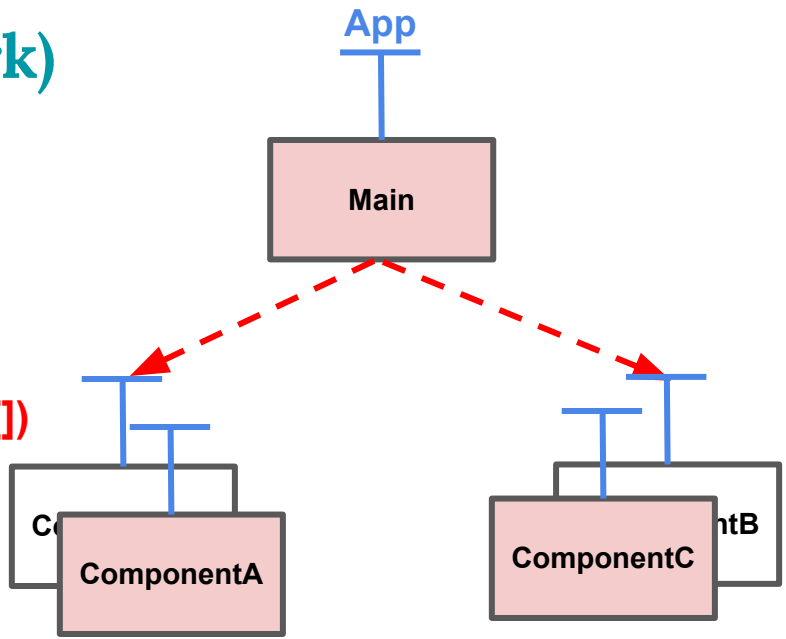
Data structure

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



Loading components to memory

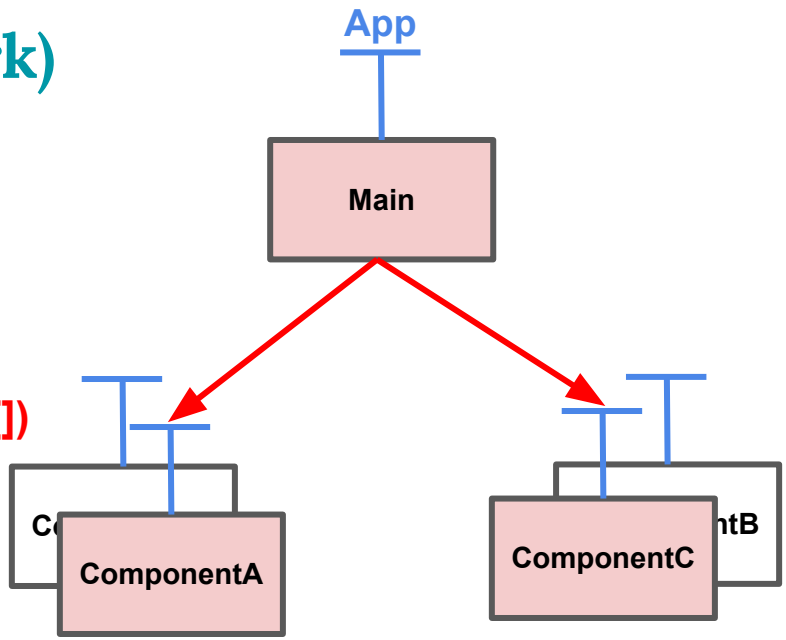
Emergent Software Systems

→ Implementation (PAL framework)

Assembly

Assembly API

- **void setMain(char mainComp[])**
- bool setConfig(char config[])
- char[] getConfig()



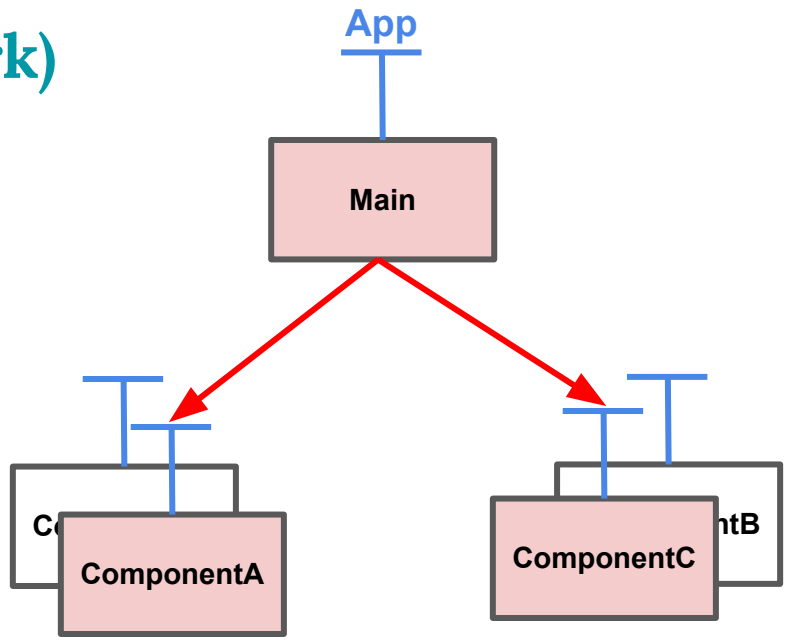
Executing system

→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- **bool setConfig(char config[])**
- char[] getConfig()



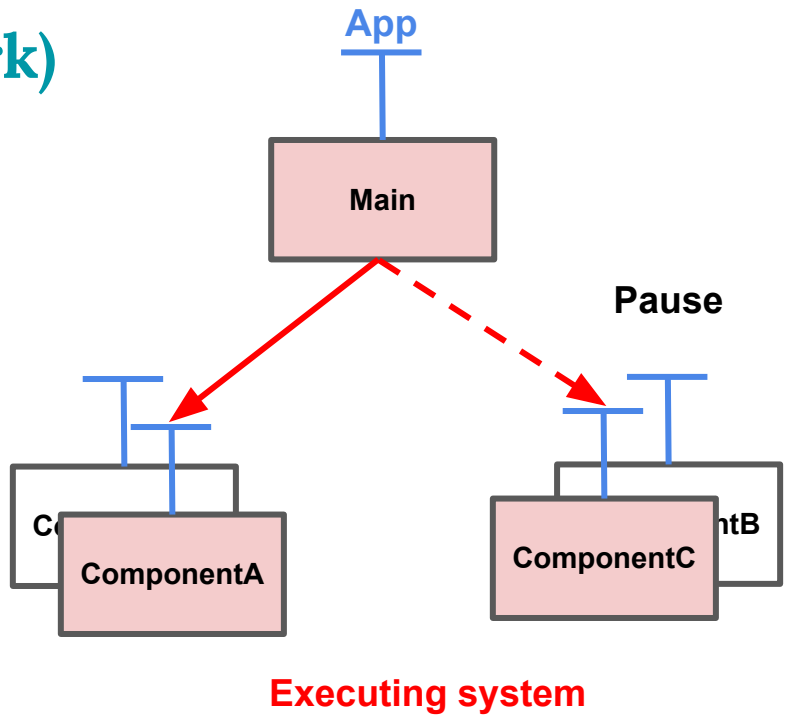
Executing system

→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- **bool setConfig(char config[])**
- char[] getConfig()

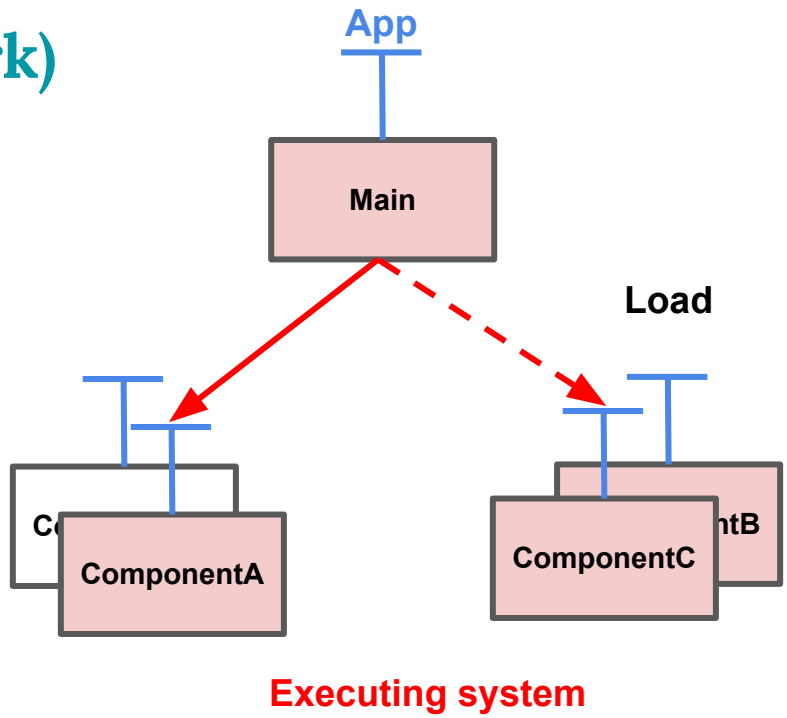


→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- **bool setConfig(char config[])**
- char[] getConfig()

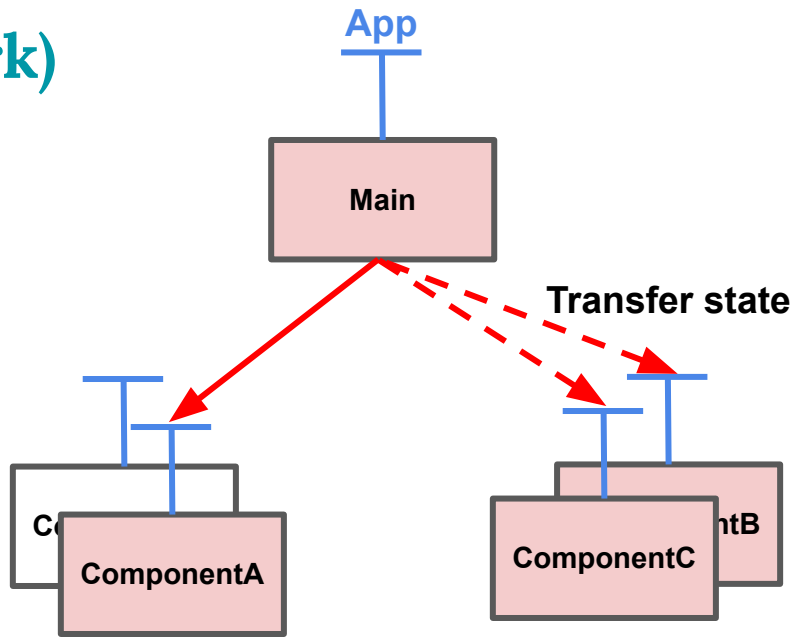


→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- **bool setConfig(char config[])**
- char[] getConfig()

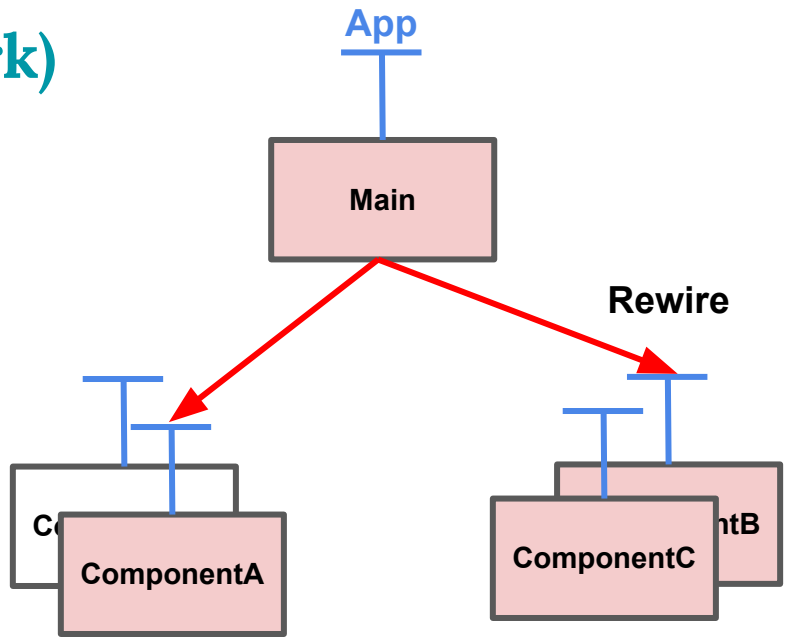


→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- **bool setConfig(char config[])**
- char[] getConfig()

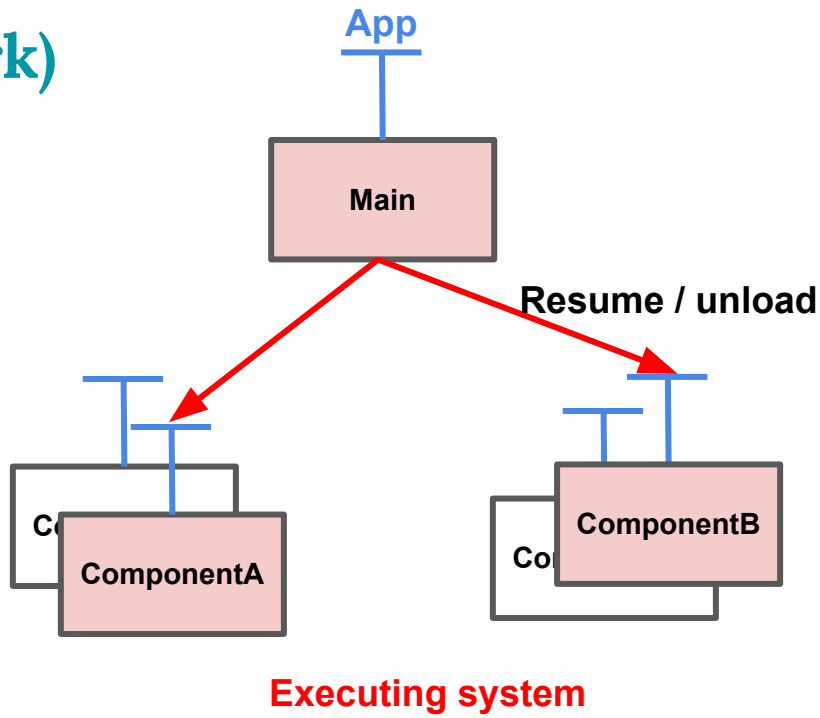


→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- **bool setConfig(char config[])**
- char[] getConfig()

