Software Model for Robot Programming and Example of Implementation for Navigation System

Samira Chaychi
Department of Computer ScienceUniversity of Luxembourg
Esch-sur-Alzette, Luxembourg
samira.chaychi@uni.lu

Denis Zampunieris
Department of Computer ScienceUniversity of Luxembourg
Esch-sur-Alzette, Luxembourg
denis.zampunieris@uni.lu

Sandro Reis
Department of Computer Science University of Luxembourg
Esch-sur-Alzette, Luxembourg
sandro.reis@uni.lu

***Abstract*—In this paper, we are going to consider a current challenge in a robotic software system. We consider a problem, which is the lack of separation of concerns in robotic systems, and propose a software model to address the problem and resolve the current challenges. The core purpose of this paper is to demonstrate the advantages of using separation of concerns principles to create a well-ordered model of independent components that address separated concerns individually. Considering the problem, we developed a software model with the help of a proactive engine to address the challenges. We use robotic operating systems to help us to implement the robot simulator.**

Keywords—software design, separation of concerns, proac- tive computing, navigation

I. Introduction

From a programmer's perspective, the current robotic systems and software applications do not offer sufficient softwaredevelopment methodology. Most of the applications are not flexible to change, reuse, or maintain, which are the most critical points in software systems. So to tackle these problems, Separation of Concerns (SoC) [4] could help the softwaresystems.

SoC is a methodology to separate computer programs into distinct sections. In this methodology, each section addresses a separate concern: a set of information concerning a computer program's code. Applying the separation of concerns in a code gives more levels of freedom for some aspects of the program's purpose for simplification and maintenance of code. A program that integrates SoC well is called a modular program [7]. When concerns are separated, there are more opportunities for a module to upgrade, reuse, and develop independently [4]. The main purpose of this paper is to demonstrate the advantages of using separation of concerns principles in resolving the current challenges in a robotic software system. The purpose of the SoC principle is to permit the creation of a well-ordered model of independent components which are addressing a separate concern. Therefore, to have a software model considering the SoC principle, we propose a model for developing a robotic software system using a Proactive Engine (PE) [3].

The PE is the implementation of a rule-based proactive system. It includes the power of object-oriented principles andthe power of rule-based systems. The proactive engine consists of a rule engine, a database, and rules. It is a middleware system that can be attached to other systems either directlyor through a shared database. A combination of the proactive rules is called a scenario. A scenario is a set of rules where each rule is responsible only for a single action. Scenariosvary in features, structure, and complexity and can be applied in various areas and situations. The complexity of scenarios varies depending on the number of rules [8].

In this project, we use Robotic Operating Systems (ROS) tohelp us to implement the robot simulation. ROS is an open-source operating system containing message-passing between processes, package management, device control, etc. In addition, running the code through multiple computers is possible. The main goal of this operating system is the reusability of code in the robotic field [5] [10].

This introduction section is followed by the problem statement section, where we present the current challenges in the development of robotic software systems. Then, we move to the literature review section, where we are going to present a paper that developed a new navigation solution and another paper that addresses the lack of separation of concern within robotic systems. Next, we are going to explain our new proposed model, and then we will present the development of our model in the example of implementation section and advantages of our proposed model and at the end conclusion and future work.

II. Problem Statement

In this paper, the problem we will consider is the lack of separation of concerns in most robotic software systems.

Since the earliest of times, from the point of view of

978-1-6654-8920-1/23/$31.00 ©2023 IEEE

the researcher in software engineering, the main challenges are in terms of modifications. They try to design a software system that allows easy extension and/or modification of the code. However, Developing the software model leads to the complexity of the software project. This complexity is because of a large variety of dependencies and communication between different parts of the system. Therefore the software system is going to be large, complex, and even close to confusing pieces of code that are not easily optimizable. In addition, many otherpossible problems can occur due to the interaction between different modules. To address these problems, developers mustbreak down the project into independent tasks.

