Continuous Operations and Fully Autonomy of a Social Service Robotic System.

Bilal Hoteit
Imad Alex Awada
Alexandru Sorici
Adina Magda Florea

Faculty of Automatic Control and Computer Science
University Politehnica of Bucharest

SYNASC2021: 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

High level Outline

1. Introduction
2. State of the Art
3. Design and Implementation
    - 3.1. HS-RFA
    - 3.2. Task Component
    - 3.3. Planning and Executing Component
4. Conclusion

High level Outline

1. Introduction

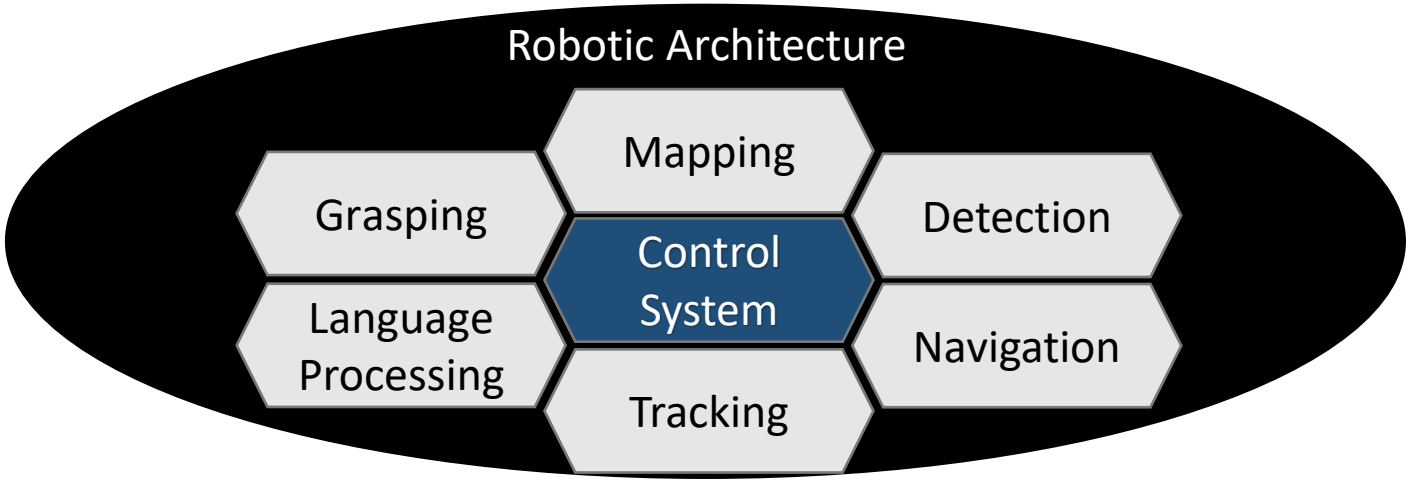
2. State of the Art

3. Design and Implementation
    - 3.1. HS-RFA
    - 3.2. Task Component
    - 3.3. Planning and Executing Component

4. Conclusion

Introduction - Robotic Architecture

- Robot is an intelligent agent – needs for a complex hardware and software architecture.
- Robotic systems varied based on the user's needs, environment, or tasks.



- Each component is related to one or more intellectual processes – AI techniques and tools.
- Network, Cloud, Edge computing enhance the capabilities of robotic systems.
- Fog Robotics is expected to become widespread in the next few years.

Introduction - Social Assistive Robot

- Robots will soon be a necessity for human life.
- It is used as a Medical Robot, Service Robot, Teacher robot.

Problem 1

- No general architecture for this type of robot in which developers can follow.

Problem 2

- How to integrate the planning approach to increase the autonomy of the robotic system.

Problem 3

- How to automate continuous operations for such a complex robotic system.

Introduction - Objective of the paper

Goal

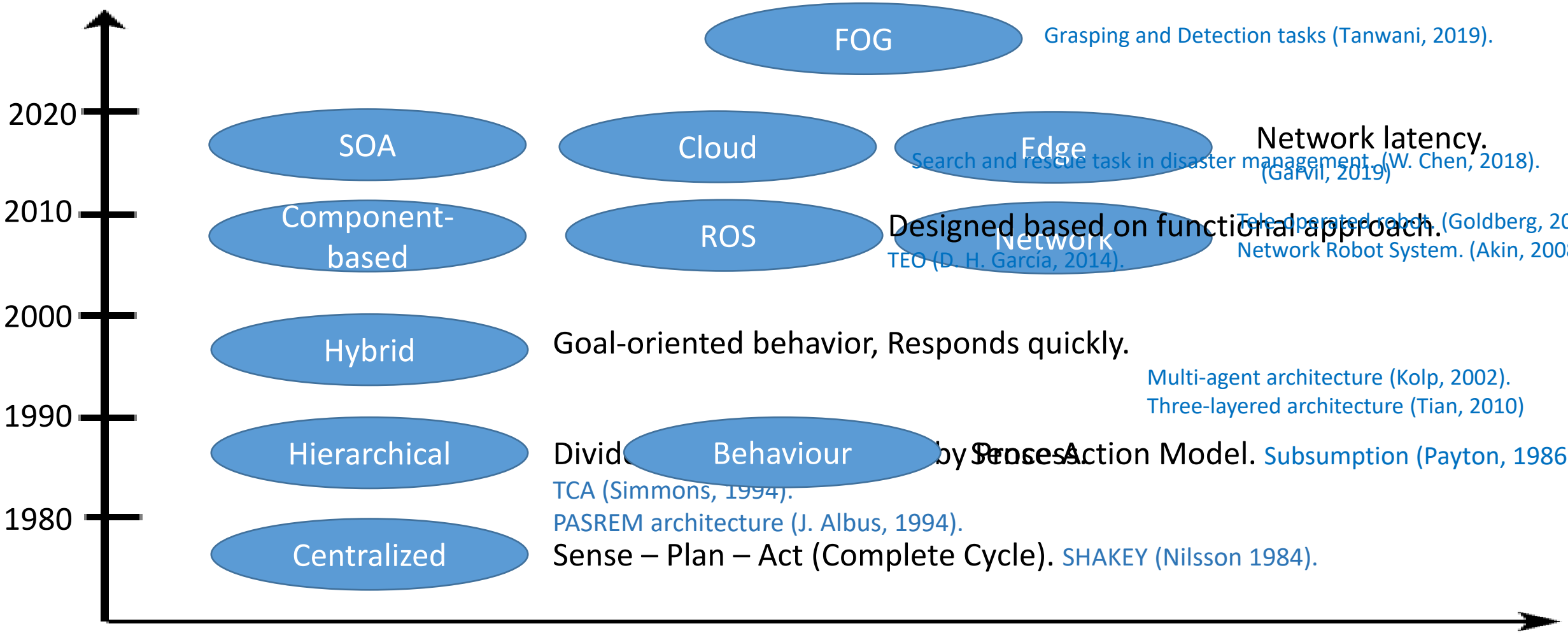
- Towards a suitable robotic system for a social service robot.
- Maintains a goal-directed behavior through AI symbolic planner (extended ROSPlan framework).
- Implement a fully autonomous operations for the robotic system.

