PORTFOLIO COURSEWORK

Name: Pavan sunder

Student Number: 20060380

Course Code: 7COM1084

Introduction to research specialism

The specialism selected on the paper is “Robotics Software Engineering: A Perspective from the Service Robotics Domain” which is related to the software engineering specialism (Sergio García.). The paper mainly focuses on the significance of software engineering and the paper mainly focuses on the service robotics The functionalities require complex software that can be difficult to design, implement, and maintain. As a result, SE practices are becoming increasingly important for the design and development of service robots. SE can help improve the quality and reliability of robotic software, as well as make it easier to evolve and maintain over time. By applying SE techniques, such as modularization, abstraction, separation of concerns, and testing, roboticists can create software that is more robust, reliable, and maintainable. Additionally, SE can help bridge the gap between different domains of expertise involves in the turn of events of service robot, such as robotics, computer science, and mechanical engineering. Ultimately, the draw on of SE practices in the event of service robots can help drive the growth and enactment of these technologies in a various field. Software engineering practices can help mitigate the risks associated with the complexity of service robots by giving a structured and systematic approach to the design and development of their software. By applying SE techniques, such as modularization, abstraction, and separation of concerns, roboticists can better manage the complexity of their systems and create software that is more robust, reliable, and maintainable. Additionally, SE can help improve communication and collaboration among the various domains of expertise involved in the development of service robots, such as robotics, computer science, and mechanical engineering. This can help reduce the likelihood of costly errors and improve the overall quality and reliability of service robots. Ultimately, the usage of SE practices in the event of service robots can help drive the growth and acceptance of these technology in a various fields.

Open research question

what are the protocols that must be satisfied by the software engineer while developing service robot?

Software engineers working on the development of service robots should follow a certain set of best practices and protocols to ensure the quality and reliability of their software. Some examples of these protocols include:

Adhering to industry standards and guidelines, such as ISO/IEC 12207 and ISO/IEC 15288, which provide frameworks for the development and lifecycle management of software in robotics and other CPS domains. Using modular and scalable architecture designs, such as the ROS (Robot Operating System) architecture, which can help improve the maintainability, evolvability, and scalability of the software. Applying design patterns and design principles, such as the SOLID principles and the Singleton pattern, which can help improve the modularity, maintainability, and reusability of the software. Implementing robust and reliable error handling mechanisms, such as try-catch blocks and exception handling, which can help improve the reliability and resilience of the software. Conducting thorough testing, verification, and validation of the software, using a variety of testing techniques and tools, such as unit testing, integration testing, and simulation, to ensure that it meets its functional and non-functional requirements. Collaborating closely with other domains of expertise, such as robotics, computer science, and mechanical engineering, to ensure that the software integrates seamlessly with the hardware and environmental components of the robotic system.

Some Software engineering practices that can be applied in the development of service robots include:

1.Modularization: Dividing the software into smaller, independent modules that can be developed and tested separately, and then integrated into a larger system. This can improve the maintainability and evolvability of the software.

2.Abstraction: Separating the functional aspects of the software from its implementation details, using high-level interfaces and abstract data types. This can help improve the understandability and modularity of the software.

3.Separation of concerns: Dividing the software into distinct, independent parts, each addressing a specific concern or aspect of the system. This can help improve the modularity, maintainability, and evolvability of the software.

4.Testing: Verifying that the software meets its functional and non-functional requirements through a variety of testing techniques, such as unit testing,

integration testing, and system testing. This can help improve the reliability and robustness of the software.

5.Collaborative development: Involving multiple stakeholders and domains of expertise in the development of the software, using collaboration tools and agile methodologies. This can help improve communication and coordination among the different teams involved in the project.

In addition to following best practices and protocols, software engineers working on the development of service robots should also prioritize continuous learning and improvement. This can involve staying up to date with the latest developments and trends in SE and robotics, attending conferences and workshops, and participating in professional development programs. By continuously learning and improving their skills and knowledge, software engineers can better understand the challenges and opportunities in the field of service robotics and apply this knowledge to develop high-quality, reliable software. Additionally, by staying abreast of the latest developments in SE and robotics, software engineers can help drive innovation and advance the field of service robotics. Ultimately, by prioritizing continuous learning and improvement, software engineers can play a critical role in the growth and success of the service robotics industry.