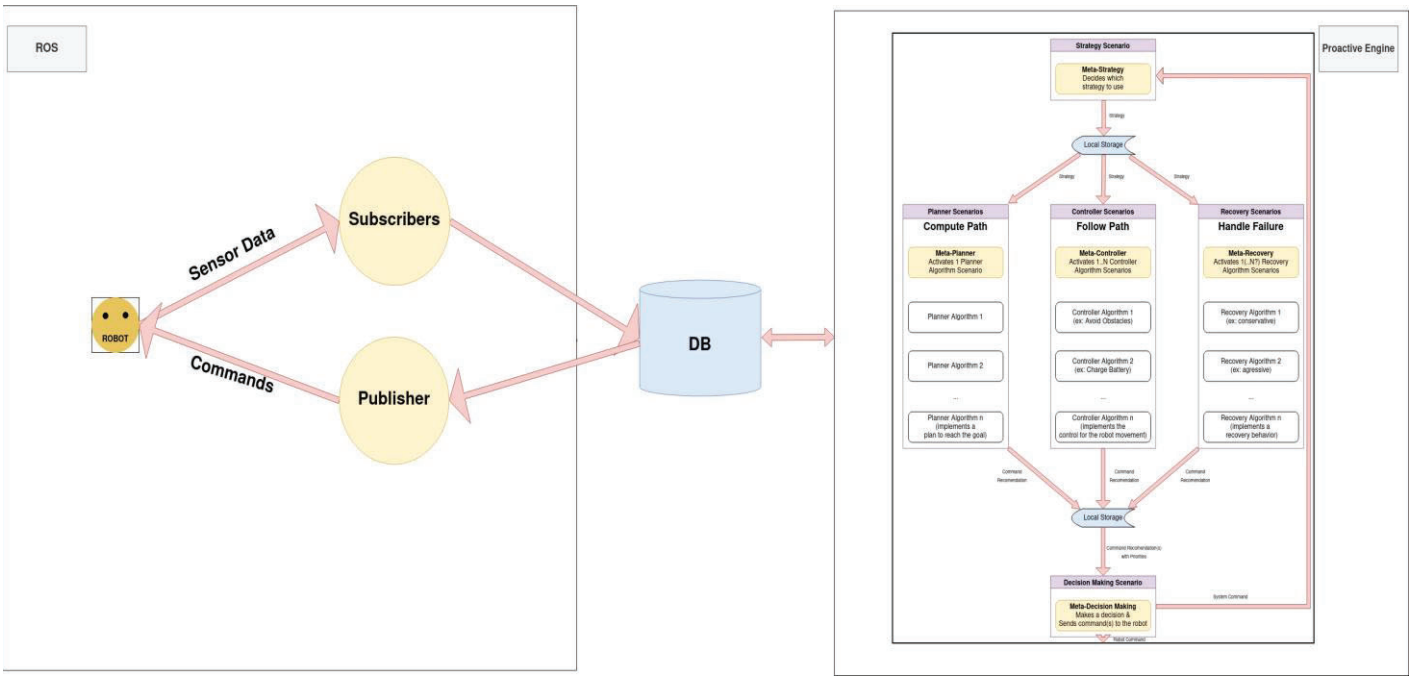

Fig. 1. Proposed Model

Reusability of software allows the developer to use the existing pieces of a software system to develop a new system which helps to reduce the time and effort to create a new one rather than programming a complete software system from scratch [9]. Therefore, modules in software systems should be independent of each other in order to have a reusable system. Most of the software system requires maintenance at some phase in the project. Another common challenge in a software system is maintaining the code during the development of the code or later. Most of the projects need to be updated during the developing time and need to fix bugs after some years. Therefore, maintaining and updating the code should not be ignored in order to have a better software system. To address these problems, developers need to have separate tasks to maintain a system.

Therefore, considering all the points, separation of concerns is a key objective in the development of a software system. In this paper, the main objective that we will tackle is the lack of separation of concerns by using PE, which uses proactive scenarios that allow separating concerns.

III. Literature Review

Researchers introduced a new navigation solution, Navigation2 [2], which builds on the prosperous heritage of ROS navigation. This project uses a behavior tree for arranging and managing new methods and tasks for having dynamic environments, which can apply to a wide variety of sensors. They proposed a new, fully open-source navigation system called Navigation2. Navigation2 uses a configurable behavior tree to arrange and manage three main navigation tasks: Planner, Controller, and Recovery [2]. At some point, the core members of developers of the Navigation2 project wanted to extend the design to fulfill a specific goal. They extended the design in several steps and analyzed each one, but in the end, they reached a point where the new design not only did not fulfill the objectives but also did it with a complex design, so they decided to stay in the current design [6]. In our paper, we are going to show that we can extend the project without having extra complexity.