Constraints

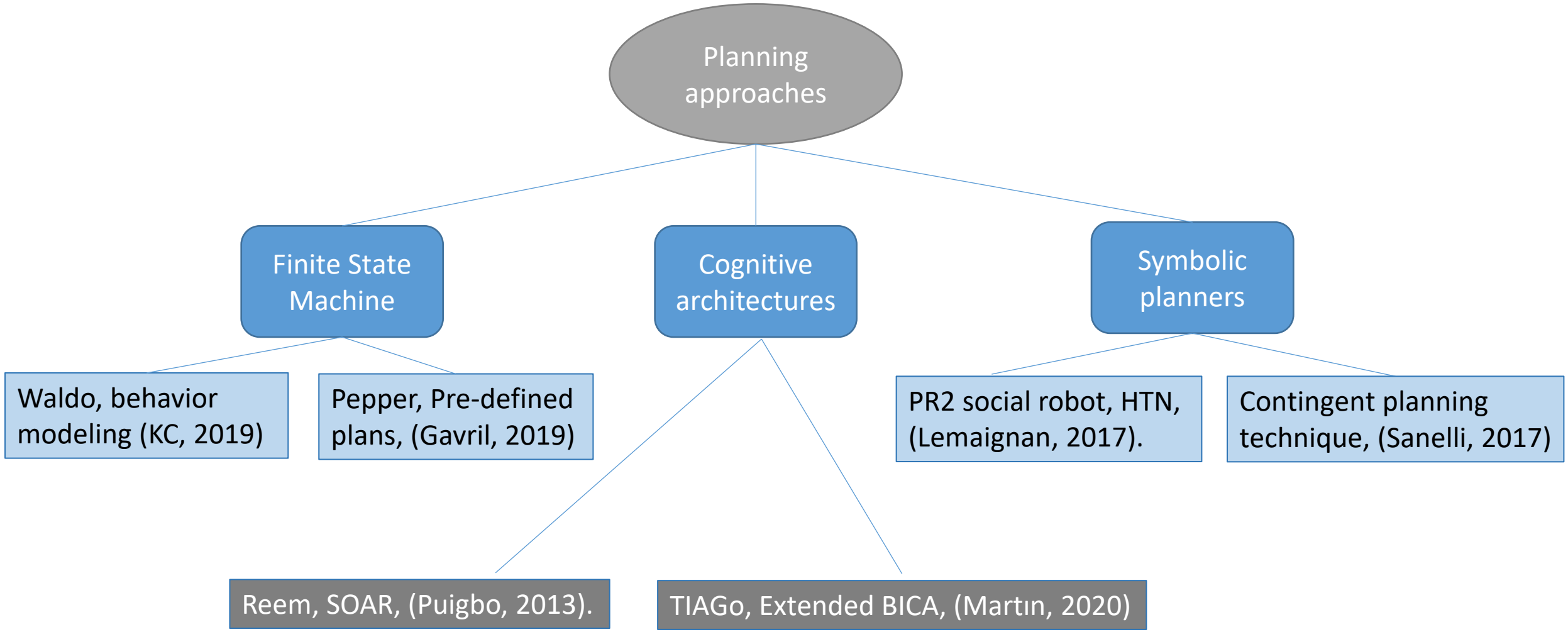
- Process in fully autonomous manner and continuous operations.
- Handle system components as a black boxes.
- Secure intensive resources.
- Provide somehow a human-like capabilities.
- Ensure the quality of the system.