Existing and related work

the challenge face related to Software Engineering for service robots?

One of the key challenges in Software Engineering for service robots is the need for robust and adaptable software that can operate effectively in complex and dynamic environments. Service robots typically operate in partially constrained environments, often populated by humans, and require a blend of hardware and software components to provide a wide range of services. This complexity and diversity of requirements make it difficult to develop software that is reliable and adaptable. Another challenge is the lack of consolidation and standardization in the field of Software Engineering for service robots, which can make it difficult for developers to access and apply best practices and guidelines. Additionally, the need for interdisciplinary expertise and collaboration can present challenges in coordinating the efforts of different domains of expertise. Overall, these challenges highlight the need for effective approaches and solutions for improving the development of service robotics software.

García, S., One of the key challenges identified in this research is the need for robust and adaptable software for service robots, which can operate effectively in complex and dynamic environments. This is particularly important for service robots that interact with humans and other objects, as any failure or malfunction can have serious consequences. The solutions developed in this research aim to address this challenge by providing modular and scalable software architecture designs, which allow for easy customization and integration of different functionalities. The use of domain-specific languages also allows for the easy specification of robot missions and behaviour, reducing the complexity of programming and improving the reliability of the system. The plan synthesis component enables the robotic system to generate plans and behaviours based on incomplete or uncertain information, improving its adaptability and flexibility. Overall, these solutions are designed to improve the robustness and adaptability of service robotics software, and to support the development of reliable and effective service robots. To address these challenges, the research described in this paragraph proposes the use of model-driven engineering and domain-specific languages to improve the development of service robotics software. These approaches allow for the creation of modular, scalable, and customizable software architectures, which can be easily adapted to different contexts and environments. The use of domain-specific languages also simplifies the specification of robot missions and behaviour, reducing the complexity of programming and increasing the reliability of the system. Additionally, the integration of these solutions into an industry-validated robotic framework allows

for the validation and evaluation of their effectiveness in real-world scenarios. Overall, this research provides a promising approach for addressing the challenges of engineering robust and autonomous service robots.

Brugali et al,. Software reuse is a key technology for improving software productivity and quality. It involves the use of existing software to construct new software, and can take the form of opportunistic reuse, where software engineers reuse pieces of software that fit the current problem, or systematic reuse, where teams invest in developing software building blocks that can be used in a variety of similar applications. Systematic software reuse was introduced with the development of libraries in the COBOL programming language but has not been as successful as anticipated due to the difficulty of encapsulating high-level functionality in subroutines. Instead, object-oriented programming and component-based software engineering have become the dominant paradigms for achieving systematic reuse in software development. These approaches allow for the modularization and encapsulation of functionality, making it easier to reuse and integrate different software components. Overall, software reuse is a critical technology for improving the productivity and quality of software development and continues to be an active area of research and development in the field of software engineering. The use of component-based software engineering (CBSE) and model-driven engineering (MDE) in robotics can help address some of the challenges that arise in the development of robotic systems, such as the need to cope with hardware and software heterogeneity and the complexity of deploying and controlling these systems in real-world environments. By using CBSE and MDE, developers can create modular, reusable components that can be easily combined and adapted to different scenarios, and use models to design, simulate, and test the behaviour of these systems before deploying them.

The use of domain-specific languages (DSLs) in robotics can further support the development of robotic systems by providing domain-specific languages that can be used to express the requirements, design, and control of these systems in a concise, abstract, and formal way. This can help reduce the complexity of coding and debugging and enable domain experts and software engineers to work together more effectively. However, the use of CBSE, MDE, and DSLs in robotics also poses some challenges that need to be addressed. For example, defining DSLs for complex domains can be difficult, and there may be a need for effective tool support for implementing and using these languages. Additionally, integrating different components and models into a cohesive and efficient system can be a challenge, and there may be a need for frameworks and standards to support the interoperability and scalability of these systems.