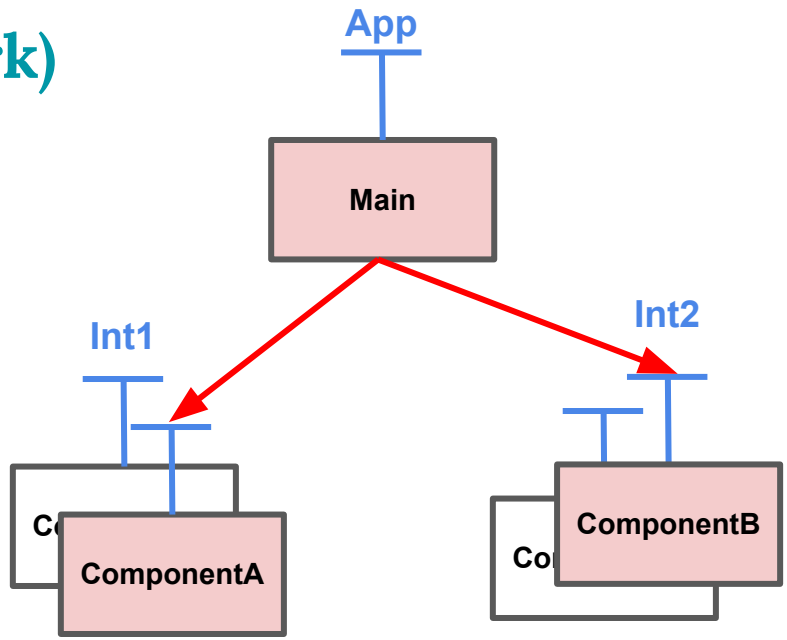


→ Implementation (PAL framework)

Assembly

Assembly API

- void setMain(char mainComp[])
- bool setConfig(char config[])
- **char[] getConfig()**



Executing system

→ Implementation (PAL framework)

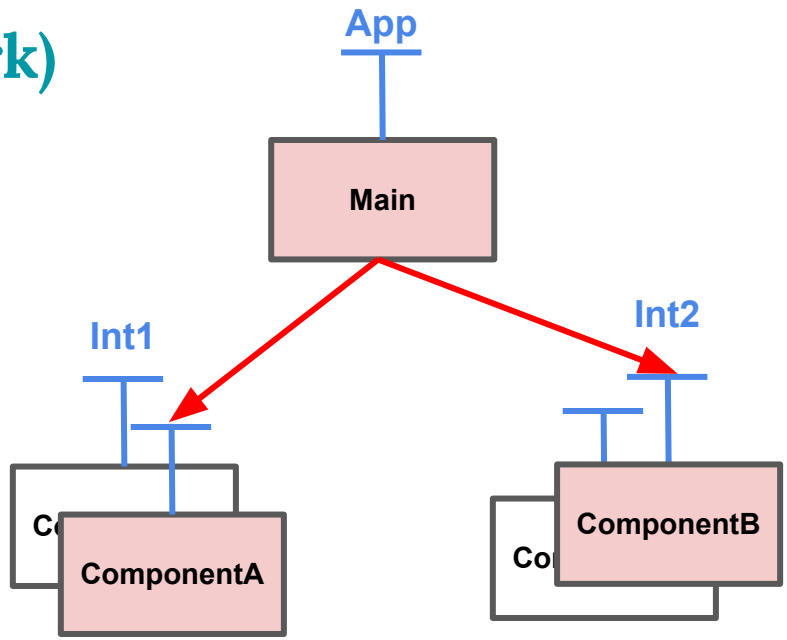
Assembly

Assembly API

- void setMain(char mainComp[])
- bool setConfig(char config[])
- **char[] getConfig()**

Architectural Description

|Main, ComponentA, ComponentB|0:Int1:1,0:Int2:2|



Executing system

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Perception

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Perception

Perception API

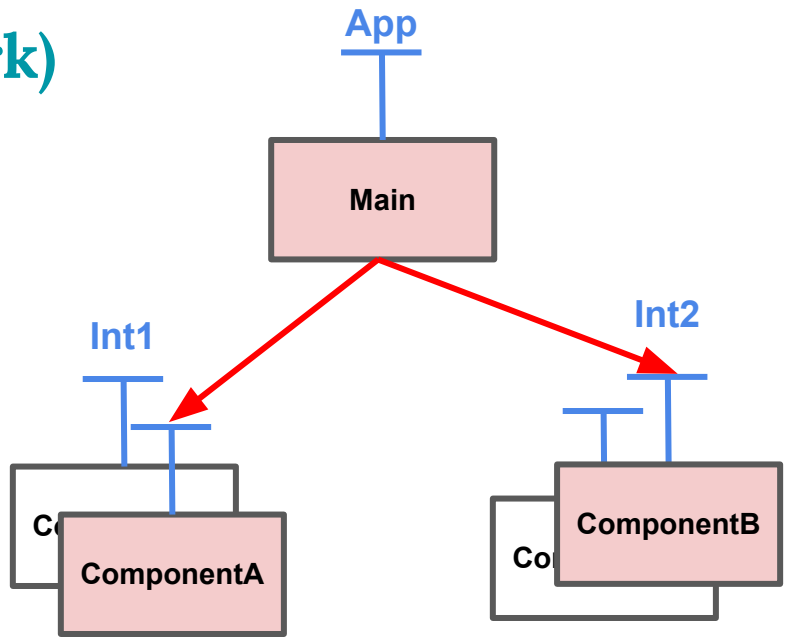
- void addProxy(char exp[])
- char[] getPerceptionData()

→ Implementation (PAL framework)

Perception

Perception API

- **void addProxy(char exp[])**
- char[] getPerceptionData()



Executing system

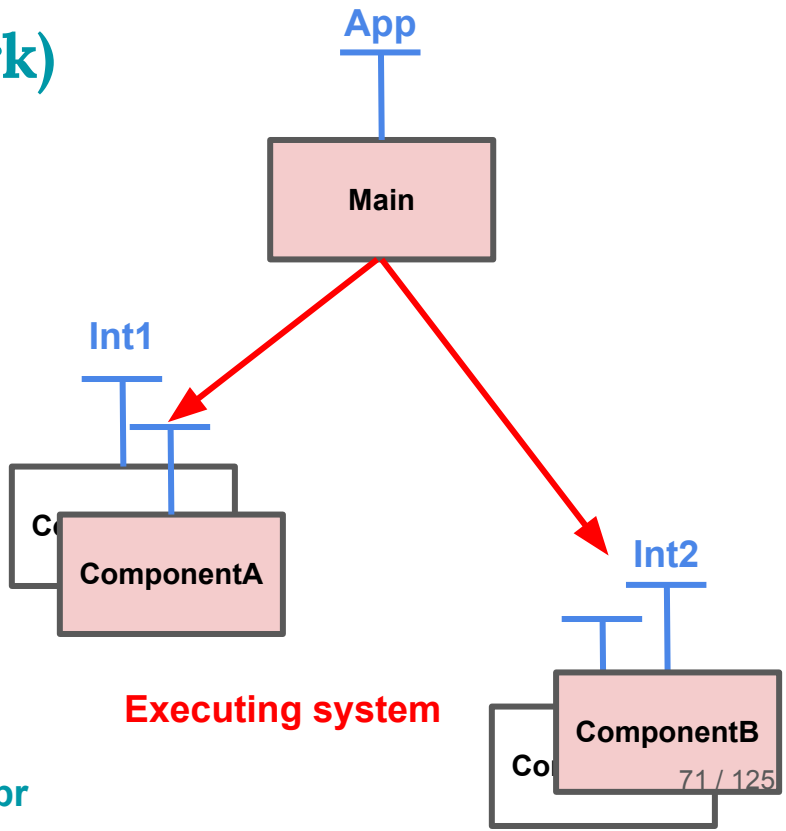
INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Perception

Perception API

- **void addProxy(char exp[])**
- char[] getPerceptionData()

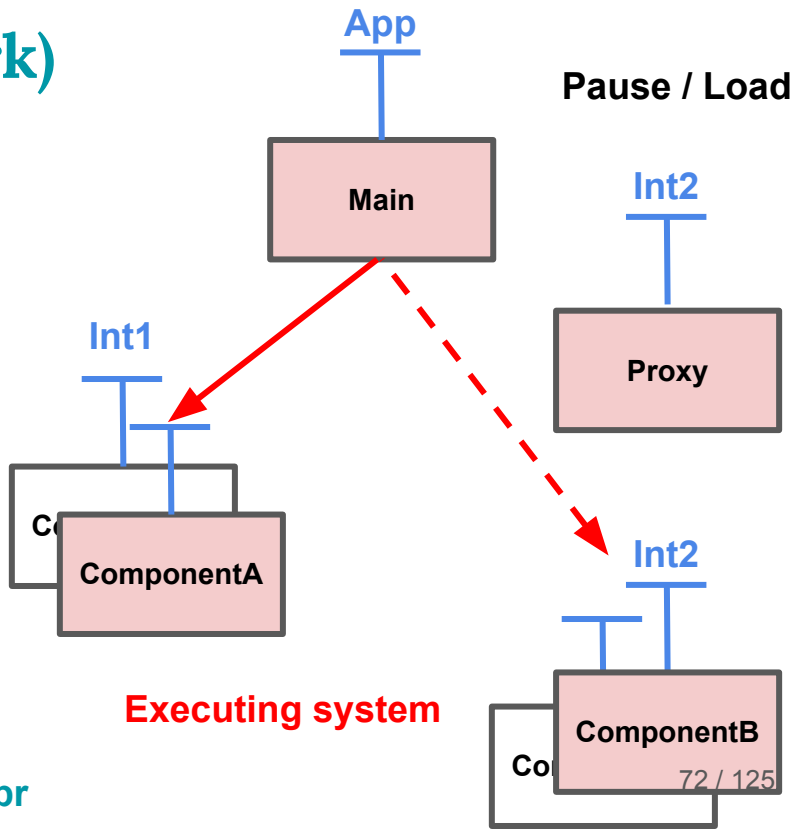


→ Implementation (PAL framework)

Perception

Perception API

- void addProxy(char exp[])
- char[] getPerceptionData()

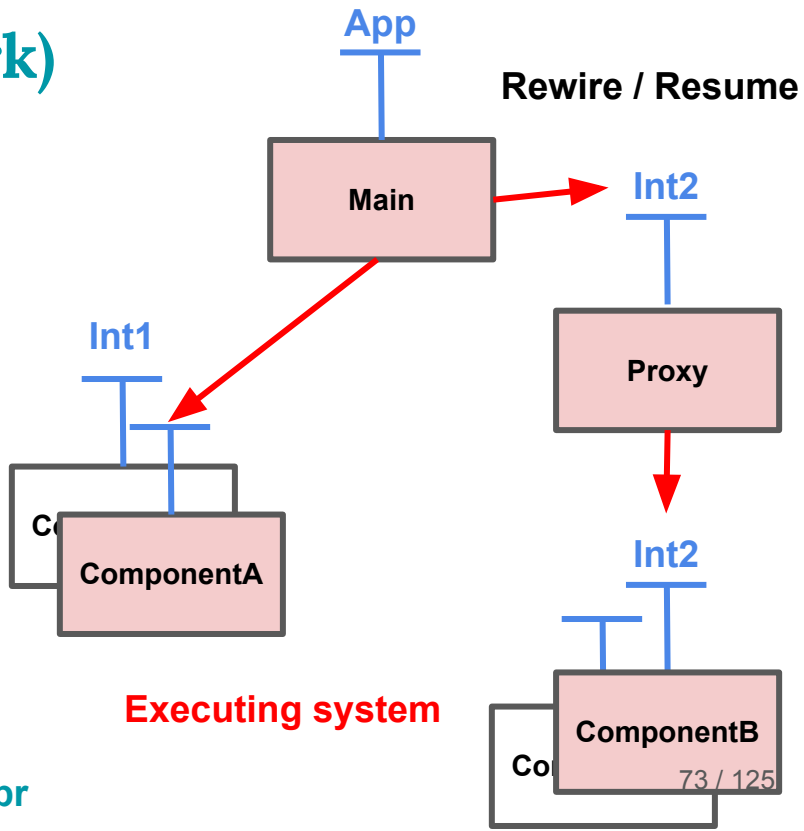


→ Implementation (PAL framework)

Perception

Perception API

- **void addProxy(char exp[])**
- char[] getPerceptionData()



Emergent Software Systems

→ Implementation (PAL framework)

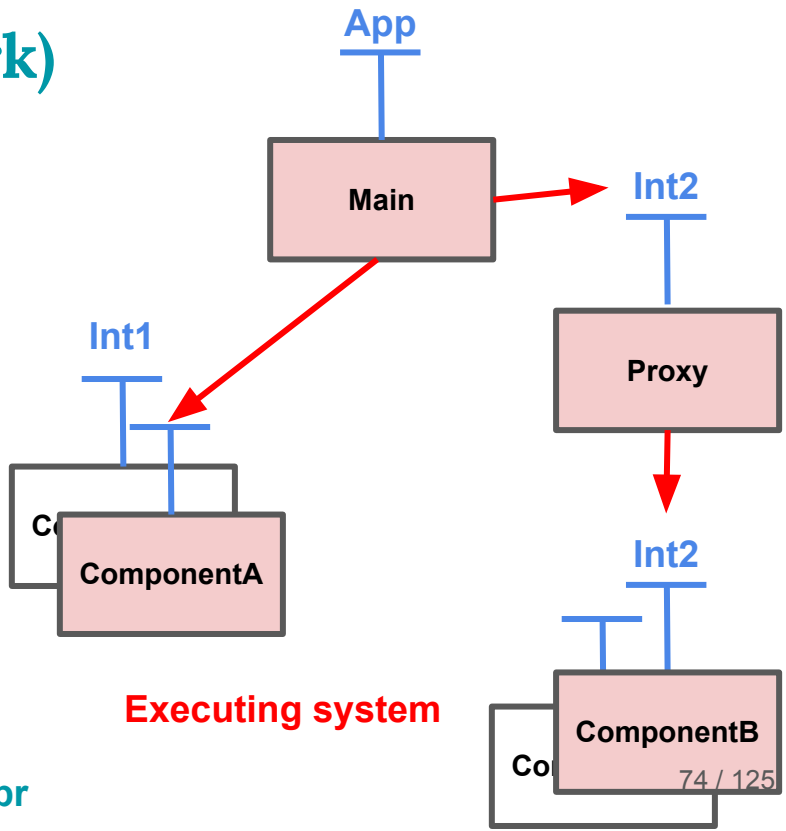
Perception

Perception API

- void addProxy(char exp[])
- **char[] getPerceptionData()**

```
{"metric" :
{ "name": "response_time",
  "value": 300,
  "count": 10 } }
```

```
{"event" :
{ "name": "mime_type",
  "value": "image",
  "count": 10 } }
```



Executing system

robertovito@ufg.br

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

Learning

→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- Exploration
- Exploitation

→ Implementation (PAL framework)

Learning

Baseline algo:

- **Setup**
- Exploration
- Exploitation

Consists of executing:

- setMain();
- addProxy("exp");
- Set up the learning algorithm;

→ Implementation (PAL framework)