Researchers at the University of Luxembourg introduced a possible new model for designing and implementing software in robotic systems. To address the lack of separation of concerns, they used PE to allow them to use proactive behavior and rule-driven programming to define proactive scenarios to have a better separation of concerns in the robotic system. Proactive scenarios are sets of condition-action rules. The researchers in this project managed to move the functionalities from the robot side toward the proactive engine such thatthey could implement them in separate

scenarios. Therefore, they reach the objectives of having separate concerns [1].We would like to implement the same concept in a ROSframework, which is a great simulation tool because the real robot's and the simulation's outcome are pretty much the samein this framework, and the code used for the simulation canbe transferred to a real robot.

IV. Proposed Model

In this paper, we propose a software model for navigation that not only fulfills the objectives but also has better separation of concerns. For the development of the robot software system, we used Proactive Engine. The PE is the implementation of a rule-based proactive system. It includes the power of object-oriented principles and the power of rule- based systems. The proactive engine consists of a rule engine, a database, and rules. It is a middleware system that can be attached to other systems either directly or through a shared database. In this proposed model, we consider each objective one scenario to have a better separation of concerns. ThePE has several scenarios; a scenario is a set of conditionaction rules. They are running in parallel, and each scenariois not aware of the existence of the other scenarios [1]. In our proposed model, we are going to have a connection between ROS and PE through a database. As you see our design in figure (1), data from ROS will be sent to the database, and the PE will use the requested data based on its needs, in figure (2),you will see the detailed design of PE. Our PE design consists of several kinds of scenarios: Strategy, Planners, Controllers, Recoveries, and Decision Making (DM).

A. Strategy Scenario

The strategy scenario is in charge of selecting a planned strategy based on some conditions and rules from the environment or input from the user. The selected strategy will be stored in the local storage, and other scenarios like controller, planer, and recovery will access it to activate the related scenarios. The strategy scenario has several different strategiesto control the robot's behavior; by selecting a different strategy,we will have a different behavior without changing any codein the system. Different strategies can apply to the system at runtime without having to relaunch the system.

B. Planner

The Planner module is in charge of computing the path and has a meta-planner scenario that reads the planned strategy from local storage and activates the corresponding scenario. There is a possibility of having several algorithms to im-plement the Planner -we will have one scenario for each algorithm- but we expect to activate only one Planner scenarioat a time.

C. Controller

The Controller module is in charge of controlling the robot's movement and reacting to the environment. In our implementation Controller has a meta-controller scenario that reads the planned strategy from local storage and activates the corresponding scenarios, depending on the expected behavior of the robot. According to the strategy, there is the possibility of activating several Controller scenarios at a time. In [11] researchers designed a nonlinear MPC Controller for a navigation system and compared it with the TEB controller which is based on a method called Timed Elastic Band, and this isthe same method that navigation2 used. Also, we would like tohave a comparison with the navigation controller in our next paper.

D. Recovery

The Recovery module is in charge of handling failure and has a meta-recovery scenario to activate the corresponding scenarios in the Recovery. We can have several scenarios basedon the robot's expected behavior that we want in a system. One scenario for each recovery behavior will be activated ifit corresponds to the chosen strategy.

E. Decision Making

The DM will receive data from the Controller, Planner, and Recovery modules. The DM receives recommendation commands (with a priority) from those types of scenarios, decides on a final command, and sends it via the database for the robot to perform. In our software system, the other scenarios do not know about the existence of the DM scenario. Each scenario makes its own decisions by making a command recommendation and then sending its result to the DM scenario indirectly via the database. Each command recommendation has its own priority assigned at creation time, given by the scenario that creates it. One of the advantages of assigning a priority to the scenario then is that when we add morescenarios to the system, we do not need to adapt the DM.It will work seamlessly with the new scenarios and command recommendations they create.

V. Implementation

In our implementation, as shown in figure (1) we have a connection between ROS and PE through a MySQL database. As you see in the design of our implementation in the ROS part, we have subscriber and publisher nodes. Subscriber nodesreceive the required data from the robot and will write it inthe database, and the publisher node will read data from the database and send it to the robot to perform.