Overall, the use of CBSE, MDE, and DSLs in robotics is an important area of research that can help improve the development, deployment, and control of robotic systems. By addressing the challenges and leveraging the advantages of these approaches, researchers and developers can create more efficient, effective, and reliable robotic systems for a wide range of applications.

Research approach

Robotics software engineering is a rapidly growing field that focuses on the design and development of software systems for robots. In the service robotics domain, this often involves creating algorithms and control systems that enable robots to perform a wide range of tasks, such as navigation, manipulation, and object recognition. software engineering in the service robotics domain is to focus on developing modular, reusable software components that can be easily integrated into a variety of different robotic platforms. This approach allows for greater flexibility and adaptability in the design and implementation of robotic systems, and enables developers to easily reuse existing components and build new ones that can be integrated into larger systems. Another key aspect of robotics software engineering in the service robotics domain is the development of robust, reliable control systems that can handle the challenges and uncertainties inherent in real-world environments. This may involve the use of advanced control algorithms, such as model predictive control and adaptive control, to enable robots to respond to changing conditions and adapt their behaviour accordingly. In addition, robotics software engineers may also focus on developing user-friendly interfaces and tools that allow non-experts to easily program and control robots, which can help to broaden the potential applications of service robots and make them more accessible to a wider range of users. In addition to these core software engineering practices, robotics software engineers in the service robotics domain may also focus on developing machine learning and artificial intelligence algorithms that enable robots to learn from data and improve their performance over time. This may include techniques such as supervised learning, unsupervised learning, and reinforcement learning, which can enable robots to learn from examples, adapt to changing conditions, and make decisions based on their goals and objectives. Another important aspect of robotics software engineering in the service robotics domain is the development of safety and security mechanisms to protect against potential hazards and malicious attacks. This may involve the use of safety-critical software engineering practices, such as fault tolerance, redundancy, and certification, to ensure that robots can operate safely and securely in real-world environments. Overall, the goal of robotics software engineering in the service robotics domain is to create powerful, versatile, and easy-to-use robotic systems that can assist humans in a variety of tasks and environments. To achieve this goal, robotics software engineers must have a deep understanding of software engineering principles and practices, as well as a strong background in control theory, artificial intelligence, and human-robot interaction. By applying these principles and practices, robotics software engineers can help to create the next generation

of service robots that are capable of assisting humans in a wide range of tasks and environments. Service robots are becoming increasingly common in a variety of settings, such as hospitals, warehouses, and homes. They are capable of performing a wide range of tasks, including navigation, manipulation, and object recognition. Robotics software engineering plays a crucial role in the development of these robots, as it involves creating algorithms and control systems that enable robots to perform their tasks effectively and efficiently. Robotics software engineers in the service robotics domain often focus on developing modular, reusable software components that can be easily integrated into different robotic platforms. This allows for greater flexibility and adaptability in the design and implementation of robotic systems. In addition, robotics software engineers may also focus on developing user-friendly interfaces and tools that allow non-experts to easily program and control robots, which can help to broaden the potential applications of service robots and make them more accessible to a wider range of users. Another key aspect of robotics software engineering in the service robotics domain is the development of robust, reliable control systems that can handle the challenges and uncertainties inherent in real-world environments. This may involve the use of advanced control algorithms, such as model predictive control and adaptive control, to enable robots to respond to changing conditions.

Personal investment

This paper aims to provide a comprehensive overview of the practices and challenges in robotics software engineering, with a focus on the service robotics domain. The paper discusses the key software engineering principles and practices that are commonly applied in the development of service robots, including modularization, abstraction, separation of concerns, testing, collaboration, and continuous integration and deployment.

The paper also explores the use of advanced software engineering techniques, such as model-based development, reusability and component-based development, dependency management, and documentation and technical communication. Additionally, the paper discusses the role of machine learning and artificial intelligence in robotics software engineering, as well as the importance of safety and security in the development of service robots.