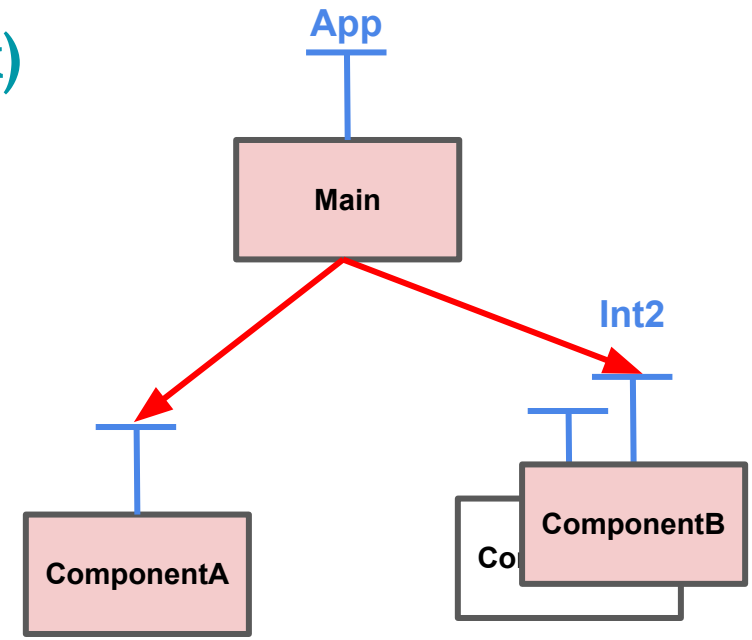
Learning

Baseline algo:

- **Setup**
- Exploration
- Exploitation

Consists of executing:

- **setMain();**
- addProxy("exp");
- Set up the learning algorithm;



Executing system

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

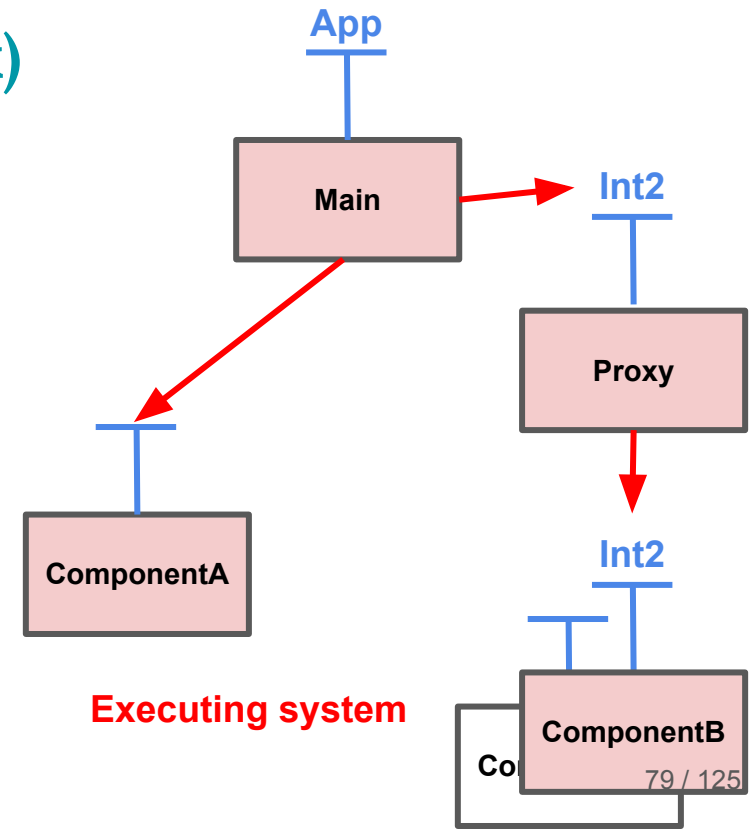
Learning

Baseline algo:

- **Setup**
- Exploration
- Exploitation

Consists of executing:

- setMain();
- **addProxy("exp");**
- Set up the learning algorithm;



Executing system

robertovito@ufg.br

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

→ Implementation (PAL framework)

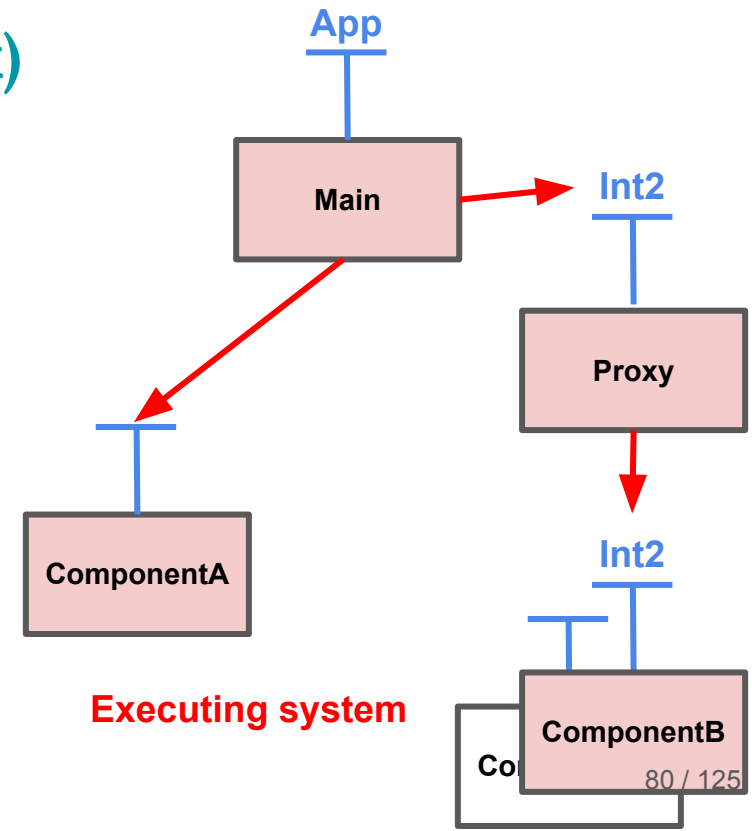
Learning

Baseline algo:

- **Setup**
- Exploration
- Exploitation

Consists of executing:

- setMain();
- addProxy("exp");
- **Set up the learning algorithm;**
  - **configs[] = getAllConfigs();**



robertovito@ufg.br

→ Implementation (PAL framework)

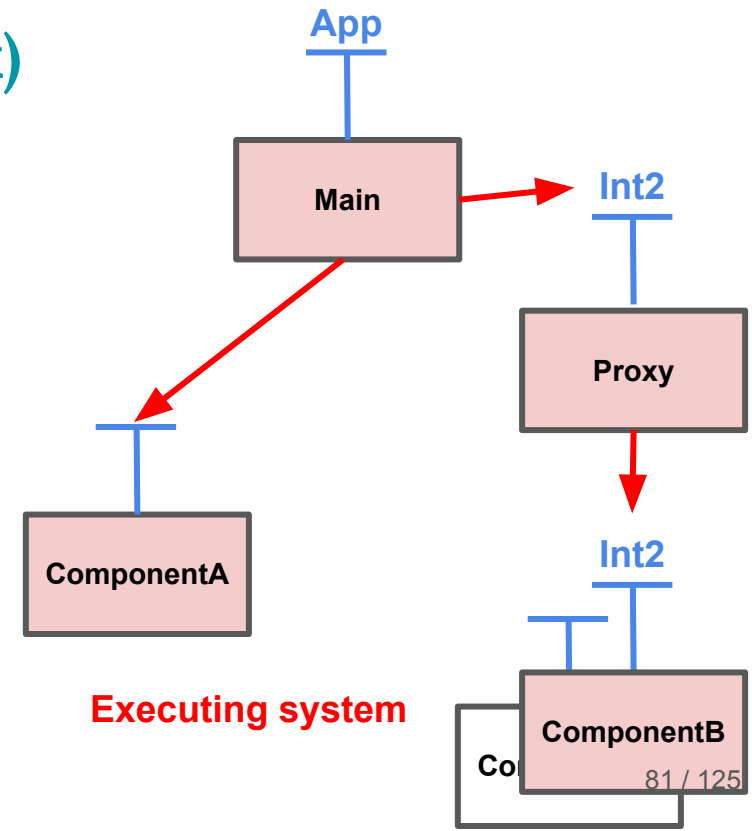
Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Consists of:

- For all compositions
  - Observe;
  - Collect perception data;
- Compare all metrics to find the best;



Executing system

robertovito@ufg.br

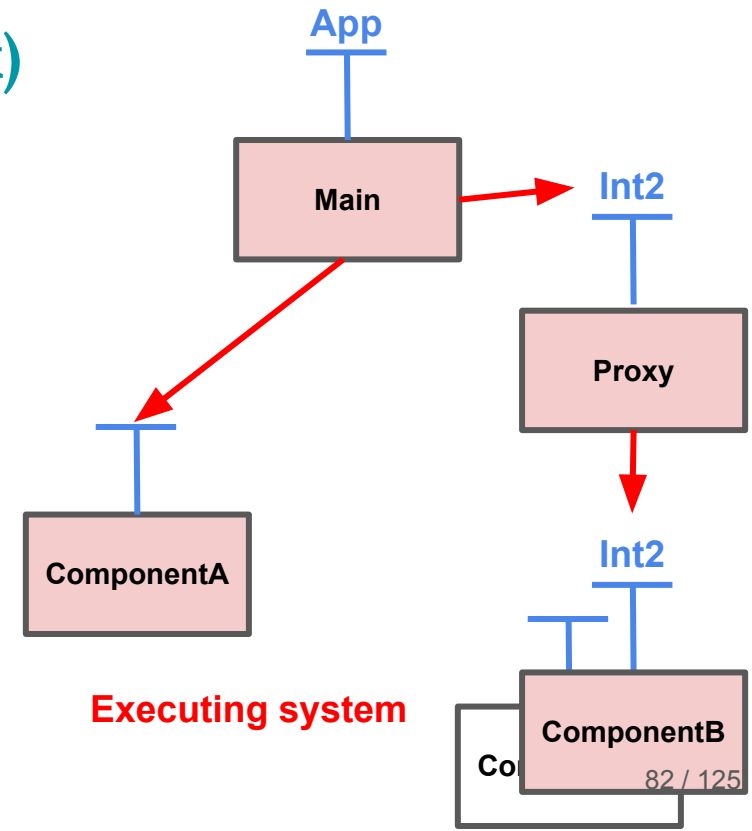
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Observing **Data:**



Executing system

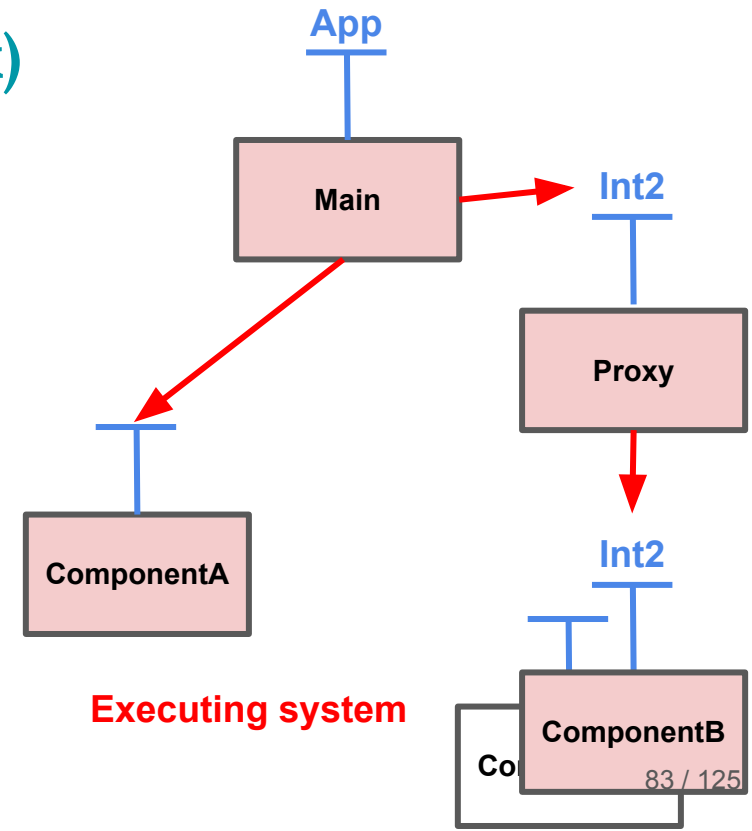
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Get perception data **Data:**



Executing system

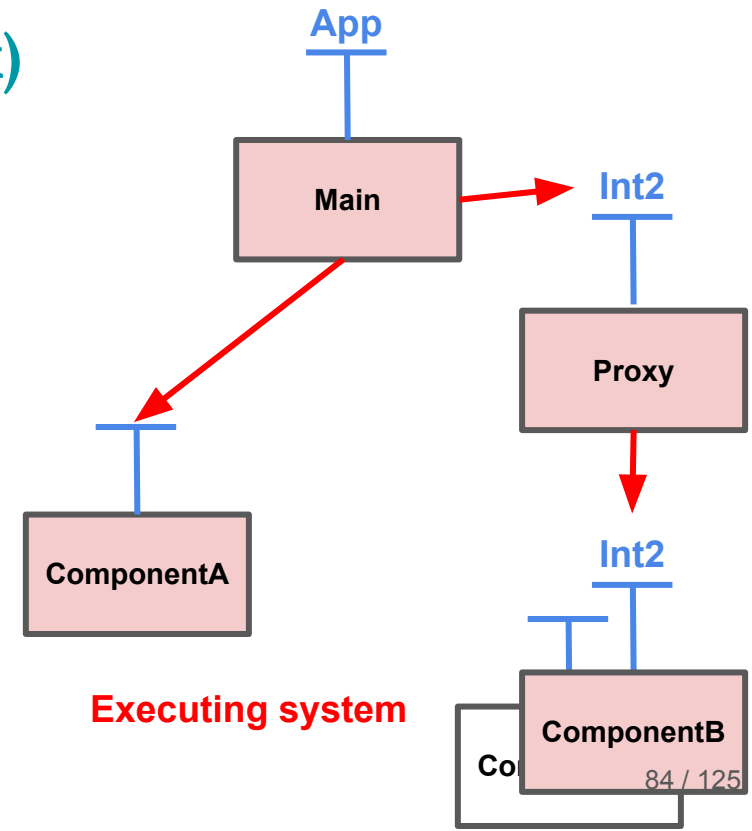
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Get perception data **Data:** (arch1: resp_time 20ms, text)