High level Outline

State of the Art - Architecture



State of the Art - Planning

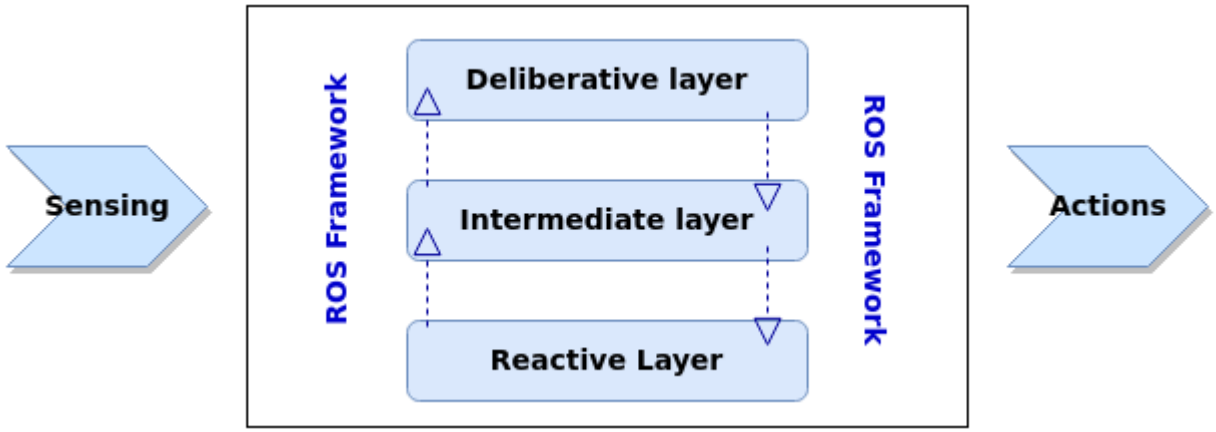


High level Outline

High level Outline

Design and Implementation - HS RFS / Overview

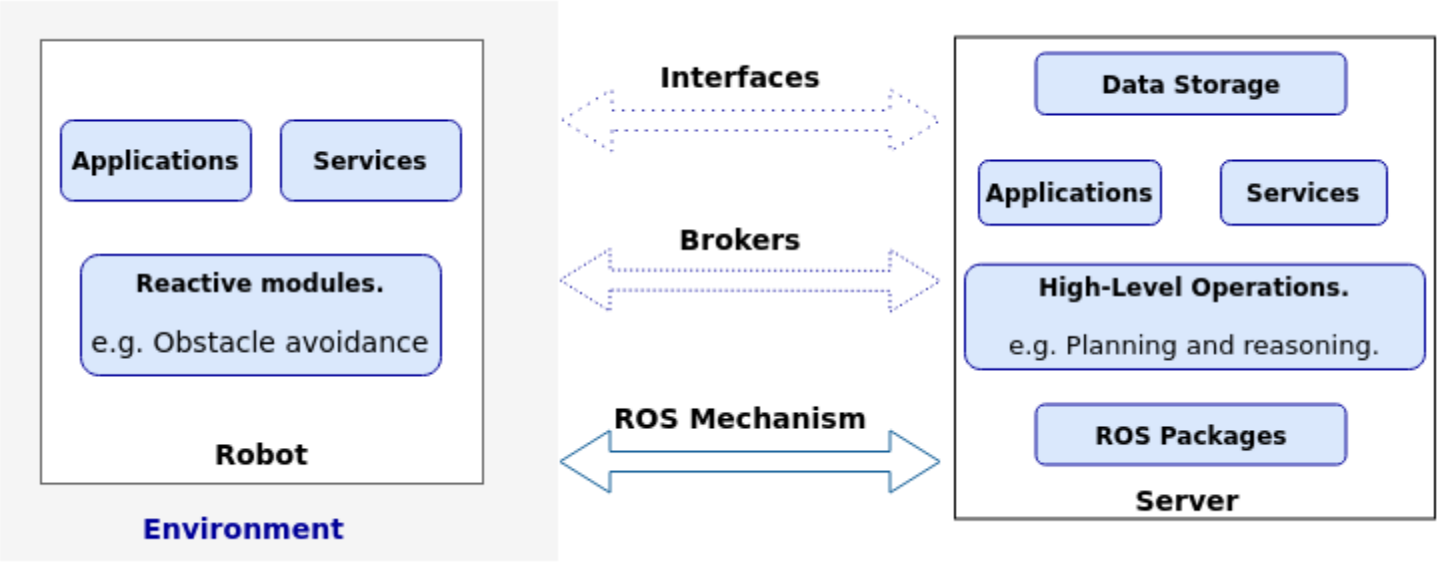
- (HS-RFS) is based on ROS, integrates robotic applications in a structured engineering manner.



- Components are managed in a layered fashion that also immerses spontaneously in HS-RFS.
- ROS contributes to the distribution of services, the ease of communication between components.
- (HS-RFS) followed hybrid architecture, implemented sense, plan, and act phases iteratively.
- System’s components provide specific skills and adopt the behavior-based model.