The paper also addresses the challenges and recurrent issues that are typically faced by robotics software engineers in the service robotics domain and discusses the solutions that have been adopted by industrial and academic practitioners. The paper concludes by discussing observations and proposed actions for researchers and practitioners in the field of robotics software engineering.

Service robots are an increasingly important and rapidly growing field within the broader domain of robotics. These robots are designed to assist humans in a variety of tasks and environments and have a wide range of applications in fields such as logistics, healthcare, telepresence, maintenance, domestic tasks, education, and entertainment.

Service robots are typically designed to be modular, flexible, and adaptable, allowing them to be easily integrated into different settings and environments. They are equipped with advanced sensors, control systems, and algorithms that enable them to navigate, manipulate objects, and interact with their surroundings.

Service robots are often equipped with machine learning and artificial intelligence algorithms that enable them to learn from data and improve their performance over time. This allows them to adapt to changing conditions and make decisions based on their goals and objectives.

Overall, the field of service robotics is a rapidly growing and exciting area of research and development, with many exciting potential applications and challenges for robotics software engineers. By applying the principles and practices of robotics software engineering, researchers and practitioners can help to create the next generation of service robots that are capable of assisting humans in a wide range of tasks and environments.

Reference

Hans Utz, Stefan Sablatnog, Stefan Enderle, and Gerhard Kraetzschmar. 2002. Miro-middleware for mobile robot applications. IEEE Transactions on Robotics and Automation 18, 4 (2002).

Saddek Bensalem, Lavindra de Silva, Félix Ingrand, and Rongjie Yany. 2011. A Verifiable and Correct-by-Construction Controller for Robot Functional Levels. Journal of Software Engineering for Robotics 2 (2011).

Davide Brugali, Luca Gherardi, A Biziak, Andrea Luzzana, and Alexey Zakharov. 2012. A reuse-oriented development process for component-based robotic systems. In SIMPAR. Springer.

Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. 2006. YARP: yet another robot platform. International Journal of Advanced Robotic Systems 3, 1 (2006), 8.

Juliet M Corbin and Anselm Strauss. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. Qualitative sociology 13, 1 (1990),.

García, S. et al. (2020) "Robotics software engineering: A perspective from the service robotics domain," Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.

Yentl Van Tendeloo and Hans Vangheluwe. 2017. The Modelverse: a tool for multi-paradigm modelling and simulation. In WSC. IEEE.

Brugali, D. (2020) "Software product line engineering for robotics," Software Engineering for Robotics.

Gherardi, L. and Brugali, D. (2014) "Modeling and reusing robotic software architectures: The Hyperflex toolchain," 2014 IEEE International Conference on Robotics and Automation (ICRA).

Maoz, S. and Ringert, J.O. (2018) "On the software engineering challenges of applying reactive synthesis to robotics," Proceedings of the 1st International Workshop on Robotics Software Engineering.

Wang, Y. (2007) "Principles of Software Engineering," Software Engineering Foundations.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. Experimentation in software engineering. Springer Science & Business Media.

Douglas C. Schmidt. 2006. Guest Editor’s Introduction: Model-Driven Engineering. Computer 39, 2 (2006).

Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. 2013. The BRICS Component Model: A Model-based Development Paradigm for Complex Robotics Software Systems. In SAC. ACM, 1758ś1764.

Herman Bruyninckx, Peter Soetens, and Bob Koninckx. 2003. The real-time motion control core of the Orocos project. In ICRA, Vol. 2. IEEE, 2766ś2771.

D. Brugali and P. Scandurra, "Component-based robotic engineering (Part I) [Tutorial]," in IEEE Robotics & Automation Magazine, vol. 16, no. 4, pp. 84-96, December 2009, doi: 10.1109/MRA.2009.934837.

Siciliano, B. and Khatib, O. (2016) Springer Handbook of Robotics. Berlin: Springer International Publishing.

Ramesh, S.V. et al. (2021) “Mitochondrial and chloroplast genomes,” The Coconut Genome, pp. 133–143.