Executing system

robertovito@ufg.br

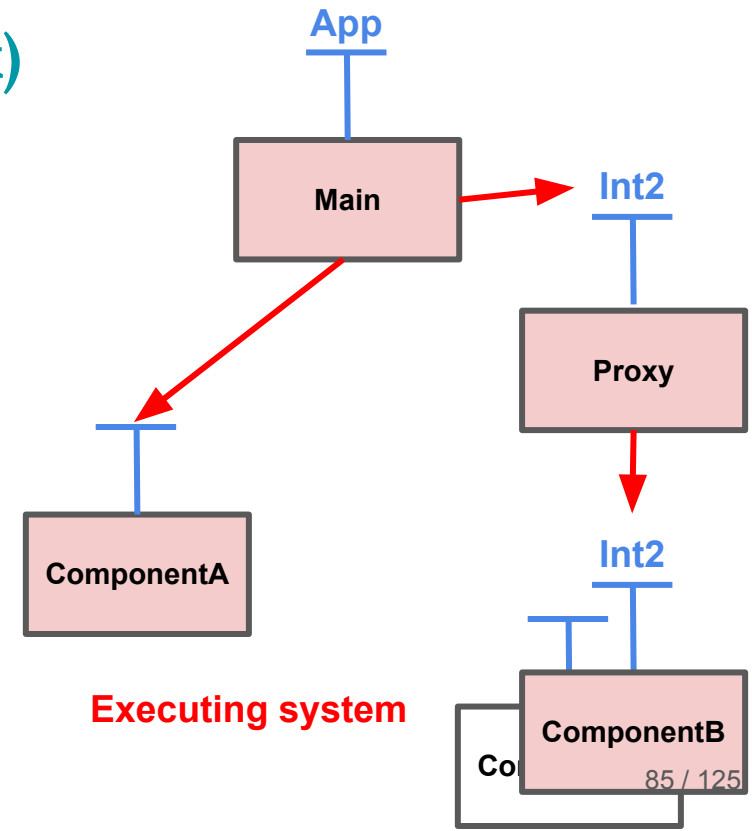
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: New action **Data:** (arch1: resp_time 20ms, text)



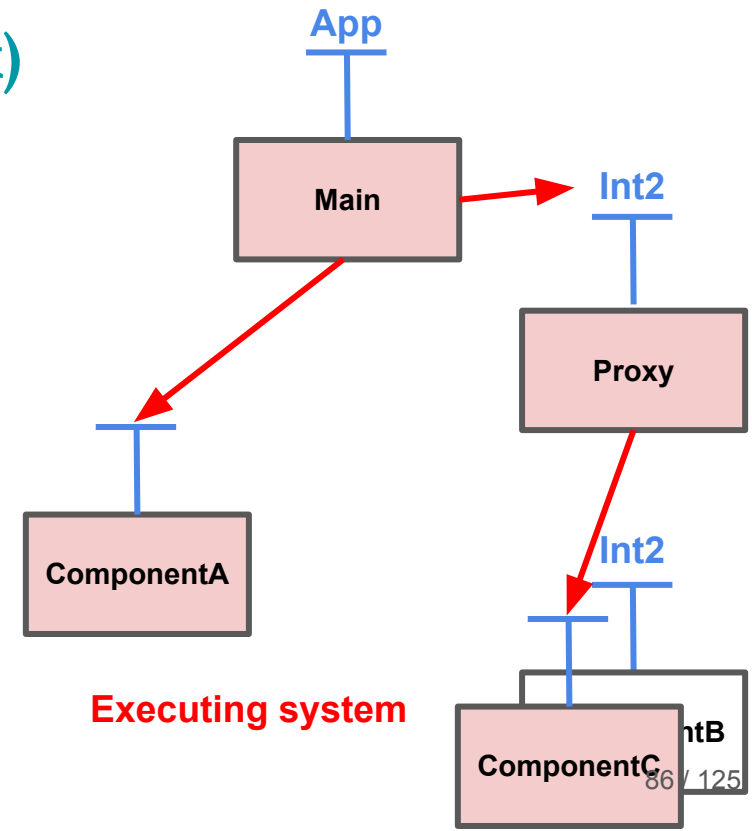
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: New action **Data:** (arch1: resp_time 20ms, text)



robertovito@ufg.br

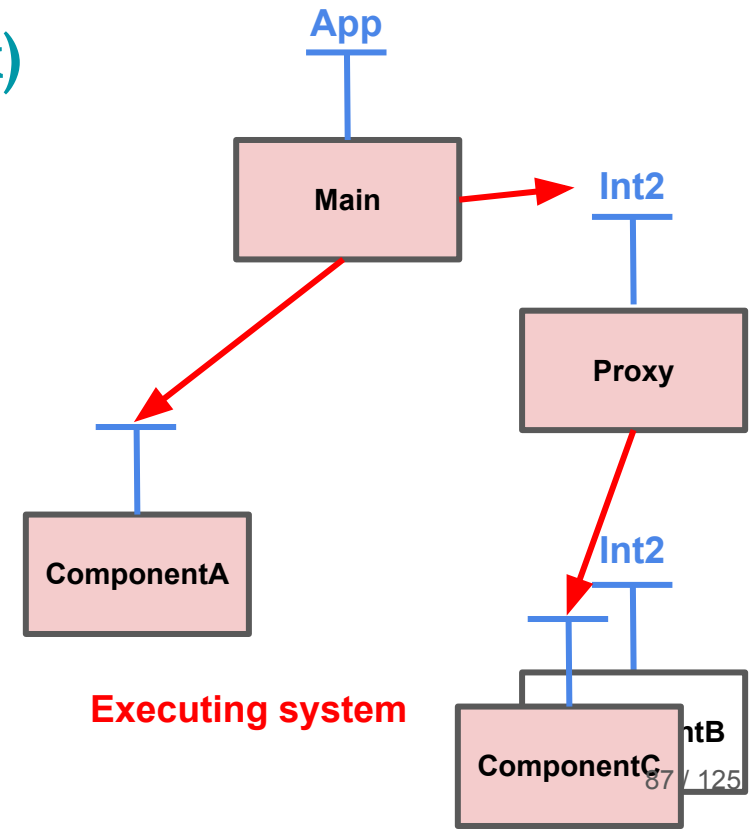
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Observing **Data:** (arch1: resp_time 20ms, text)



robertovito@ufg.br

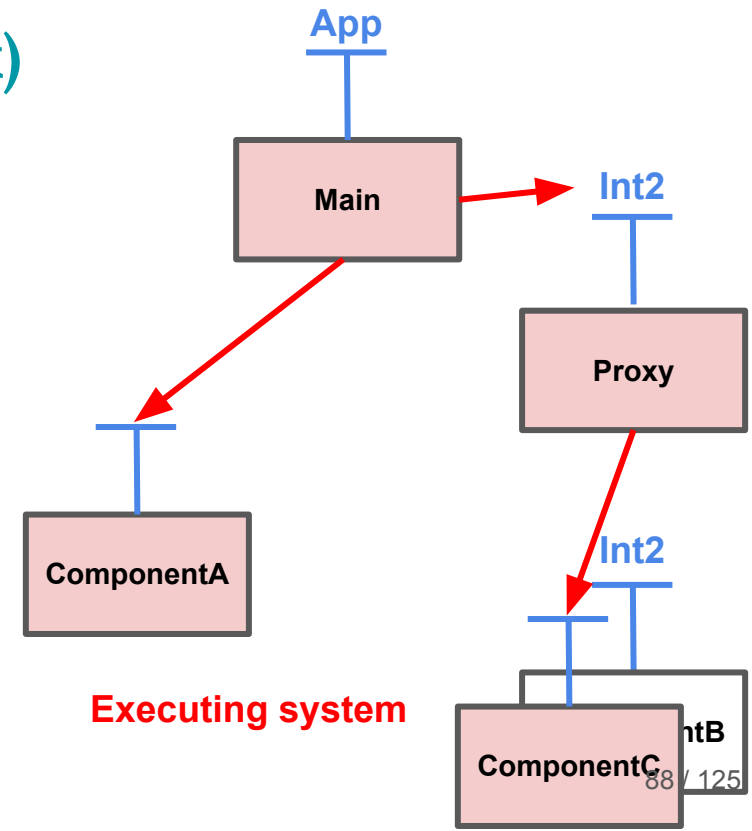
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Get perception data **Data:** (arch1: resp_time 20ms, text)



robertovito@ufg.br

INF INSTITUTO DE INFORMÁTICA UFG UNIVERSIDADE FEDERAL DE GOIÁS

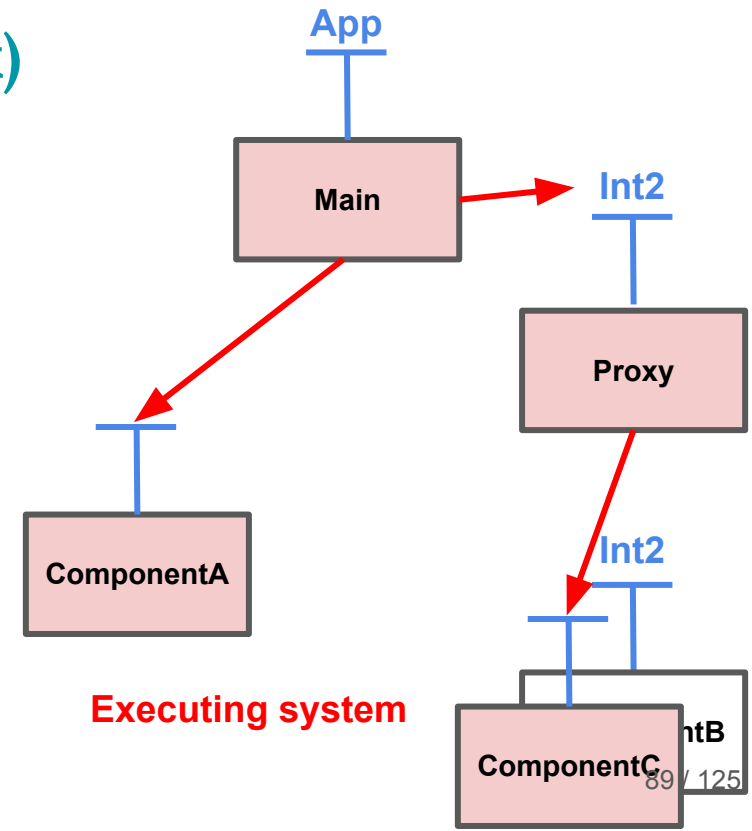
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Get perception data **Data:** (arch1: resp_time 20ms, text)
(arch2: resp_time 30ms, text)



Executing system

robertovito@ufg.br

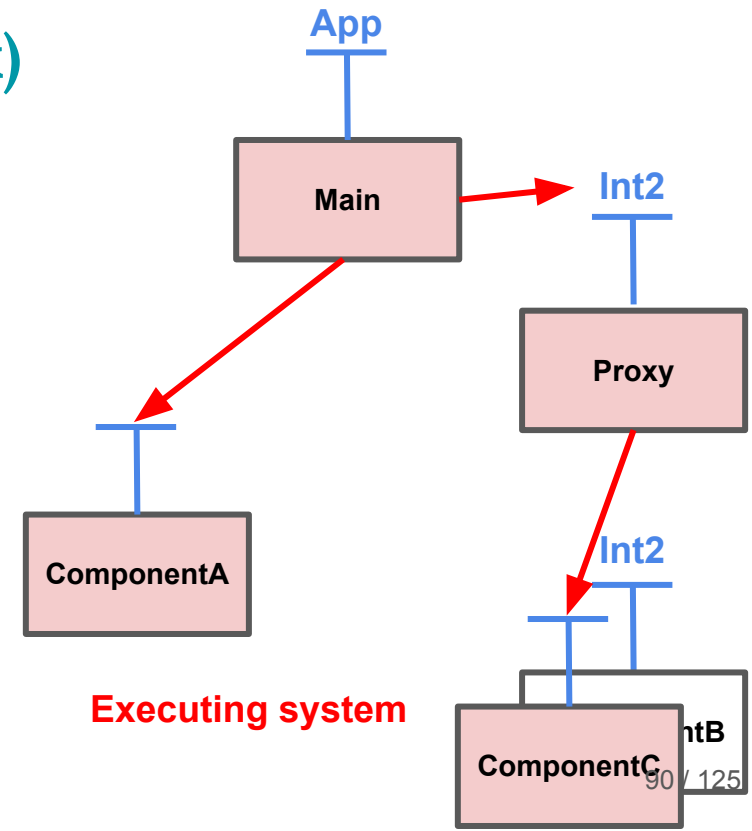
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Select best **Data:** (arch1: resp_time 20ms, text)
(arch2: resp_time 30ms, text)



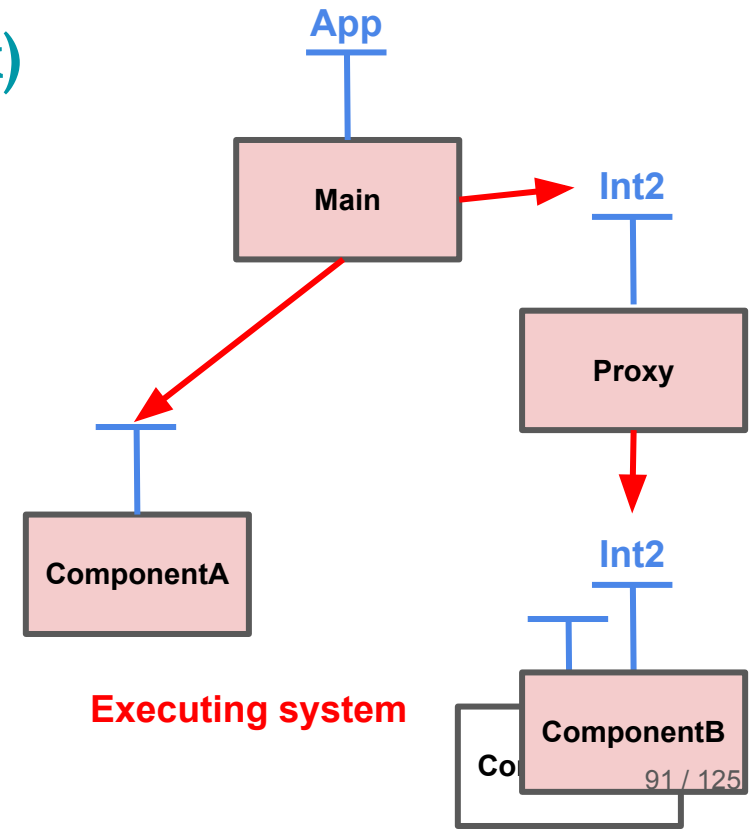
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- **Exploration**
- Exploitation

Status: Select best **Data:** (arch1: resp_time 20ms, text)
(arch2: resp_time 30ms, text)



→ Implementation (PAL framework)

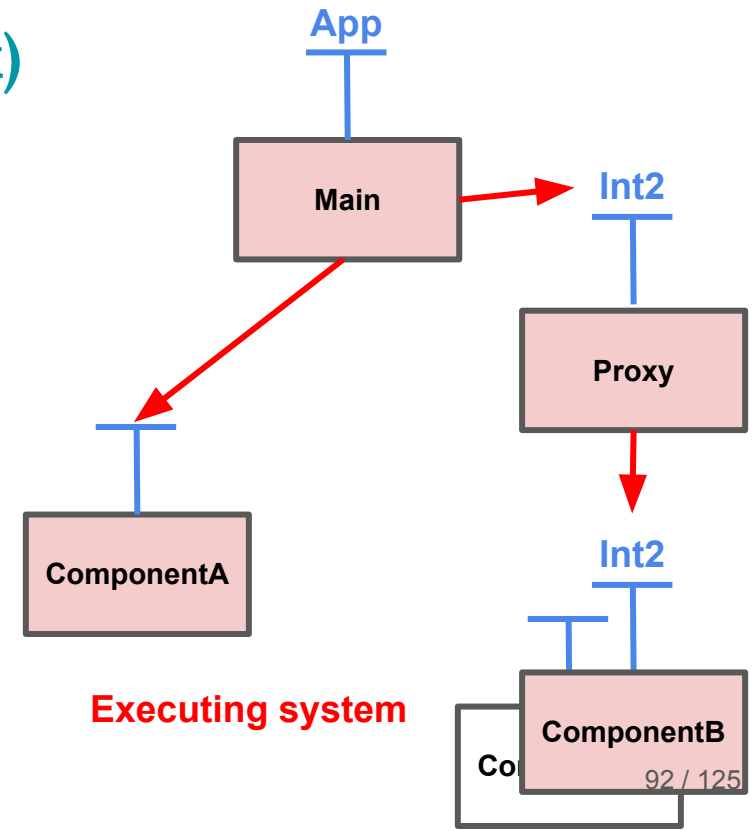
Learning

Baseline algo:

- Setup
- Exploration
- **Exploitation**

Consists of:

- For the best found composition;
  - Observe;
  - Collect data perception;
- If something changes: trigger exploration, otherwise continue;



Executing system

robertovito@ufg.br

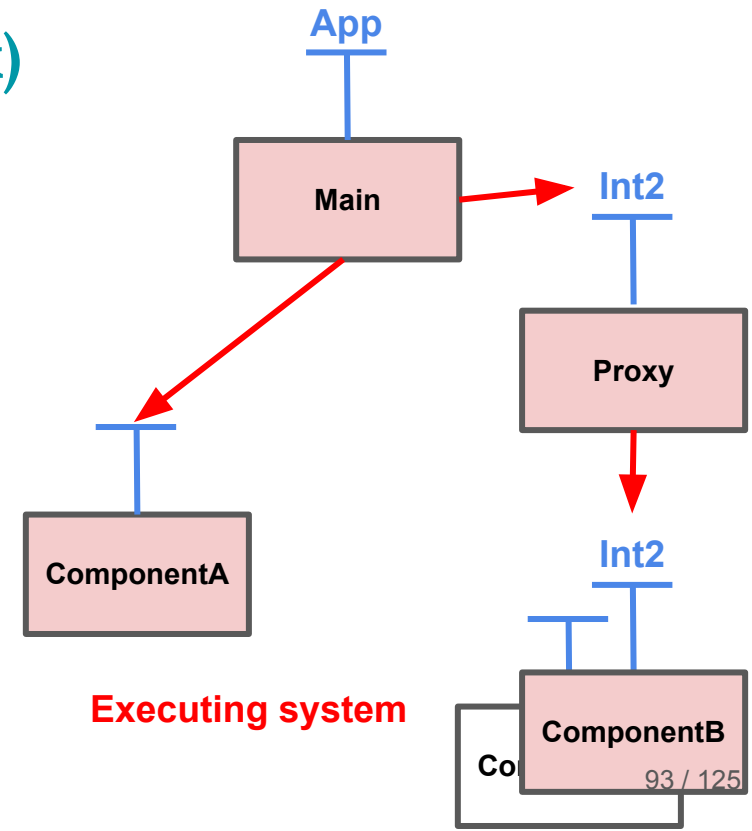
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- Exploration
- **Exploitation**

Status: Observing **Data:** (arch1: resp_time 20ms, text)
(arch2: resp_time 30ms, text)



Executing system

robertovito@ufg.br

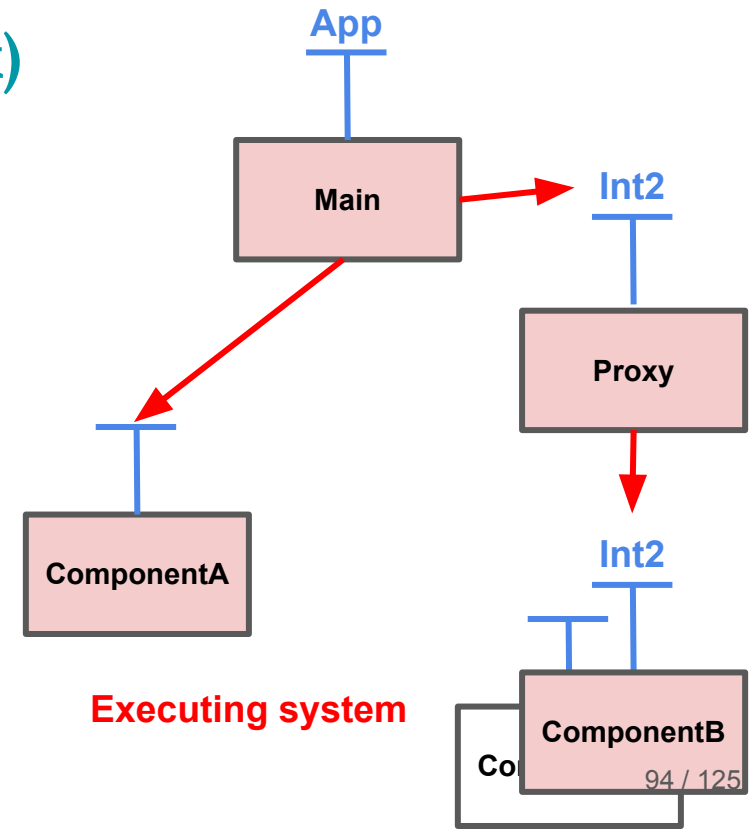
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- Exploration
- **Exploitation**

Status: Get perception data **Data:** (arch1: resp_time 20ms, text)
(arch2: resp_time 30ms, text)



robertovito@ufg.br

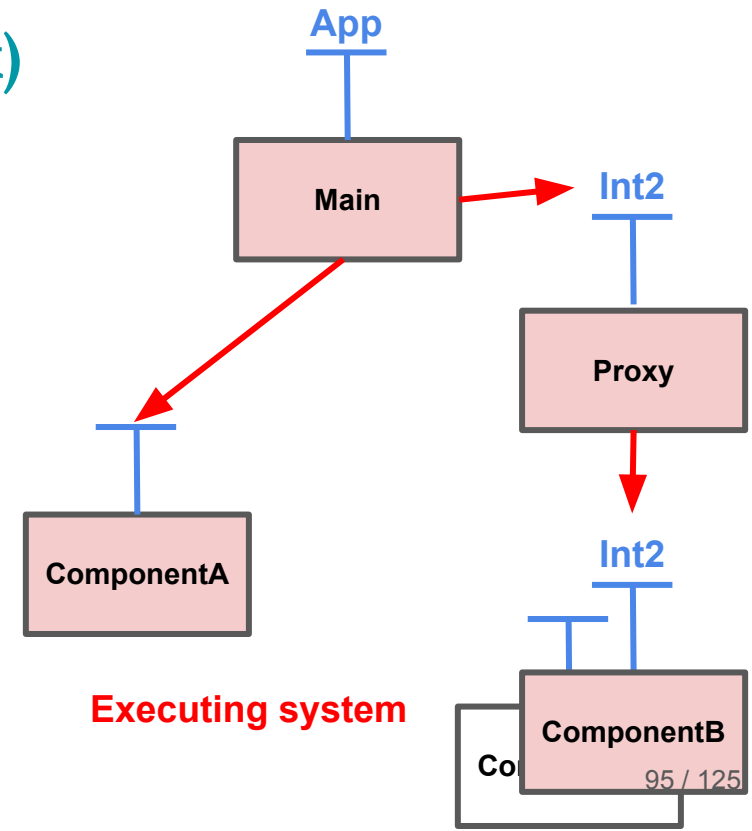
→ Implementation (PAL framework)

Learning

Baseline algo:

- Setup
- Exploration
- **Exploitation**

Status: Get perception data **Data:** (arch1: resp_time 20ms, text)
(arch2: resp_time 30ms, text)
(arch1: resp_time 25ms, text)



robertovito@ufg.br

Practice

robertovito@ufg.br

Case studies

- **Application of the Emergent Software System Concept**
- **Emergent Web Server**
  - Application of Emergent Software Systems concept to build a web server.
- **Emergent Microservice**
  - Application of Emergent Software Systems concept to build microservices.

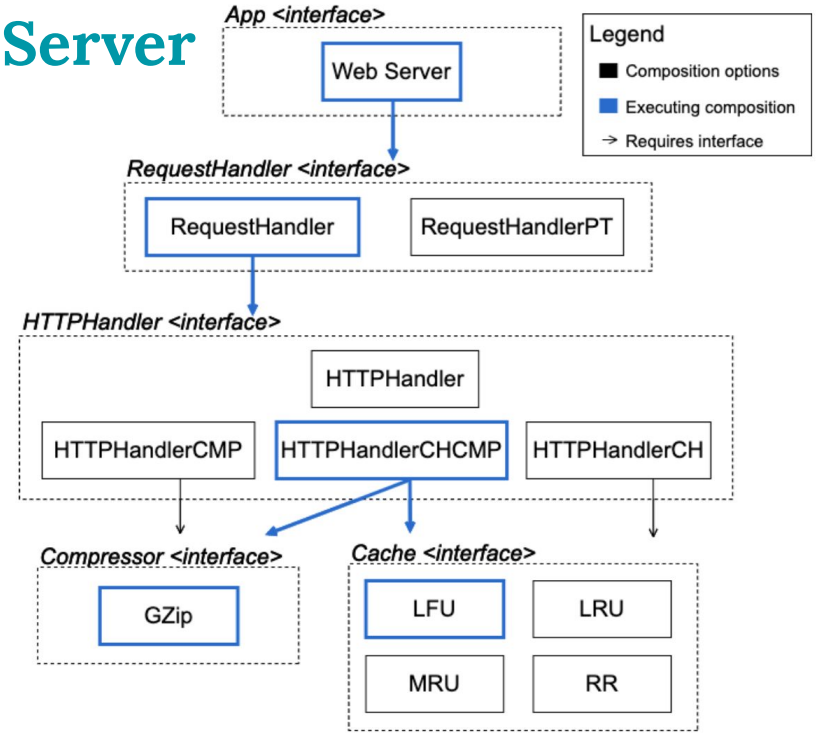
Case studies

➔ Emergent Web Server

- **Motivation:**
  - Web-based systems make up a large sum of systems we daily interact with;
  - Manually optimising web-based systems is challenging;
- **Goal:**
  - To find the best web server composition, at runtime, with no predefined domain-specific knowledge;

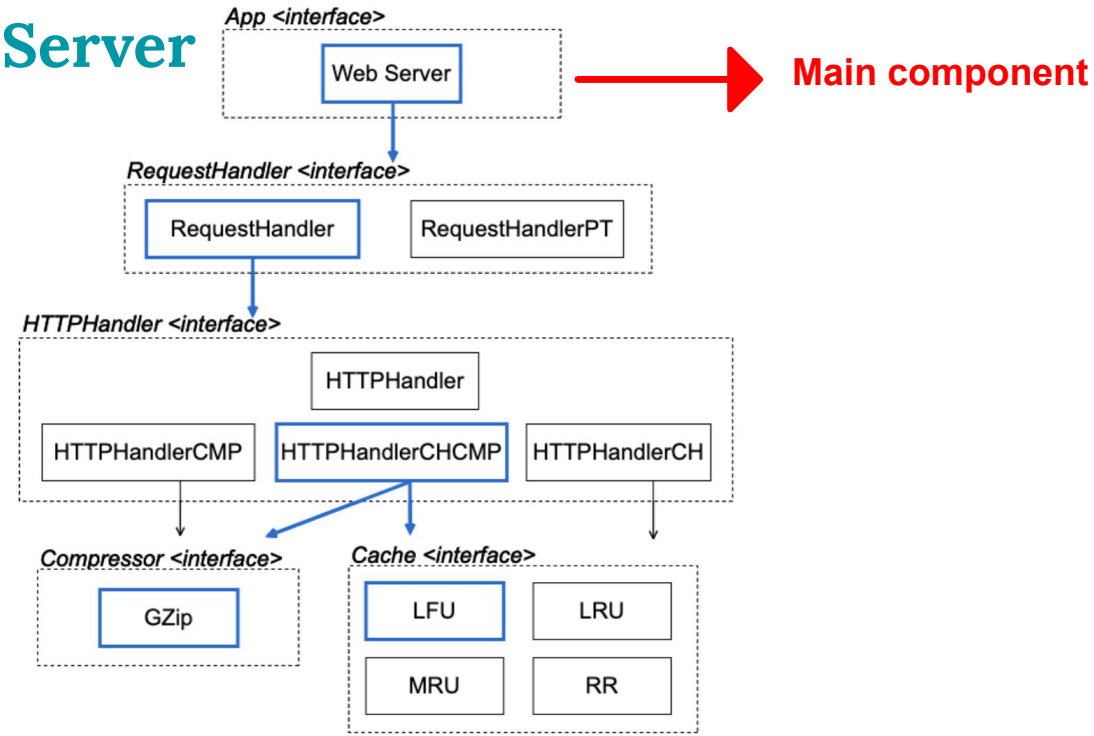
Case studies

➔ Emergent Web Server



Case studies

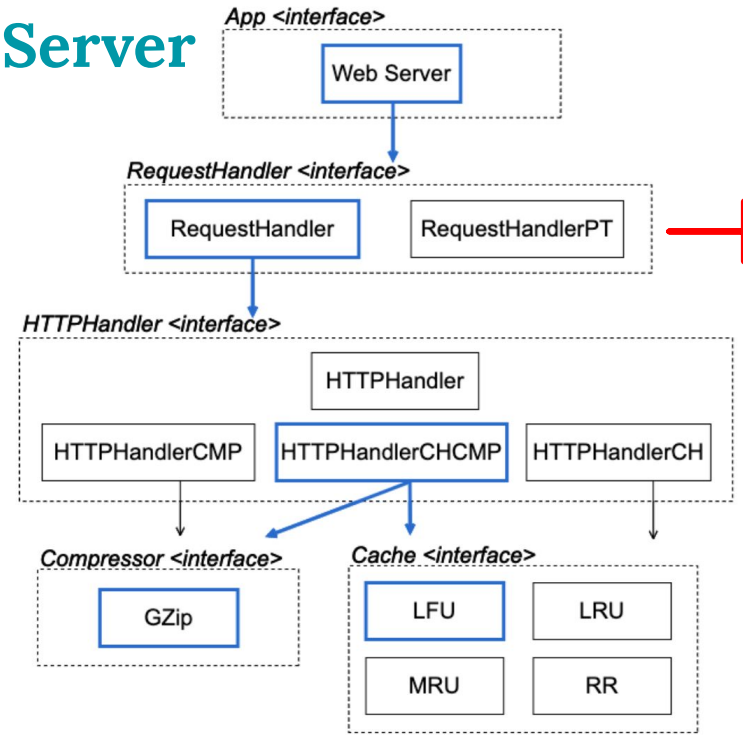
➔ Emergent Web Server



robertovito@ufg.br

Case studies

➔ Emergent Web Server

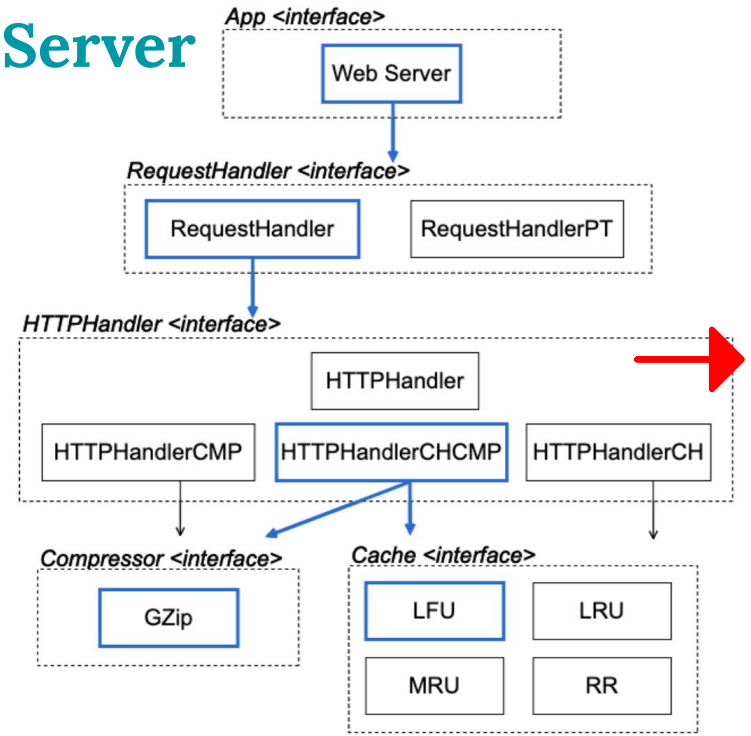


Handling TCP connections:

- **Creating a thread per client;**
- **Assigning a client to a thread from a pool;**

Case studies

➔ Emergent Web Server



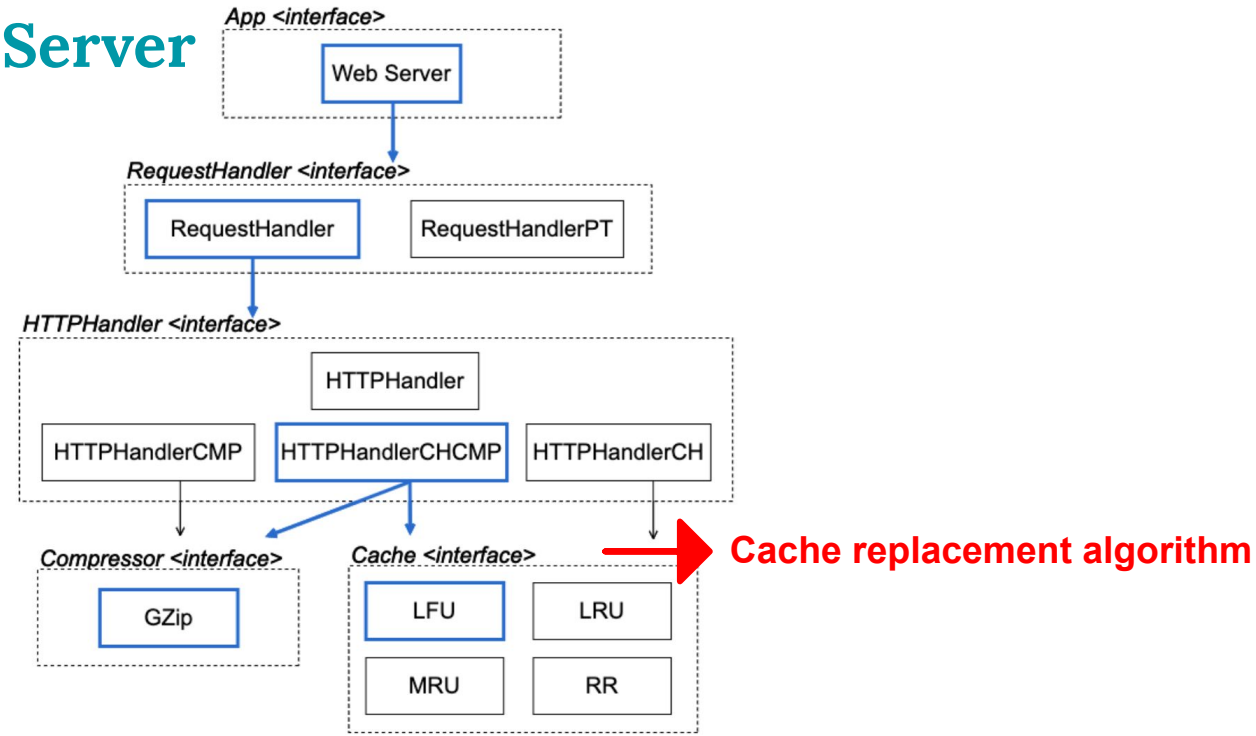
Handling HTTP request:

- **Default;**
- **Cache;**
- **Compression;**
- **Cache-compression;**

robertovito@ufg.br

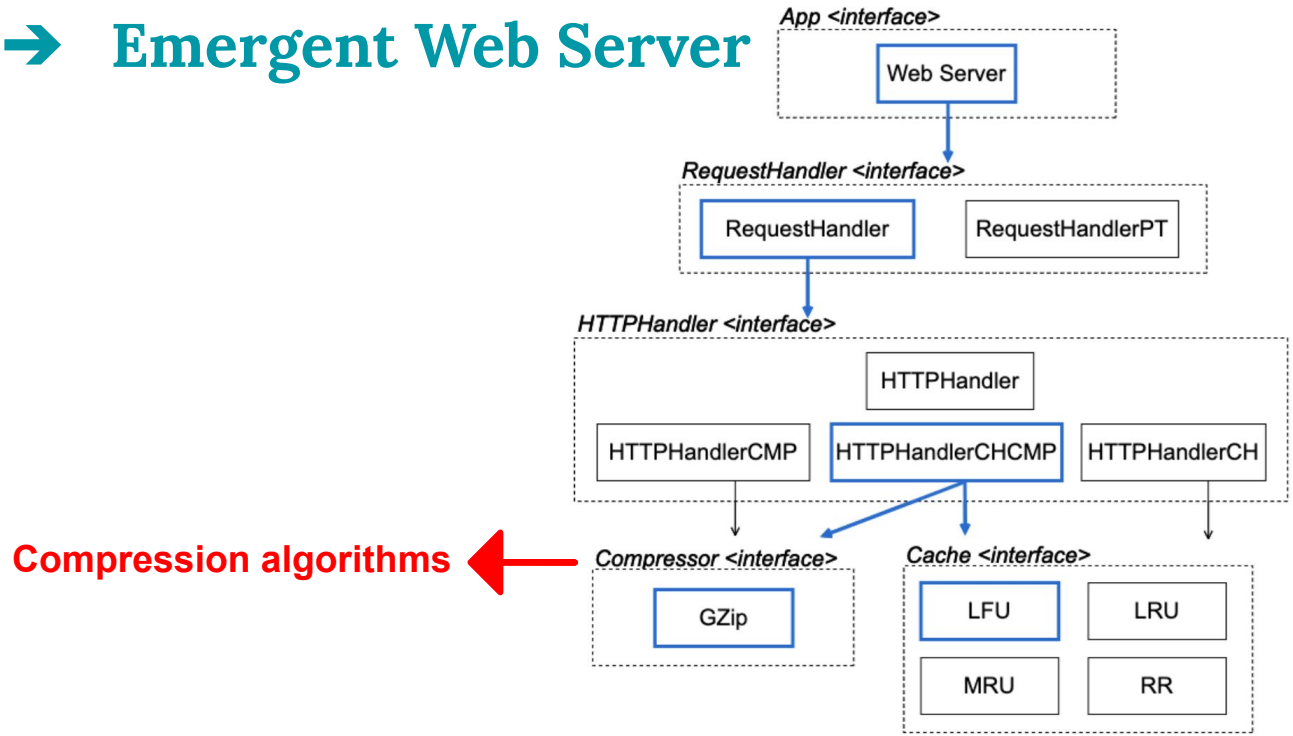
Case studies

➔ Emergent Web Server



Case studies

➔ Emergent Web Server

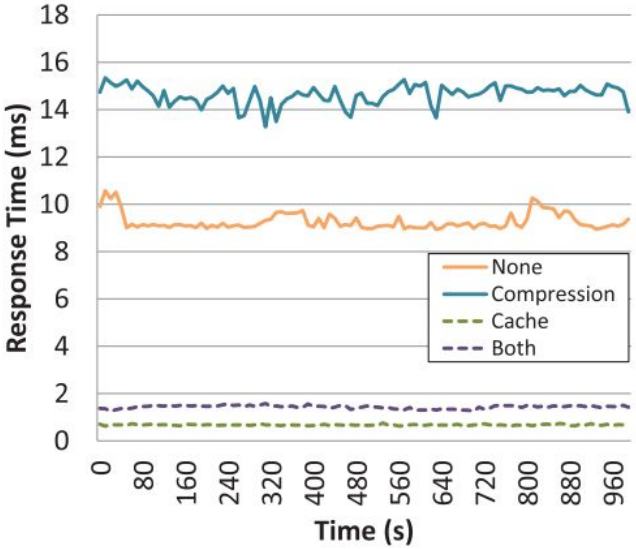


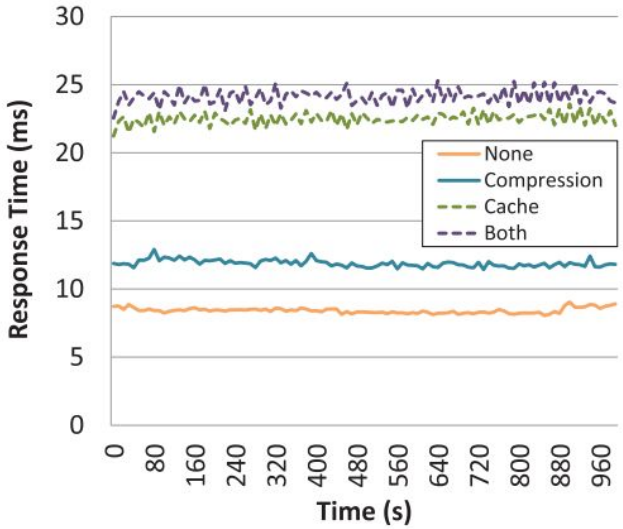
robertovito@ufg.br

Case studies

➔ Emergent Web Server

- Ground truth: Optimality divergence

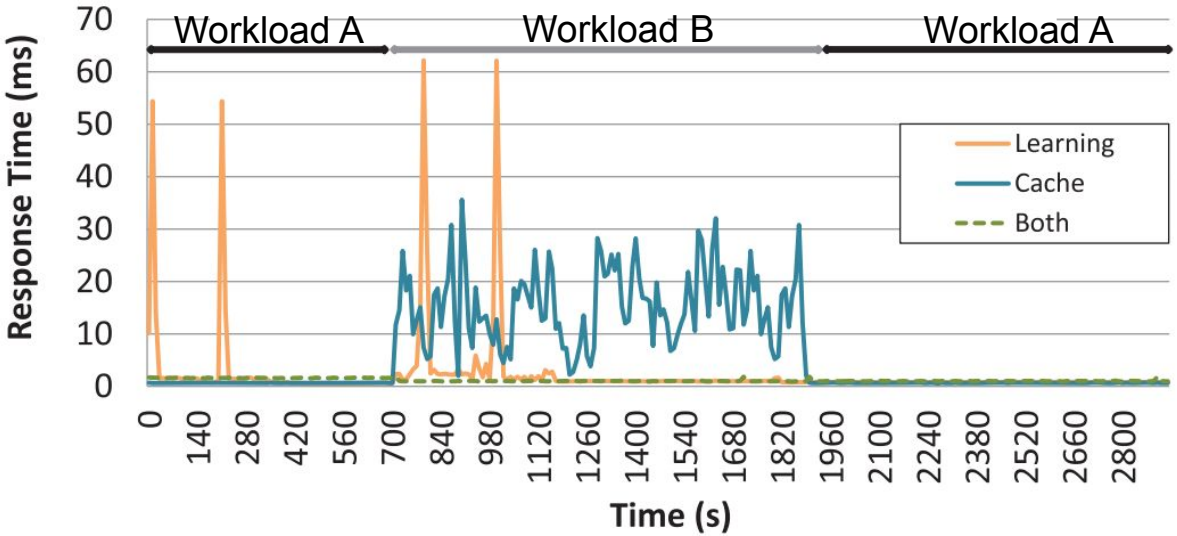




Case studies

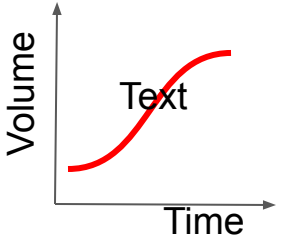
➔ Emergent Web Server

- Online Learning

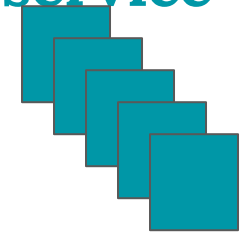


Case studies

➔ Emergent Microservice

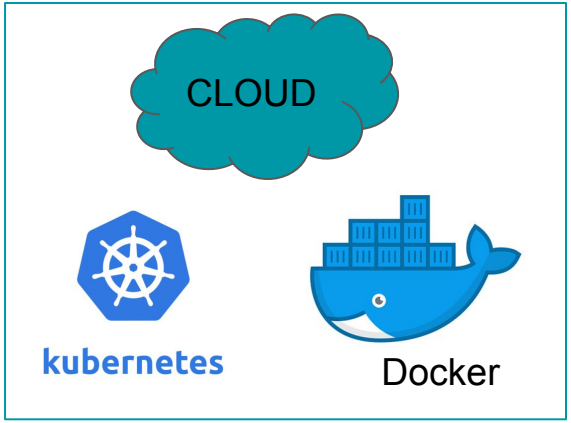


Changes in Workload Volume

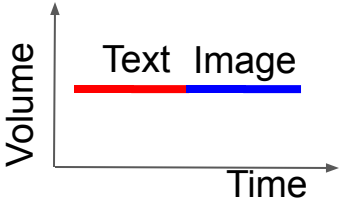


Microservices

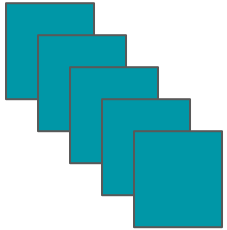
Relies on (at runtime)