Design and Implementation - HS RFS / Methodology

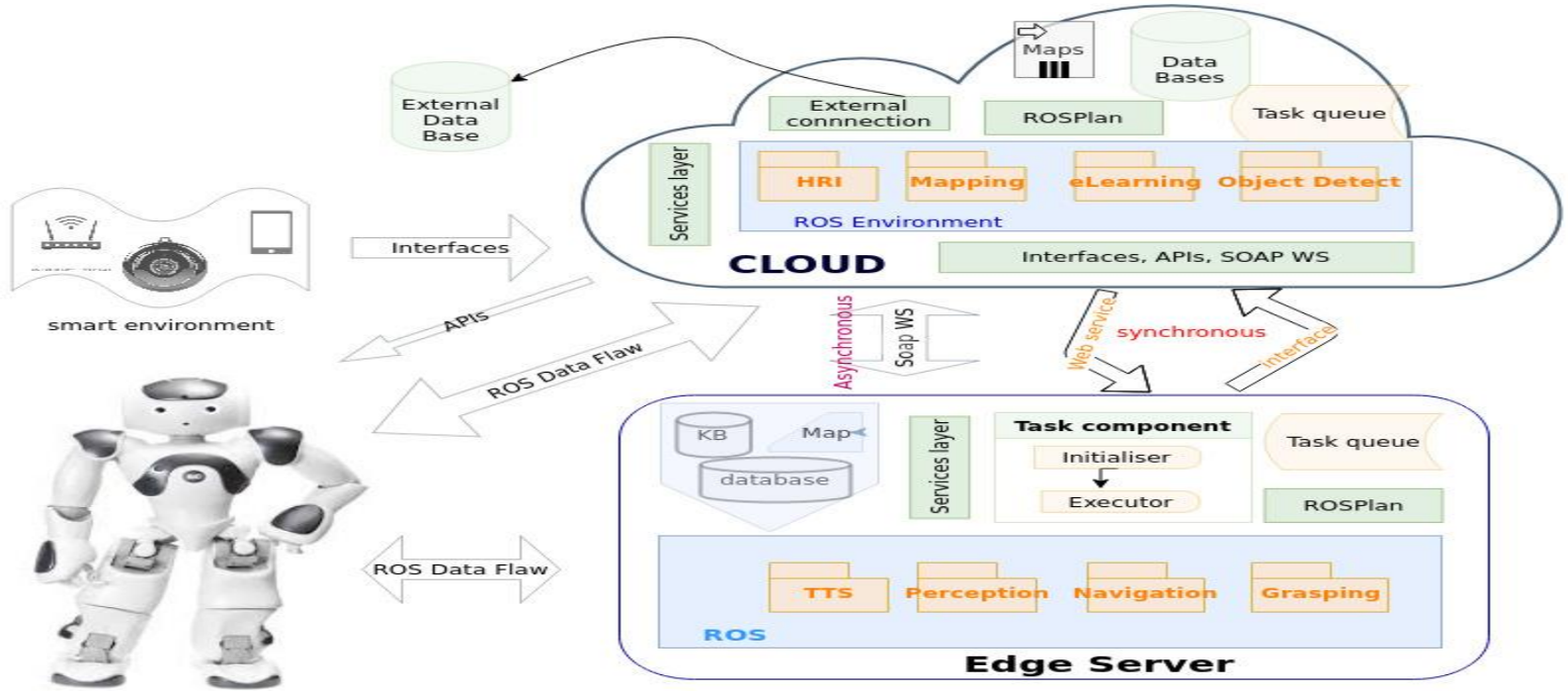
- The advancement of communication resulted in a fully distributed automated system.



- Relying on a local server that processes the data flow sent, stores useful information in database.
- Local server provides intensive resources to publish results in real time.

Design and Implementation - HS RFS / Description

- Relying on a cloud server that handles transmitted data, prepares the data for storage and analysis.



- Standard APIs are provided for asynchronous and synchronous data exchange.
- Multimaster fkie package is used to control and manage multiple ROS master networks.

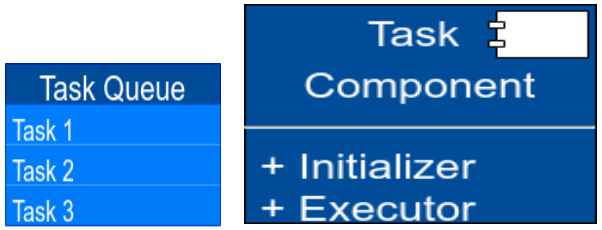
Design and Implementation - HS RFS / Description

System Components

- Several components such as Robotic applications (Navigation, Mapping, and Detection).

Task Component

- Accept missions from HRI, internal process, or application.
- Decompose, Schedule, Manage, Match and execute tasks.
- Saves jobs in a specific task queue on a local server.



Extended ROSPlan Framework

- Generates and executes plan relying on local server.
- Cloud clone creates a global plan, advises the executable ones.
- Symbolic planner implemented on both servers.