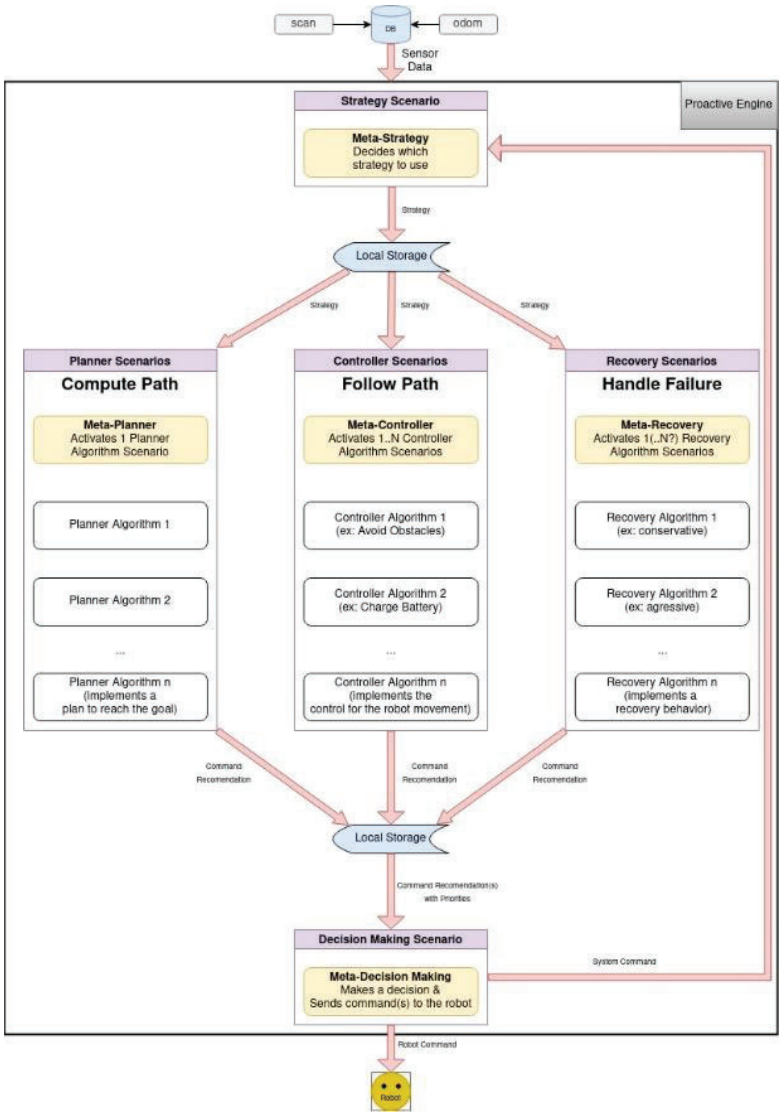

Fig. 2. Navigation model with PE

As you see the detailed design of PE in figure (2), all scenarios are running in parallel and independent, and they are not aware of existing of each other to communicate. We consider the localization robot that aims to reach the goal point and avoid obstacles. In our PE implementation, the strategy scenario, as you see in the proposed model, has several pre-defined strategies to control the robot's behavior in the strategy scenario to choose from. For instance, one strategy can be, go to the goal point and avoid obstacles, and the other couldbe, go to the goal point and do not consider obstacles. Also, another strategy could be checking the robot's battery level and changing the robot's direction to the charging station, if needed. All these different behaviors can change at runtime without relaunching the system. For example, if the battery level is low, the robot will change the direction to go to the battery station and behave differently. Conversely, if the robot is not considering obstacles at runtime, the strategy can easily be changed to have a robot that considers obstacles in the environment. Our system can choose a different scenario basedon the system's conditions and rules. The data these conditionsneed can come from the environment or input from the user. As you see in figure (2) feedback loop shows that the decision-making scenario can send a command to the strategy scenario to change the planned strategy at a run time.

The strategy scenario will store the planned strategy in local storage, and the meta scenarios can access this data. Therefore, the corresponding scenarios from the Planner and Controller will be activated by the meta scenarios of the Planner and Controller. All active scenarios are running in parallel and independent of each other and making a commandrecommendation based on their own algorithm.

The main goal of the Planner module is to compute thepath based on the start and goal points and go to the goalpoint. In our implementation, the scenario "turn&move" will be activated, and the robot will try to reach the goal point. There is a possibility of having several algorithms for thePlanner; then, we could have other behaviors like; first,

turn, and then move forward. We expect to activate only one Plannerscenario at a time, and the scenario that is activated is the one that corresponds to the strategy chosen.