Platform



Changes in Workload Pattern



Microservices

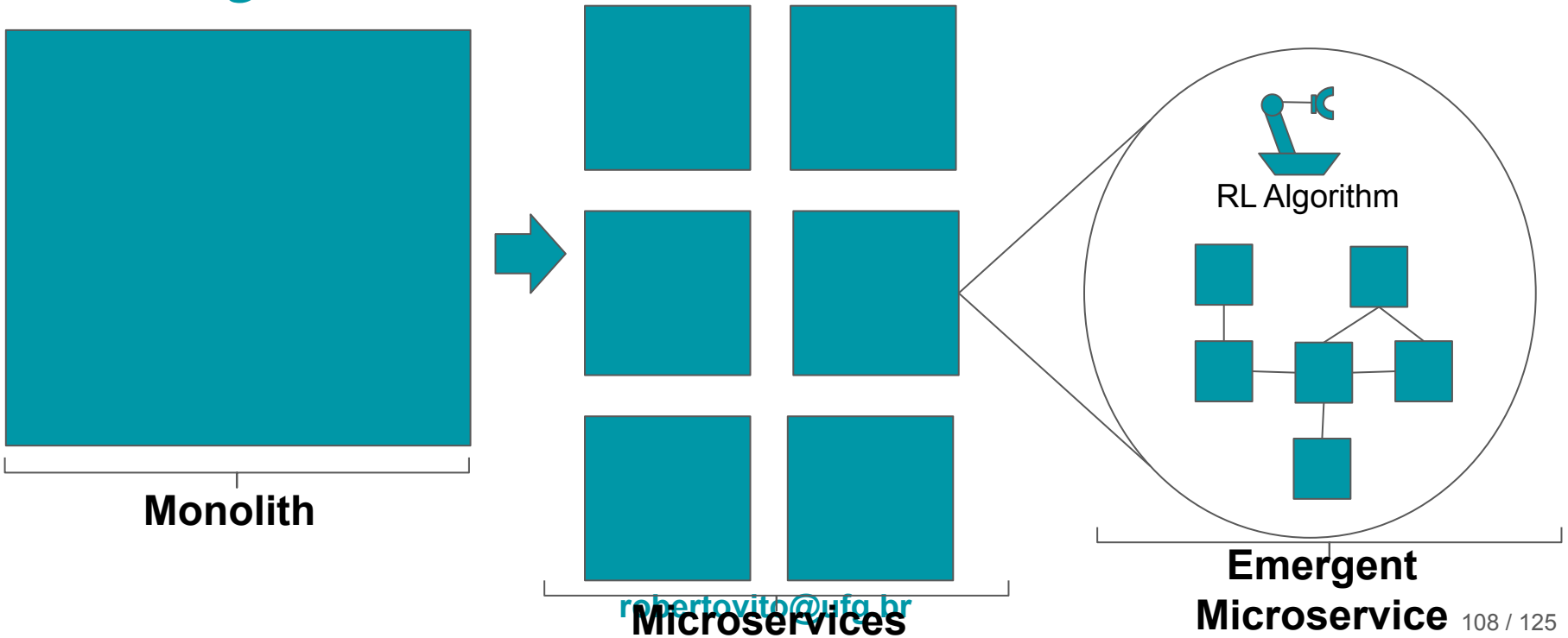
Relies on (at runtime)