Design and Implementation - HS RFS / Description

System Communication

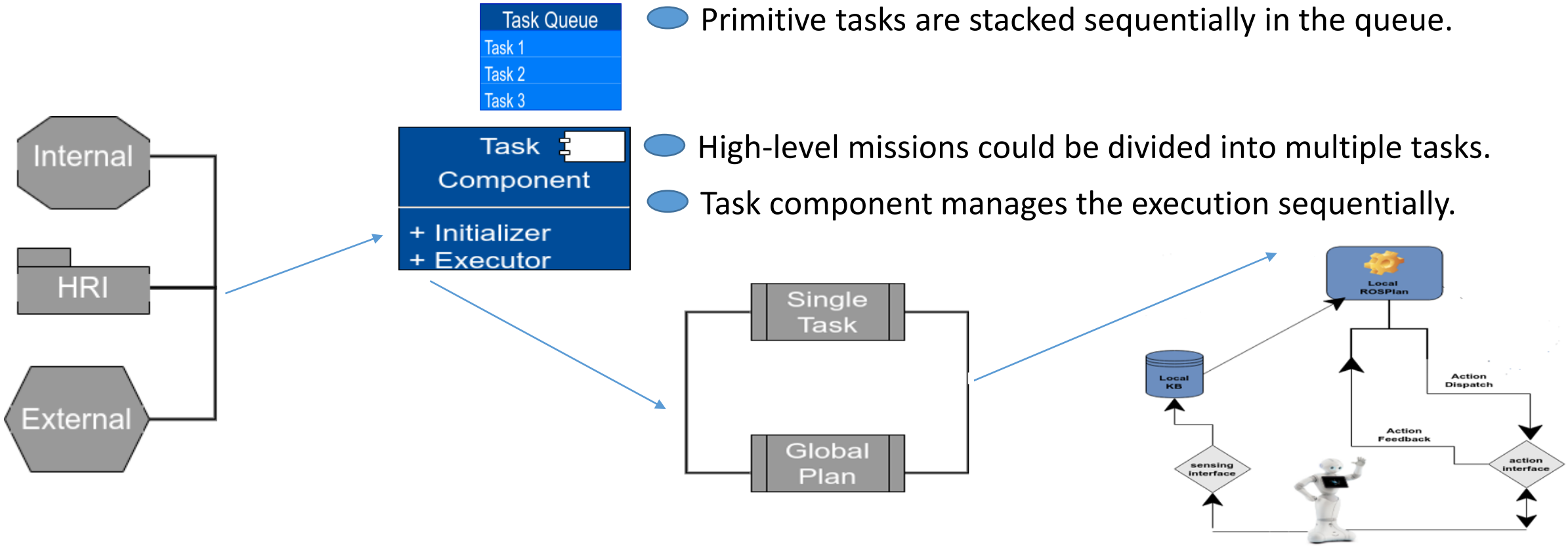
- The ROS framework runs on the robot, local, and cloud server.
- Nodes executed on the robot publishes to both cloud and local nodes simultaneously.
- Data is exchanged between local and cloud servers by using web interfaces or APIs.
- Edge plays the mediating role by transferring indirect data between the robot and cloud.

System Function

- Robotic components can existed on cloud or local based on application characteristics.
- Cloud infrastructure complements local infrastructure.
- The local robot infrastructure can be a backup of cloud during a bad internet connection.
- Robots can rely on data collected in a first or separate step.
- The system will work efficiently with cloud, while it works as possible with the local only.

High level Outline

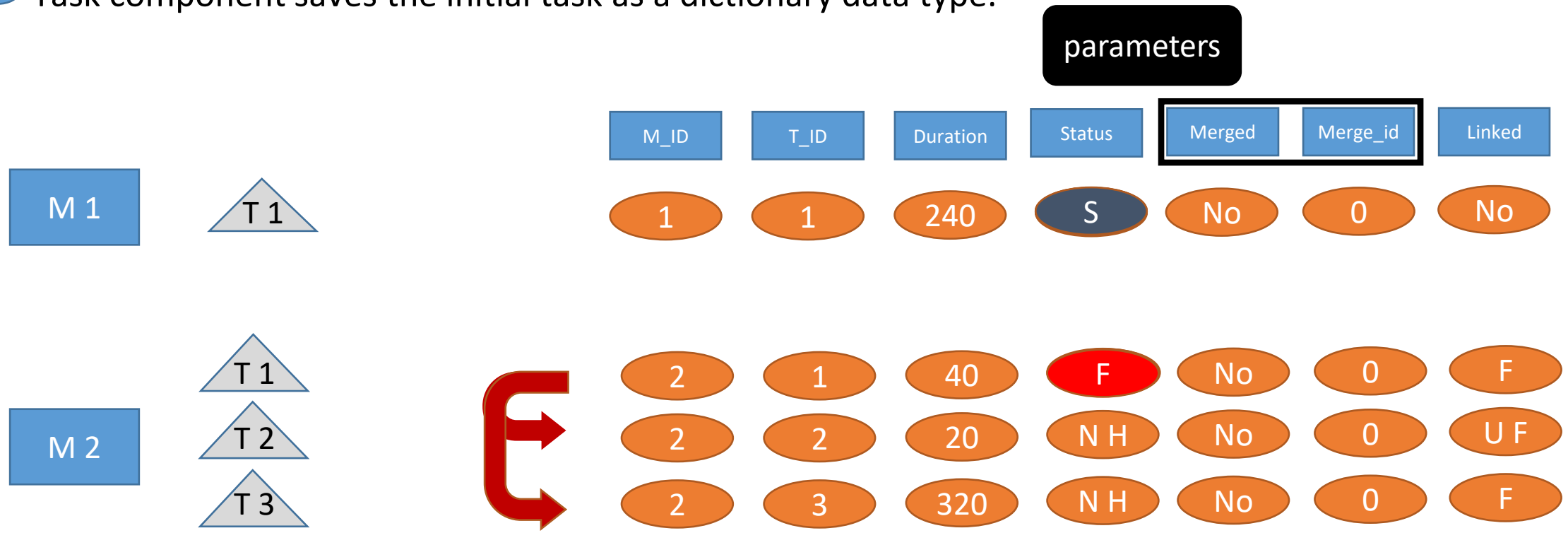
Design and Implementation – Task Component



- Primitive tasks are stacked sequentially in the queue.
- High-level missions could be divided into multiple tasks.
- Task component manages the execution sequentially.
- A task can be requested either explicitly through HRI component or implicitly through an external system.
- The tokenized process or NLP splitted between edge and cloud relying on the google cloud services.

Design and Implementation – Task Component

- Task component saves the initial task as a dictionary data type.