The main task of the Controller module is to control the robot's movement. In our implementation, we have "avoid&move to the left" and "avoid&move to the right";also "battery level checking". The chosen scenarios could be changed at run time based on some conditions.

The Recovery module, like Planner and Controller, has a meta-recovery scenario to activate the corresponding scenarios, we are working to implement some recovery scenarios as well, but for the moment, we have not implemented any yet. Finally, we have the DM scenario; this scenario reads recommendation commands and priority levels created by the Controller, Planner, and Recovery scenarios and will make the final decision based on the priority and type of eachactive scenario. In the end, the DM will send the finalcommand to the robot to perform. In our implementation, we assigned priority for all the scenarios in Planner, Controller, and Recovery at creation time. So we can easily add more scenarios to Planner, Controller, and Recovery without having to adapt the DM scenario.

VI. Advantages

In our implementation, we consider each objective one scenario; we break down all the pieces into independent tasks without interacting with each other. They are runningin parallel, and each scenario is not aware of the existenceof other scenarios. Therefore, our implementation has better separation of concerns [1].

Another challenge that we consider in this paper is extending the code, making it more complex. In our implementation, we can add different algorithms for different types of robots and platforms without changing the current code or needing extra settings and configuration. Therefore, extending the code in our implementation will not make it more complex. For example, in our example implementation, we have less complexity in a system for computing the path and controllingthe robot since there are separate scenarios for each, and we can extend them easily.

Also, our implementation has different rules and strategies that can apply to the robot's behavior at runtime without relaunching the system again. As we mentioned in the implementation of PE, the advantage of having a feedback loop makes it possible to choose a different strategy to have a completely different behavior without changing any piece of code in the system at runtime.

VII. Conclusions and Future Work

We considered a problem: the lack of separation of concerns in robotic systems and, more specifically, in the navigation project. We proposed a software model to address the prob- lem and resolve the current challenges in developing robotic software systems. With the help of the proactive engine, the proposed software system has both powers of object-oriented principles and rule-based systems. For easy programming of each scenario, a scenario is allocated to only one objectiveon the robot. Therefore, we have a software system that integrates the separation of concerns well. We presented a basic implementation of the entire design and will add the recovery behaviour and complete it soon. In future work, we will compare our system's performance with other existing systems like [2] using software metrics both at compile time and runtime.

References

[1] A. Frantz, D. Zampunieris, "Separation of Concerns Within Robotic Systems Through Proactive Computing," 2020 Fourth IEEE Interna- tional Conference on Robotic Computing (IRC), 2020, pp. 197-201, doi:10.1109/IRC.2020.00039

[2] S. Macenski, F. Mart´ın, R. White, J. Clavero. The Marathon 2: ANavigation System. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[3] D. Zampunieris, "Implementation of efficient proactive computing us- ing lazy evaluation in a learning management system (extended ver- sion),"IGI Publishing, vol. 3, no. ISBN 0863411711, pp. 103–109, 2008.

[4] https://en.wikipedia.org/wiki/Separation of concerns

[5] Stanford Artificial Intelligence Laboratory et al. (2018). Robotic Oper- ating System. Retrieved from https://www.ros.org

[6] https://github.com/ros-planning/navigation2/issues/565

[7] P. Laplante, "What every engineer should know about software engi- neering, "CRC Press, pp. ISBN 978–0 849 372 285, 2007.

[8] D. Shirnin, S. Reis, D. Zampunieris, "Experimentation of proactive computing in context aware systems: Case study of human-computer interactions in e-learning environment," 2013 IEEE Interna- tional Multi-Disciplinary Conference on Cognitive Methods in Situa- tion Awareness and Decision Support (CogSIMA), 2013, pp. 269-276, doi:10.1109/CogSIMA.2013.6523857

[9] Moko, Anasuodei & Ojekudo, & Akpofure, Nathaniel. (2021). Software Reusability: Approaches and Challenges. International Jour-nal of Research and Innovation in Applied Science. 06. 142-146. 10.51584/IJRIAS.2021.6510.

[10] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open source software, 2009, vol. 3, no. 3.2: Kobe, Japan.

[11] Quang, Hiep Do et al. "Design a Nonlinear MPC Controller for Autonomous Mobile Robot Navigation System Based on ROS." Interna- tional Journal of Mechanical Engineering and Robotics Research (2022).