NOTHING

robertovito@ufg.br

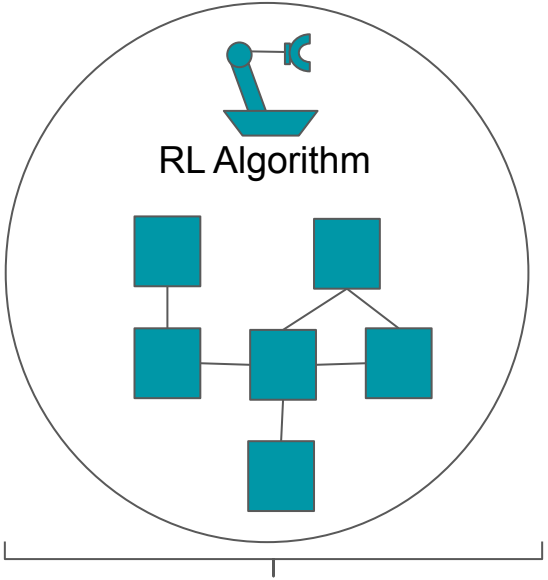
Case studies

➔ Emergent Microservice



Case studies

➔ Emergent Microservice

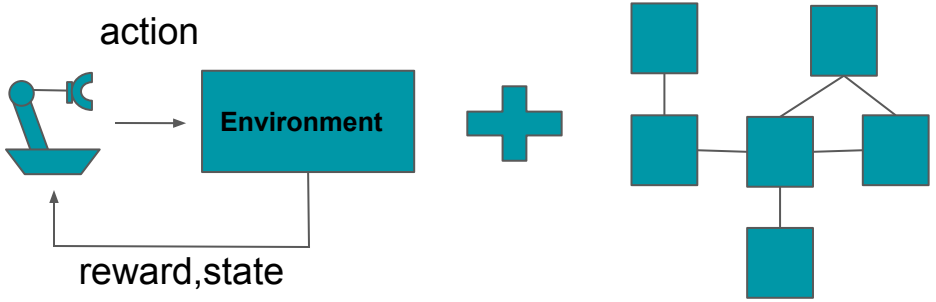


Emergent Microservice

=

Reinforcement Learning

Component-based Models

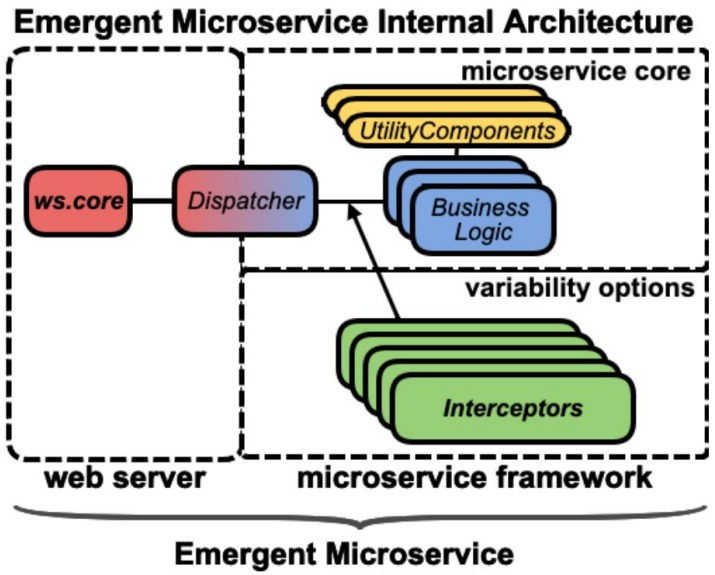


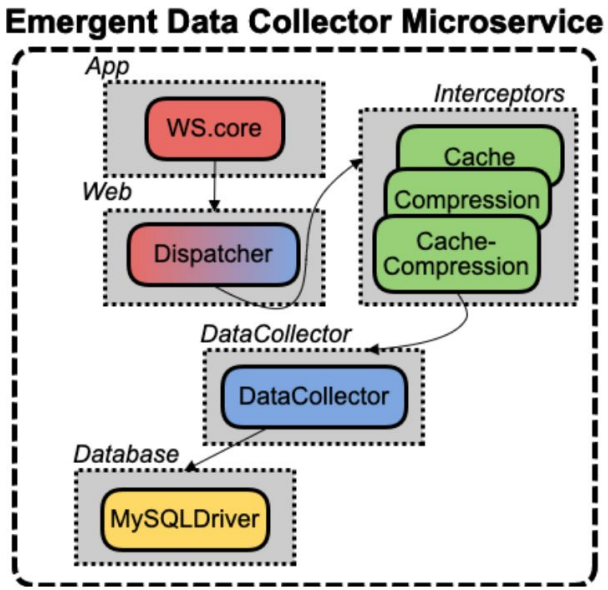
Emergent Software Systems concept

Microservice

Case studies

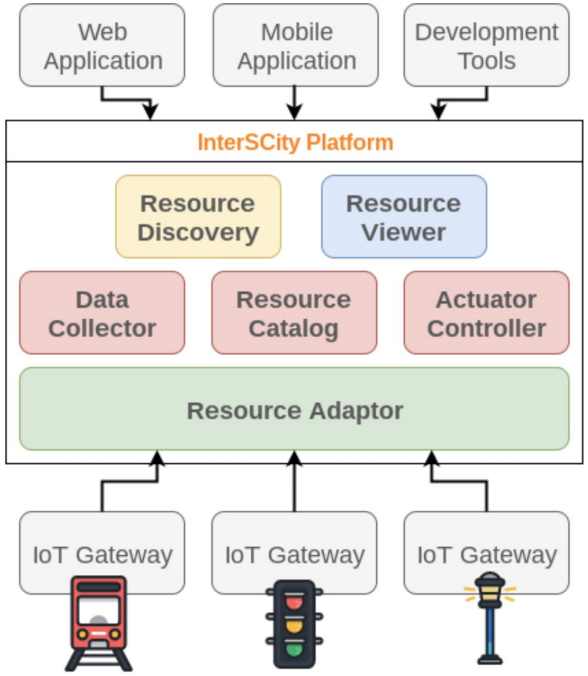
➔ Emergent Microservice





Case studies

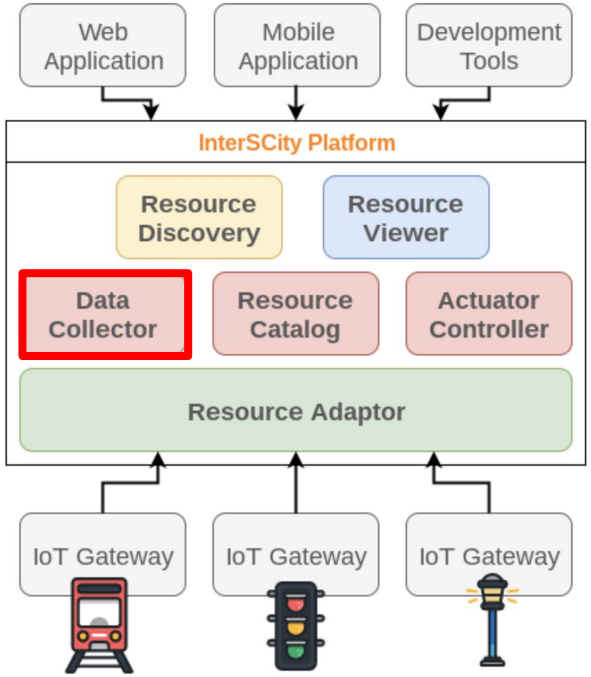
➔ Emergent Microservice



robertovito@ufg.br

Case studies

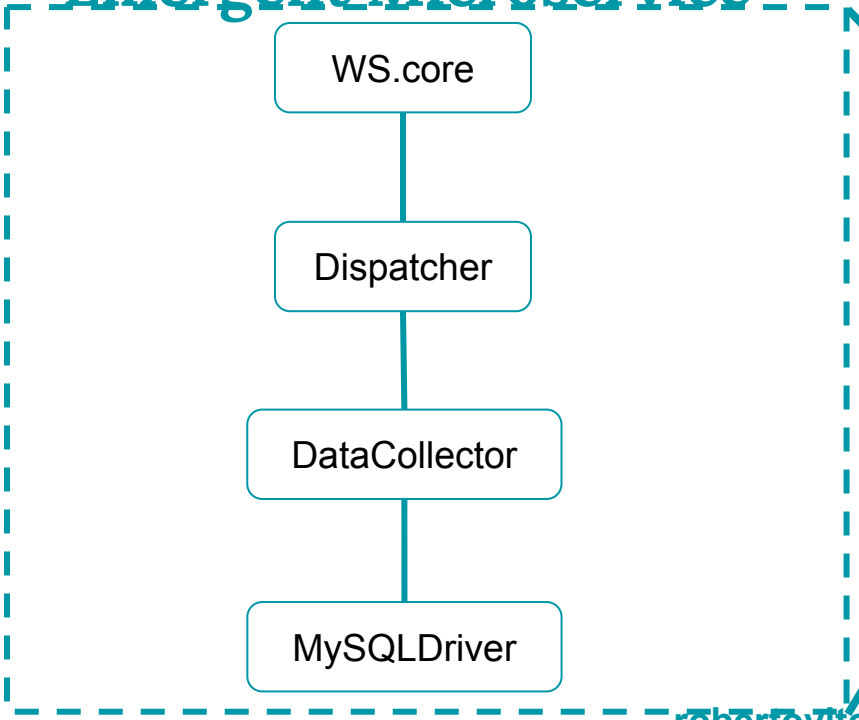
➔ Emergent Microservice

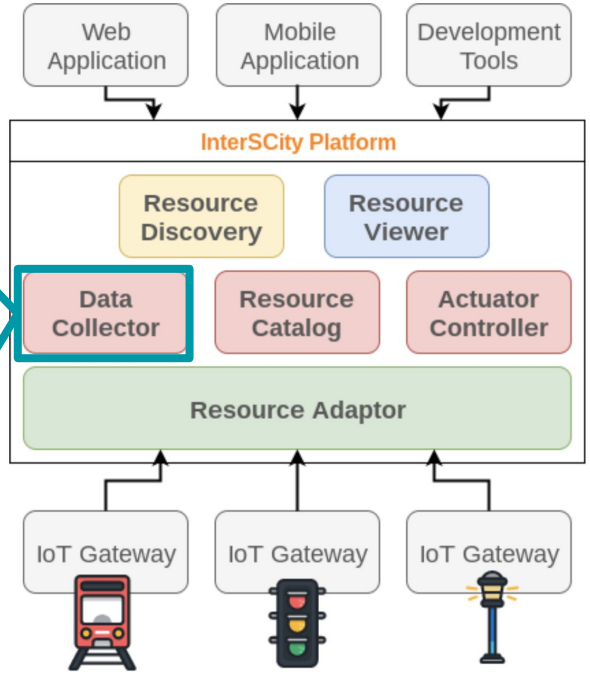


robertovito@ufg.br

Case studies

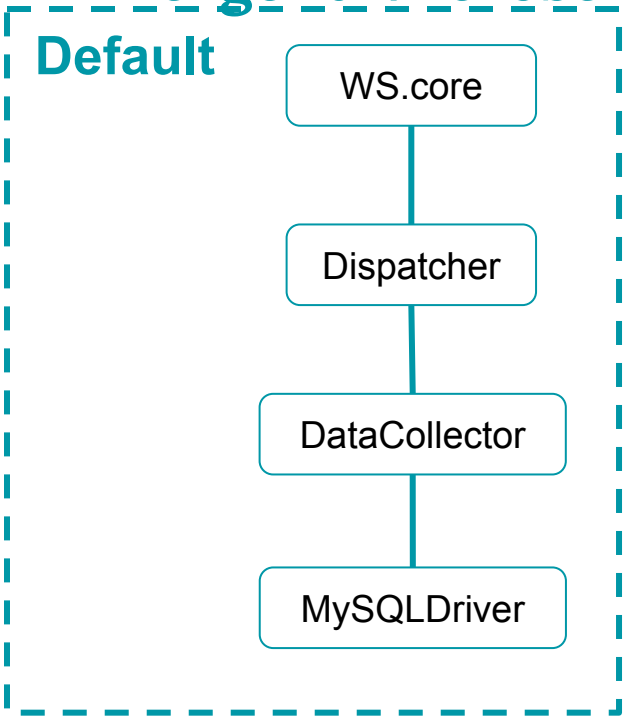
→ Emergent Microservice



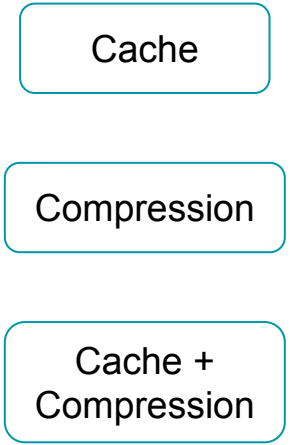


Case studies

➔ Emergent Microservice



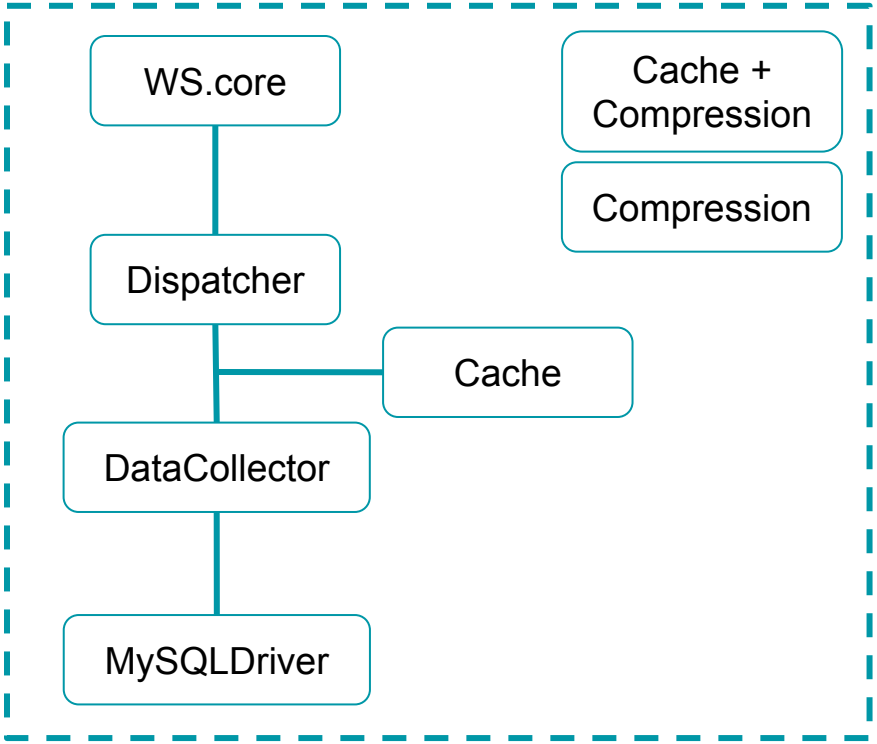
Components that generate composition variants for DC



robertovito@ufg.br

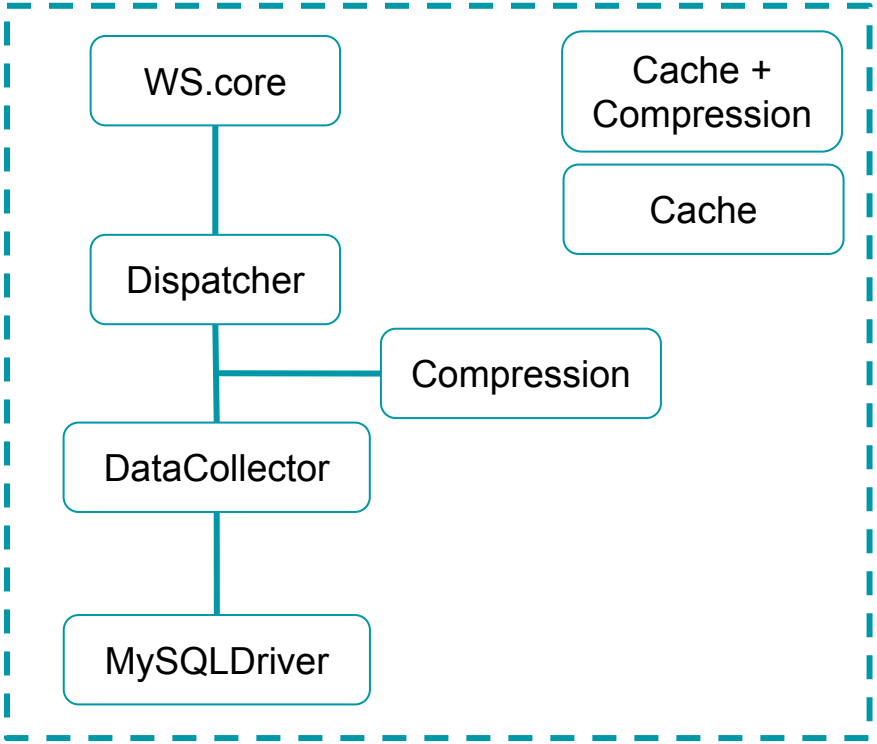
Case studies

➔ **Emergent Microservice**



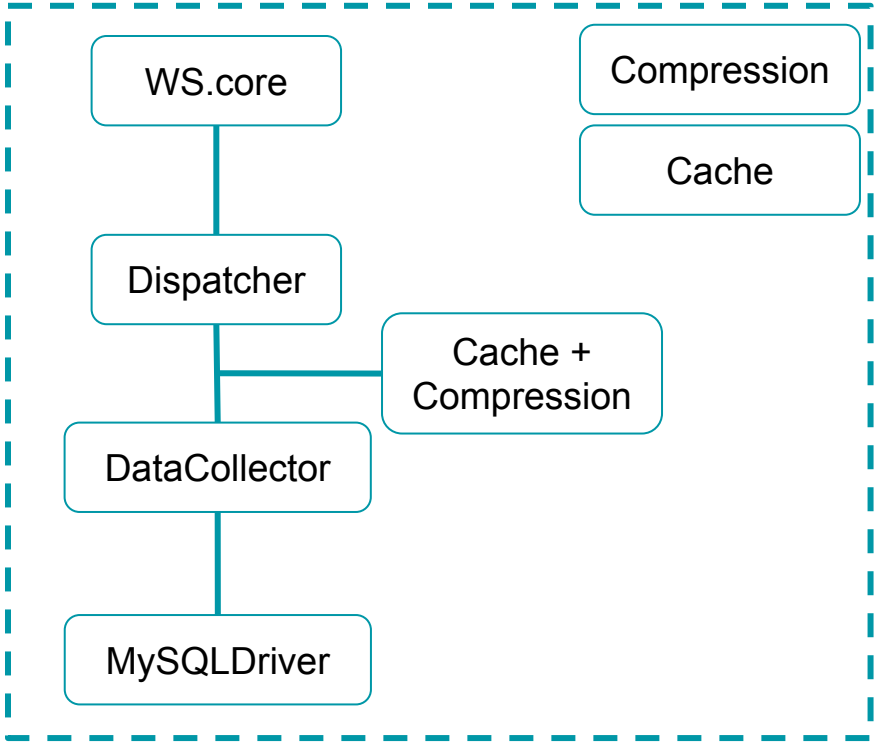
Case studies

➔ **Emergent Microservice**

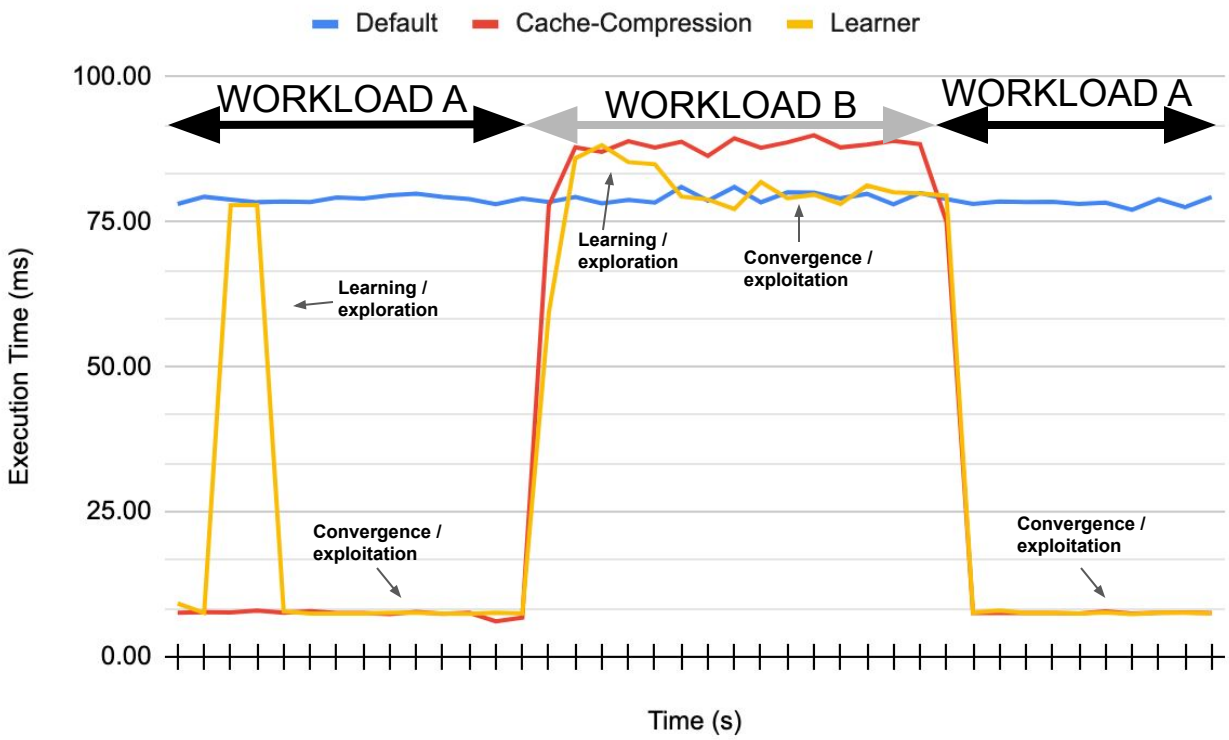


Case studies

➔ **Emergent Microservice**



Case studies



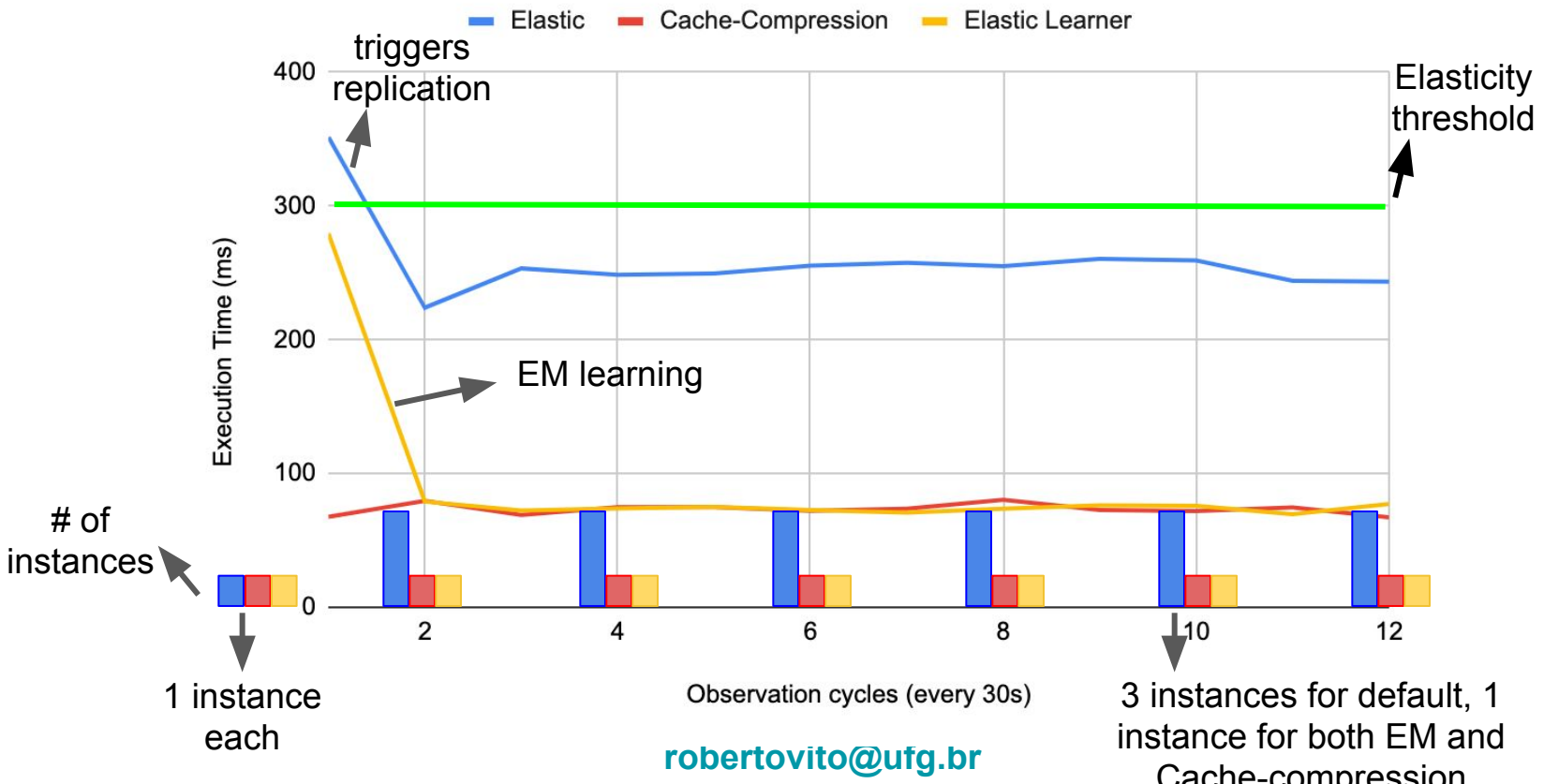
robertovito@ufg.br

Case studies

➔ Emergent Microservice

- Horizontal Pod Autoscaler (elasticity):
  - Set a metric (e.g., CPU utilisation);
  - Set a metric threshold;
  - Monitor the system;
  - Replicate POD in case the metric surpasses the threshold;

Case studies



robertovito@ufg.br

Tool

➔ Dana programming language

Introducing the language

Showing its syntax, compiling and running a program

Please visit: https://github.com/robertovrf/sbrc21_minicurso

Demo

➔ Interactive Emergent System

Running Interactive Emergent System tool and showing how to interact with PAL through a command prompt

Please visit: https://github.com/robertovrf/sbrc21_minicurso

Demo

➔ Emergent Web Server

Executing the Emergent Web Server

Please visit: https://github.com/robertovrf/sbrc21_minicurso

Practical

➔ Hands-on experience with the Emergent Web Server

Access our Github repository and follow the ShortCourse Practical

Please visit: https://github.com/robertovrf/sbrc21_minicurso

INCT InterSCity

robertovito@ufg.br

FAPESP 60 CIÊNCIA, CULTURA E DESENVOLVIMENTO 1962 - 2022

INF INSTITUTO DE INFORMÁTICA

UFG UNIVERSIDADE FEDERAL DE GOIÁS