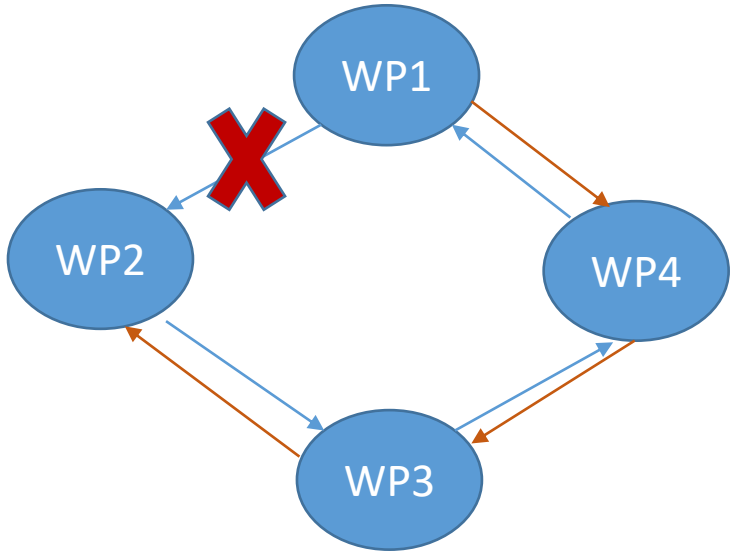


- With such approach, the system runs in a fully continuous operations.

High level Outline

Design and Implementation - ROSPlan

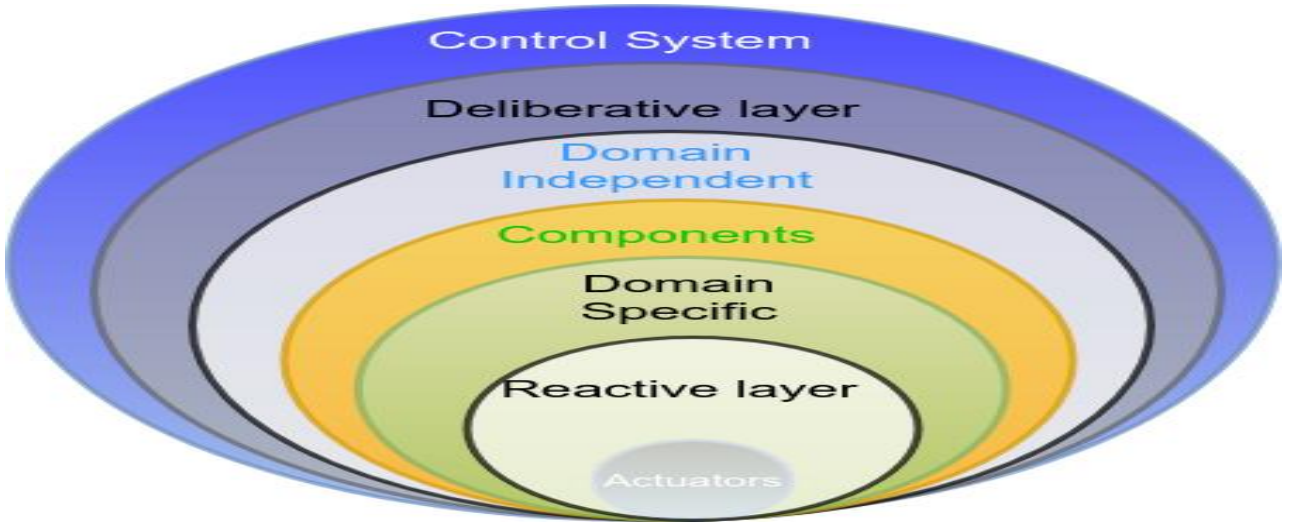
- Solving Problems during run time considering new situations and facts.



- Streamlined integration of planning and robotics approaches.
- Mitigating designing, development and implementation overheads.
- Dealing with the planning technique as black box or single component as the other components.
- Extending ROSPlan framework to cover most flaws and limitations.

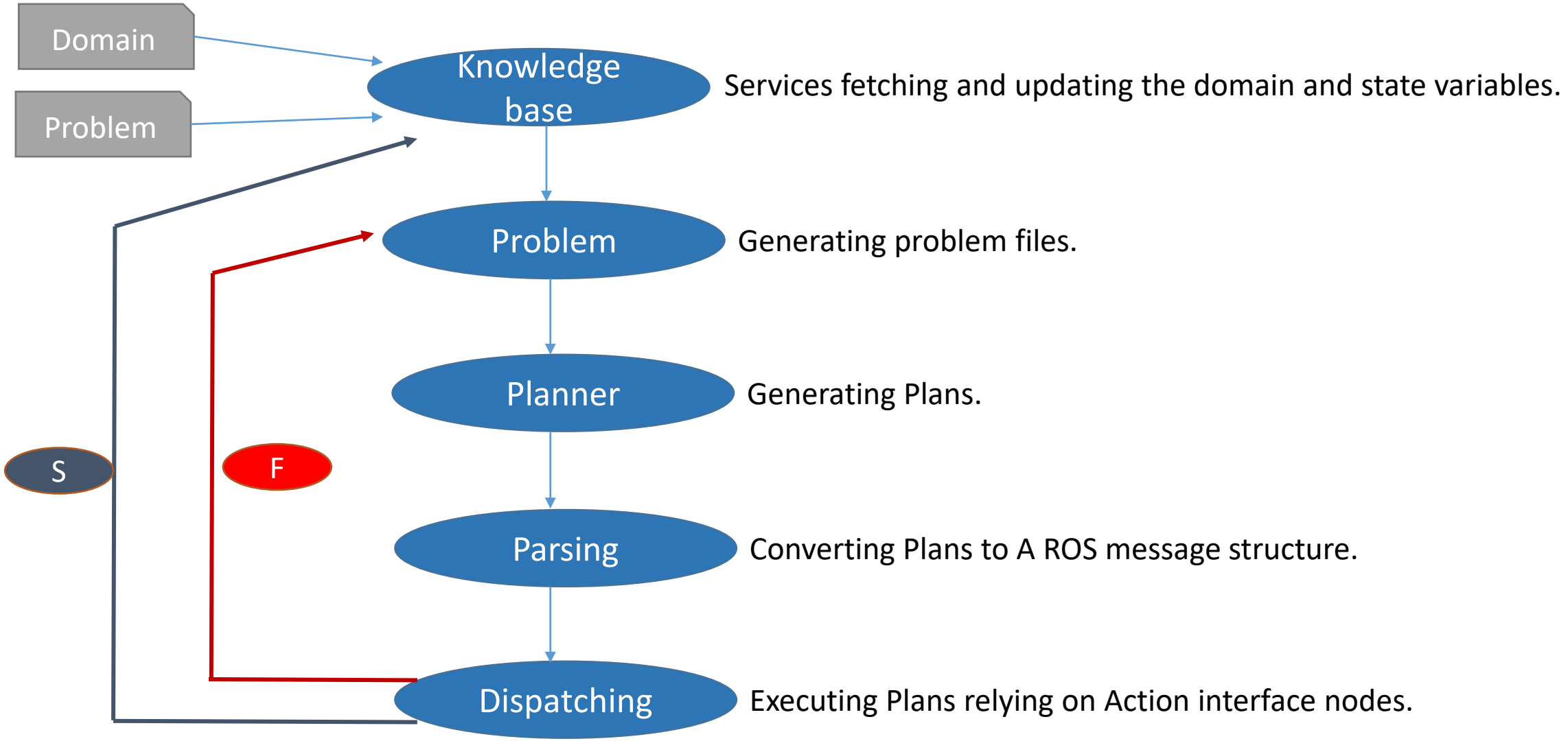
Design and Implementation - ROSPlan

- The objective is to plan missions to achieve efficiently the requested task based on the capabilities.



- The system relies on a planning and executing sub-systems that use several robotic components.
- The planning sub-system uses deterministic approach to decrease time, to facilitate domain modeling.
- The executing sub-system relies on several action interfaces that implement linkage mechanism.

Design and Implementation - ROSPlan



Design and Implementation - ROSPlan

(define (domain service_robot)
(:requirements :strips :typing :fluents :disjunctive-preconditions :durative-actions)
(:types Landmark robot student voicefile)

(:predicates
(**robot**
(**conne**
(**visite**
(**notifi**
(**detec**

(:funct

(:durat
:param
:durati
:condi
(at sta
(at start (robot_at ?k ?locfrom)))
:effect (and
(at end (visited ?locto))
(at start (not (robot_at ?k ?locfrom)))
(at end (robot_at ?k ?locto))))

Planning technique

- ADL can be used to test the planning domain, problem files, by generating plans using FF-Forward planner.
- Since actions in a robotic context are time-dependent, we use PDDL 2.2 that is capable of handling time constraints of our robotic domain model.
- The ROSPlan framework uses temporal planning technique relying on the temporal planner POPF.

(at start (visited ?loc))
(at start (robot_at ?k ?loc)))
:effect (and
(at end (detected_at ?loc ?p))
(at end (robot_at ?k ?loc)))))

Design and Implementation - ROSPlan

(define (**problem Alert-Students**)
(:domain service_robot)
(:objects
wp0 wp1 wp2 wp3 - landmark
ROB - robot
mark - student
filename - voicefile
)
(:init
(robot_at ROB wp2)
(connected wp0 wp0)

(= (distance wp0 wp0) 0)

)
(:goal (and
(notified wp0 filename)
(notified wp1 filename)
(notified wp2 filename)
(notified wp3 filename)
))

(define (**problem Detect-Student**)
(:domain service_robot)
(:objects
wp0 wp1 wp2 wp3 - landmark
ROB - robot
mark - student
filename - voicefile
)
(:init
(robot_at ROB wp0)

(:goal (and
(detected_at wp0 mark)
(detected_at wp1 mark)
(detected_at wp2 mark)
(detected_at wp3 mark)
))

Generated **plan** for **problem Alert-Students**

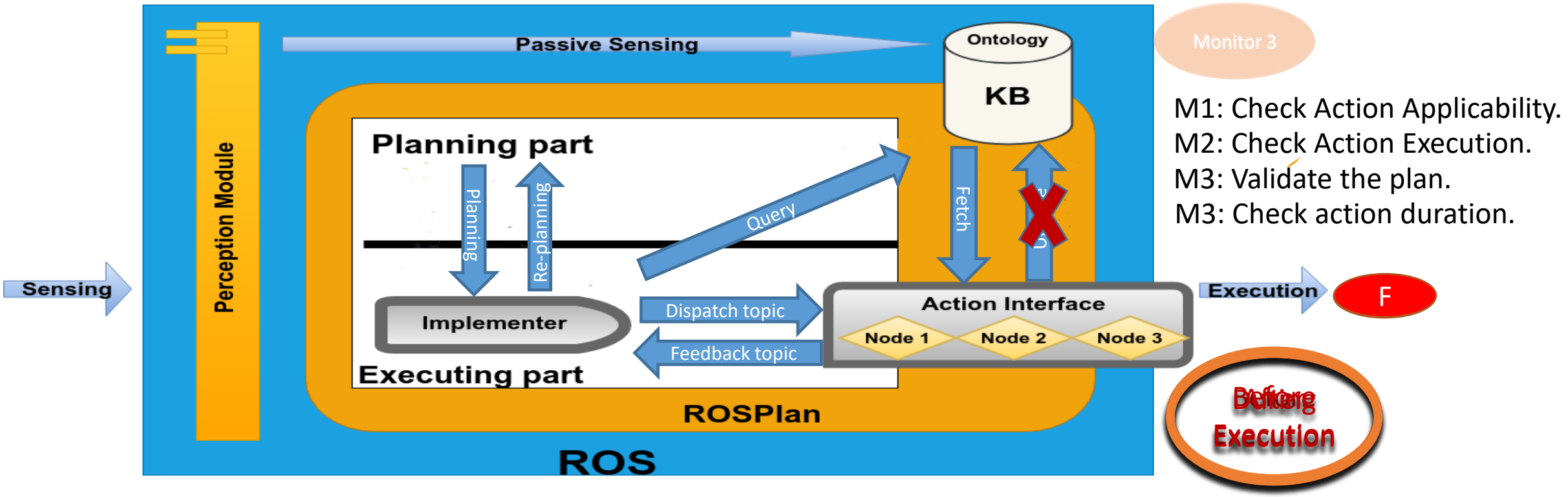
0.000: (goto_landmark ROB wp2 wp2) [10.000]
10.001: (notify_landmark ROB wp2 filename) [20.000]
30.002: (goto_landmark ROB wp2 wp1) [10.000]
40.003: (notify_landmark ROB wp1 filename) [20.000]
60.004: (goto_landmark ROB wp1 wp3) [10.000]
70.005: (notify_landmark ROB wp3 filename) [20.000]
90.006: (goto_landmark ROB wp3 wp0) [10.000]
100.007: (notify_landmark ROB wp0 filename) [20.000]

Merged problem

(:goal (and
 (forall (?w - landmark) (detected_at ?w mark)   )
 (forall (?w - landmark) (notified ?w filename)))
))

Design and Implementation - ROSPlan

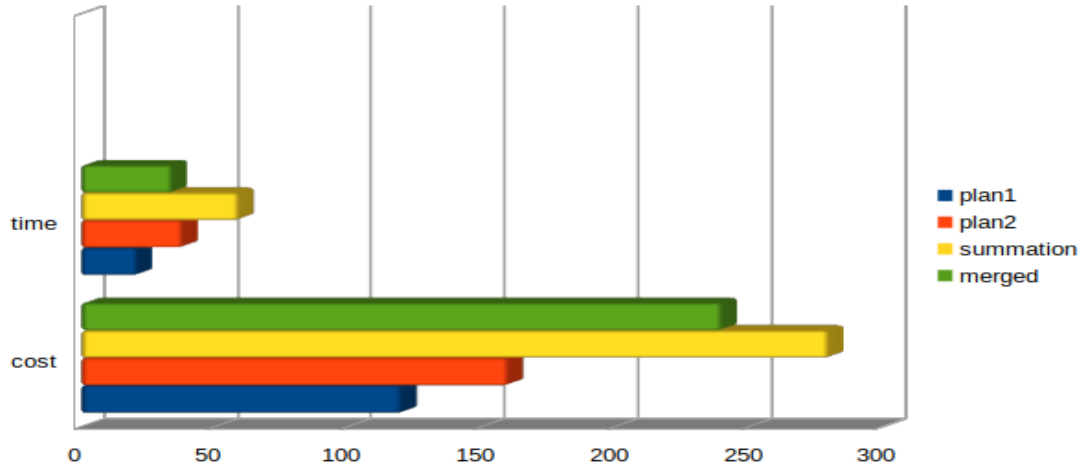
- The implementer controls the execution, respects Action duration, informs Task component.



- The system implements a closed-loop: planning (re-planning), monitoring, and execution phases.
- The implementer can trigger re-planning based on a validation process (pre-processing step) .

Design and Implementation - ROSPlan

- The loop is broken during any failure triggering re-planning. (controller's duration-time, planned action-time, ROS failure message, action applicability, plan validity, execution error.)
- System can implement the partial and receding-horizon approach. Since, mission can be decomposed into several tasks, A sub-goaling technique based on locations can also be implemented through a Scattering algorithm that separates the goals based on its locality.
- Since the problem generator generates a new problem file included the remaining goals and the facts (new states), planner generates plan considering the available facts, and can advise the plan at run-time.
- The planning process varies greatly on the edge or the cloud. Task component calls only 2 ROSPlan services after publishing the plan.



High level Outline

1. Introduction

2. State of the Art

3. Design and Implementation

    3.1. HS-RFA

    3.2. Task Component

    3.3. Planning and Executing Component

4. Conclusion

Conclusion

- A robotic system achieves diverse set of tasks autonomously and continuously. Robot has a goal directed behavior and a human-like capabilities.
- The system provides continuous operations through implementing Task component and provides autonomous functionality through extending ROSPlan framework as a planning and executing framework.
- KB stores symbolic representation, POPF2 generates a temporal plan considering Action duration, distance between locations and paths available. Parsing node converts the plan into a valid structure of ROS messages.
- A main procedure handles the action execution processes in which actions had a related interface node to link with the lower controllers.
- The deliberation and planning techniques take place at two levels. The independent domain planning is executed at the top level while the domain specific is implemented at the lower.
- A closed loop consisting of the planning, monitoring, and execution phases, can be broken, as the re-planning phase is triggered.
- Dynamic planning concepts such as receding-horizon and partial planning strategies could be implemented easily.

Thanks

Thank you for your attention!

Do you have any Questions?

SYNASC2021: 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing