Doctoral Dissertation

Human-Robot Interaction System for Non-Expert Users to Create and Debug Robot Behaviors Using Visual Programming

Pattaraporn Tulathum
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Kenichi Matsumoto
Software Engineering Laboratory (Division of Information Science)

Submitted on January 31, 2023

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of Engineering

Pattaraporn Tulathum

Thesis Committee:

Supervisor Kenichi Matsumoto
(Professor, Division of Information Science)
Takahiro Wada
(Professor, Division of Information Science)
Takashi Ishio
(Associate Professor, Division of Information Science)
Raula Gaikovina Kula
(Assistant Professor, Division of Information Science)
Tsukasa Ogasawara
(Vice President, Nara Institute of Science and Technology)
Gustavo Garcia
(Visiting Associate Professor, Nara Institute of Science and Technology)
Jun Takamatsu
(Senior Researcher, Microsoft)

Human-Robot Interaction System for Non-Expert Users to Create and Debug Robot Behaviors Using Visual Programming*

Pattaraporn Tulathum

Abstract

This dissertation proposes a robot behavior creator, a robot simulator, and debugging features to develop a Human-Robot Interaction system for non-expert users. As a problematic consequence of an aging society with an increasing labor shortage, there is a need for service robots to efficiently support work in many places, such as convenience stores. Especially for non-expert users (e.g., shop staff) who do not understand a robotic system, it is challenging to create desired robot behaviors. This requires a tool to enable non-expert users to create and fix issues in robot behavior programs.

This thesis provides a concept of a robot behavior program for non-expert users based on the human decision-making process with the implementation of a robot behavior creator and debugger. It consists of two features: (i) Behavior Trees implemented on a graphical user interface; and (ii) a robot simulator. Furthermore, this thesis provides a subjective system evaluation with non-expert users to show the effectiveness of the proposed system. The results with ten subjects show that non-expert users can create and fix robot behaviors based on the given tasks in the convenience store scenario. According to the System Usability Scale (SUS), the proposed system has a good usability level.

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, January 31, 2023.

The proposed system also provides four debugging features for the robot behavior programming: (i) breakpoints, (ii) node execution monitoring, (iii) execution log, and (iv) robot variables. These features allow non-expert users to identify and fix issues in the system. The experimental results show that 14 non-expert users could achieve all the assigned tasks. According to the SUS, the proposed system has a high marginal usability level. However, only one non-expert user could utilize the breakpoints. Based on the experimental results and interviews with non-expert users, the concept of breakpoints is not easy to understand and use for creating a program.

Overall, this dissertation demonstrates that the proposed methods enable non-experts users to create, edit and debug robot behavior programs through experiments with non-expert subjects.

Keywords:

Human-Robot Interaction (HRI), Robot Behaviors, Creating and Debugging System, Non-Expert Users, Behavior Trees

ロボット動作の作成とデバッグをビジュアルプログラミングにより行う非熟練ユーザ向けヒューマンロボットインタラクションシステム*

Pattaraporn Tulathum

内容梗概

本論文では、非熟練ユーザに適した人間とロボットのインタラクションシステムの構築に焦点を当て、ロボット動作作成ツール、ロボットシミュレータ、およびデバッグ機能を提案する。労働力不足を伴う高齢化社会の問題の到来により、作業を支援するサービスロボットが求められている。特にロボットシステムに理解を持たない非熟練者（店員など）に対しては、彼らが望むようなロボットの行動を作り上げることは非常に困難である。そのため、非熟練ユーザでもロボットの動作を作成・修正できるようなツールが必要となる。提案システムは、非熟練ユーザでもドラッグアンドドロップでロボット動作プログラムを作成できるロボット動作作成機能と模擬環境下で動作テストを行うことができるロボットシミュレータを提供する。これは、(i) グラフィカルユーザインタフェースを用いて実装されたビヘイビアツリーと、(ii) ロボットシミュレータによる可視化機能の 2 つから構成される。10 人の被験者による実験の結果、非熟練者でも与えられた状況の下でロボットの行動を作成・修正することが可能であることを示した。システムユーザビリティスケール (SUS) によれば、提案システムは良好なユーザビリティを有していることが分かった。また、提案システムは、ロボット動作プログラミングのために (i) ブレークポイント、(ii) ノード実行監視、(iii) 実行ログ、(iv) ロボット変数、の 4 つのデバッグ機能を提供する。これらの機能により、専門家でないユーザでもシステムの問題点を特定し、修正することができる。実験の結果、14 名の非熟練者のいずれもがブレークポイントを活用

*奈良先端科学技術大学院大学　先端科学技術研究科 博士論文, 2023 年 1 月 31 日.

できなかった。インタビューの結果、ブレークポイントの概念が理解しづらく、プログラム作成に利用しづらいことが原因であると考えられる。SUS によれば、提案システムは受け入れ可能なユーザビリティを有していることが分かった。本論文の主な貢献は以下の 4 つである。第一に、非熟練者向けのロボット動作プログラムの概念を提案した。第二に、コンビニエンスストアを模擬した環境に人のアバターとロボットを配置し、ロボット行動プログラムを作成しテストするロボット動作生成ツールを提供した。第三に、生成されたロボット行動プログラムの問題点を特定し、修正するための 4 つのデバッグ機能を搭載した。第四に、ロボットプログラミングの経験のない非熟練ユーザによる主観的なシステム評価を行った。

キーワード

ヒューマンロボットインタラクション (HRI), ロボットの行動を作成, デバッグするシステム, 非熟練者, ビヘイビアツリー

Acknowledgements

This dissertation is a task with many challenges and obstacles. I would like to express my gratitude to all the supporters and advisers who helped me along the way by kindly sharing their knowledge and advice.

I would like to express my deep gratitude to Professor Kenichi Matsumoto for giving me opportunities to study in his laboratory during last two years in the doctoral course. Moreover, thank you for providing a kind guidance and encouragement until I could achieve the PhD.

I would like to express my sincere gratitude to Doctor Tsukasa Ogasawara for giving me opportunities to study in his laboratory since the master's course. None of this would have been possible without his kindness, wisdom, and continued support, I could not complete this work and achieved the PhD.

I am very grateful to Associate Professor Takashi Ishio for sharing his advanced knowledge in software engineering field. His valuable comments always were a great motivation.

I am very grateful to Assistant Professor Raula Gaikovina Kula for his generous advice and sharing experiences. His suggestion could help me overcome many struggles related to research and daily life.

I am very grateful to Doctor Jun Takamatsu for sharing his advanced knowledge in robotics field. Your advice and comments are helpful and have given me a wider perspective and opportunities.

I am also grateful to Associate Professor Gustavo Garcia for your valuable suggestions and questions for new perspectives in research and daily life.

This thesis could not have been possible without the efforts of the rest of my thesis committee: Professor Kenichi Matsumoto, Professor Takahiro Wada, Associate Professor Takashi Ishio, Assistant Professor Raula Gaikovina Kula, Doctor Tsukasa Ogasawara, Associate Professor Gustavo Garcia, and Doctor Jun Takamatsu. I would like to thank you for your invaluable time spent reviewing my dissertation and providing comments and suggestions.

I would like to thank all members of the Software Engineering laboratory, the Robotics laboratory, and NAIST for being good friends, teaching me the Japanese

language and culture, and supporting me several times. It was a very touching moment, a great lesson, and unforgettable memories.

To my dear teammates in the NAIST-RITS-Panasonic team, thank you for sharing many advanced knowledge, interesting experiences, and wonderful moments since 2018. Especially Associate Professor Gustavo Garcia, Dr. Lotfi El Hafi, and Dr. Masaki Yamamoto, thank you for being good and respectful examples in both work and life.

To all my dear friends in Japan, especially thanks to Pedro Miguel Uriguen Eljuri and Tomoko Yui, who have always been my great friends and shared wonderful moments and experiences. Without you, my life in Japan would have been more suffering.

To all my dear friends in Thailand, especially Benjarut Sripetchdanon, Chutiphon Kongsompot, and Supapid Eknikom, thank you for always wishing each other good things even though we are far apart.

To my beloved boyfriend, Assistant Professor Bodin Chinthanet. Thank you for always being by my side both in joys and in sorrows. Since I was studying at Kasetsart University, I found that you are a role model and an impressive senior. Thank you for choosing me and going through several difficult times.

Last but not least, I would like to thank my family for their love and encouragement. They always find a way of making me feel the distance and supporting me with their affection and understanding. Especially to my two younger sisters, Tan and Pare, who always support me mentally and allow me to follow my dreams.

List of Publications

Refereed Journal Paper

- **Robot Behavior Debugger for Non-Expert Users in Convenience Stores Using Behavior Trees**
  Pattaraporn Tulathum, Bunyapon Usawalertkamol, Gustavo Alfonso Garcia Ricardez, Jun Takamatsu, Tsukasa Ogasawara, and Kenichi Matsumoto.
  Advanced Robotics, vol. 36, no. 17-18, 2022, pp. 951-966.
  (Accepted as a journal paper)

Refereed International Conference Proceedings Paper

- **Human-Robot Interaction System for Non-Expert Users in Convenience Stores Using Behavior Trees**
  Pattaraporn Tulathum, Bunyapon Usawalertkamol, Gustavo Alfonso Garcia Ricardez, Jun Takamatsu, Tsukasa Ogasawara, and Kenichi Matsumoto.
  in Proceedings of the 2022 IEEE/SICE International Symposium on System Integration (SII), 2022, pp. 1072-1077.
  (Accepted as a conference paper)

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation and objective

In recent years, many countries such as Japan have been facing labor shortage issues (e.g., decreasing productivity, reducing innovation and development, and difficulty in scaling the business) caused by a declining fertility rate and an aging society [1]. With the advancement of robot technology, robots can be used to facilitate human beings in their daily lives, which is an alternative way to reduce the labor shortage problem [2]. One of the concerns when using robots in a real-world environment is their suitability for the situation and location, as users must consider factors such as human and environmental safety or the accuracy of the work assignments. To address this challenge, robot users who do not have the knowledge and understanding of robots (i.e., non-expert users) should consider and decide on their own how the robot will work according to the situation and location.

The objective of this dissertation is to provide support to non-expert users in configuring the robot behavior by themselves. Specifically, this dissertation proposes a system for non-expert users to create, test, and debug a robot behavior program without the requirement of robot programming experience. The proposed system was designed based on analogies of software development by providing visual information and drag-and-drop composition to assist the users in making a program. The proposed system was evaluated with 24 subjects in total for two experimental purposes. The first experiment was conducted to evaluate to find out whether those subjects are able to create the robot behavior program with the GUI with ten subjects who have no experience with robot programming. The second experiment was evaluated with 14 subjects who could potentially be robot users in the example scenario in terms of effectiveness and usability to create robot behavior programs. The evaluation results show that non-expert users can use the proposed system to create, test, and debug robot behavior programs.

1.1.1 Human-Robot Interaction (HRI)

As labor shortage issues can have a negative impact on business and the economy, adopting a robot is one of the alternatives that can reduce the labor shortage problems and facilitate human beings in their daily lives as a robot can autonomously interact, communicate, and deliver service to customers [3]. HRI becomes essential as the robot should respond to human needs and cooperate to reduce human workloads. Service robots are one example of HRI being used to serve customers in different locations and purposes as they can interact with humans and the environment. Figure 1.1 shows that the service robots perform tasks in various environments and situations such as bookstores, restaurants, hospitals, and convenience stores. Each environment has different location settings, target users, and purposes. Therefore, robot behaviors have to be designed differently based on different environments and situations. For example, a serving robot in a restaurant requires to work interactively while a cleaning robot in a hospital requires to work silently.

Figure 1.1. Examples of HRI applications in various environments and situations such as bookstores, restaurants, hospitals, and convenience stores.

1.1.2 Operating environment

Convenience stores are one of those service locations where store employees and robots can work together in the same environment [4]. In such an environment, it is challenging for robots as they must respond automatically to human behavior and be ready to cope with a constantly changing environment [5]. For example, robots can make in-store management quick and easy, performing a variety of tasks such as stocking items on shelves and cleaning the environment. Robots can also interact with and serve customers while shopping in the store. Moreover, robots can help improve store operations as they can work continuously in 24-hour stores. Therefore, the robot behavior has to be appropriately programmed to handle the dynamic environment. As in the case of humans, robots should be able to observe customer actions and decide on appropriate interactions. Fig. 1.2 shows the example of a situation in a convenience store, where a customer is looking for something.

Figure 1.2. An example of a situation in a convenience store where a robot interacts with a customer who is shopping in the store.

When the robot notices that the customer is “looking,” it responds to the behavior by saying “Are you looking for something?” However, the robot reaction can be anything decided by users to communicate with the customer such as speech (e.g., “Welcome,” “Thank you,” and “Please wait a moment”) and movement (e.g., waving the robot arm, stop or continue working, and move to a different place).

1.1.3 Robot behavior

Robot behavior refers to the actions and movements of a robot in responding to its conditions from surrounding environments such as humans, other robots, animals, or the layout of the working places. Using sensors, the robot can perceive information and make sense of its surroundings and situations. After that, it will obtain that data to process and decide to perform behaviors. Choosing a suitable robot behavior is challenging due to several factors. For example, a robot often operates in complex environments that can change quickly and unpredictably. Moreover, the robot may need to communicate with humans, surrounding environments, or other

robots. Therefore, it is difficult to design proper robot behaviors. Another factor is the target user of the robot.

In this research, two types of robot users were considered based on their expertise in robotics [6, 7]. The first one is expert users, such as robotics engineers with knowledge and experience in robot programming. The second one is non-expert users, such as shop assistants with no robotic knowledge or experience. Expert users can customize the robot behavior, but they may not be familiar with the actual in-store environment and require requirements from non-expert users. Non-expert users, on the other hand, are more familiar with the store environment and know what behavior the robot is supposed to behave in the store. However, they need to rely on expert users to edit or reprogram the robot behavior because they do not have the knowledge and experience to do it themselves.

1.2 Contributions

The contributions of this dissertation are five-fold. First, a concept of a robot behavior program for non-expert users. Second, a robot behavior creator with the simulated convenience store environment. Third, a robot behavior debugger for identifying the issues in the robot behavior program. Forth, a subjective evaluation with real non-expert users. Fifth, a discussion of the evaluation results with a list of comments from non-expert users for future researchers. The following is a brief description of each contribution:

1. A concept of a robot behavior program for non-expert users based on the human decision-making process. The robot behavior program describes the interaction of a robot and surrounding environments by a series of if-then rules, such as a pair of human actions and its robot reaction. This concept is represented by using Behavior Trees. (Chapter 3: Section 3.2)

2. A robot behavior creator that allows non-expert users to create and test the robot behavior program. This includes a graphical user interface and a visualization of the simulated convenience store environment. Non-expert users can create the program and test it in the provided simulator, which can be

used to observe how the robot interacts with the human avatar. (Chapter 3: Section 3.3)

3. A robot behavior debugger that allows non-expert users to identify issues in the robot behavior program. There are four available debugging features including (i) breakpoint, (ii) node execution monitoring, (iii) execution log, and (iv) robot variable monitoring. (Chapter 3: Section 3.4)

4. A subjective evaluation with non-expert users who do not have any experience in robot programming. This evaluation confirms the effectiveness of the proposed system for creating and fixing the robot behavior program in terms of (i) the number of non-expert users who can finish tasks, (ii) the spent time, and (iii) the system usability (SUS score). (Chapters 4 and 5)

5. A discussion and lessons learned from the experimental results. This shows how non-expert users perceive the proposed system based on their comments. The discussion provides insight into the potential factors that may affect the effectiveness of the proposed system. It also opens the possibility to improve the proposed system for future researchers. (Chapter 6)

1.3 Dissertation layout

The rest of this dissertation is organized as follows:

Chapter 2 This chapter introduces the ideas of human-robot interaction (HRI), end-user robot programming, and debugging system design. It also provides the related works for each idea.

Chapter 3 This chapter introduces the proposed solution to create, test, and debug robot behavior programs for non-expert users. First, the concept of the robot behavior program and behavior trees are detailed. After that, the details of a robot behavior creator and a robot behavior debugger are presented.

Chapter 4 This chapter explains the experiments for evaluating the proposed system. First, the details of the environment for the proposed system are explained. Then, the task descriptions with the participant information are presented. Finally, the tasks for evaluating the proposed system are discussed.

Chapter 5 This chapter presents the results of the experiments. It includes the performance of participants to solve given tasks, system usability test, and discussion.

Chapter 6 This chapter addresses the lessons learned from the results of subjective evaluations including feedback from the subjects.

Chapter 7 This final chapter concludes the dissertation and highlights the features of the proposed solutions.

Summary

- The motivation of this dissertation is the necessity of using robots to solve labor shortage issues.
- My objective is to develop a system for non-expert users to create, test, and debug a robot behavior program without the requirement of robot programming experience.
- The contributions of this dissertation are five-fold: First, a concept of a robot behavior program. Second, a robot behavior creator with the simulator. Third, four robot behavior debugging features. Forth, a subjective evaluation with real non-expert users. Fifth, a discussion of the evaluation results.

CHAPTER 2

Related works

2.1 Overview

Allowing non-expert users to program a robot makes the technology more accessible and reduces issues such as labor shortages. This chapter introduces the related works of this dissertation including (i) Human-Robot Interaction (HRI), (ii) End-user Robot Programming, and (iii) the design of debugging system.

2.2 Human-Robot Interaction

Human-Robot Interaction is a rapidly growing field of study that involves understanding how humans interact with robots and developing effective, efficient, and user-friendly systems [8]. HRI plays an important role in service robots for studying robot behaviors to interact and assist humans in daily lives [3, 9, 10]. Using robots to

serve human is one possible way to deal with the aging problem and labor shortage problem [11] as a variety of environmental factors that influence the spread of technologies (e.g., labor costs, power dynamics within organizations and the character of particular job responsibilities [12]). A convenience store is one of the challenging places where a service robot can cooperate and coexist with humans [13]. In this situation, the service robot has to make responses based on the behavior [4, 5]. Shi et al. [14] reported that some shop owners agreed that they would rather operate robots than hire people because the robot cost is cheaper and more likely to attract customers. Therefore, the robot behaviors need to be explicitly configured. To make service robots work in different environments, the robot behaviors need to be configured specifically. Oishi et al. [15] created a prototype tool for non-expert users to configure pre-programmed robot behaviors, but did not allows them to custom the complex behaviors. However, expert users do not know the appropriate robot behaviors for the actual environment. In contrast, non-expert users know the needed behaviors but do not have the knowledge and experience to make them [6]. Overcoming that limitation is quite a challenge as it is needed to find a convenient approach to empower non-expert users to configure robot behavior by themselves [16]. In this dissertation, I use Behavior Trees to represent a robot behavior program. This allows non-expert users to use drag-and-drop composition to create a program instead of writing the actual source code.

2.3 End-User robot programming

End-user Robot Programming enables non-experts to program robots without extensive prior knowledge or experience, leading to greater accessibility and wider use of robots in everyday life [17, 18, 19]. Creating robot programs by end-users, especially non-expert users, is interesting as they may have different perspectives from the expert users (e.g., robot engineers). However, empowering non-expert users to create robot programs for end users can be challenging as they have neither knowledge nor experience [20, 21]. Weintrop [22] reported that I can encourage non-expert users to make a program via more accessible methods such as drag-and-drop programming instead of the traditional typing method. There are several studies that have

proposed visual programming tools to create a robot program. For example, Akiki et al. [23] proposed block-based programming that helps end-users including non-expert users to make robot programs by preparing interfaces and functionality from expert users. Mayr-Dorn et al. [24] assessed end-user programming of robotics in industrial production cells. They highlighted that non-programmers would need additional assistance for robot programming. Savidis [25] proposed the programming experience requirements for visual programming based on standard programming environments. Coronado et al. [26] showed a comparison between different GUI frameworks for non-expert user programming, Behavior Tree and Block-based tools, based on their functionality but did not evaluate those tools in the real situation with humans. Balakirsky et al. [27] proposed an ontology approach to perform a simple kitting task. Iovino et al. [28], Nicolau et al. [29], Marcotte and Hamilton [30] showed that the Behavior Tree is one of the popular frameworks for describing the behavior of artificial intelligence or robots. Tulathum et al. [6], Rovida et al. [31], Marzinotto et al. [32] used the extended Behavior Tree to describe the robot behavior program. Other researchers attempted to generate the Behavior Tree from human demonstration [33]. Unlike related works, I not only focus on providing a tool for non-expert users to create the robot behavior program but also focus on the testing and debugging process as well.

2.4 Design of debugging system

The design of a debugging system is a critical aspect of end-user robot programming, as it helps to identify and resolve errors in robot behavior, resulting in more reliable and accurate performance. Debugging is a basic term in software development and it is defined as "the attempt to pinpoint and fix the source of an error" [34]. In practice, debugging is a complex process influenced by users' experience levels and the tools they have at their disposal. Lawrance et al. [35], Bednarik [36] found that software developers usually find a clue in the program first for debugging tasks. Ikeda and Szafir [37, 38] reported that the provided visualization tool via 2D and 3D displays could improve the expert users' debugging capabilities. Eclipse IDE, a well-known tool for the Java language, provides features for debugging software

such as showing logs, variables, and breakpoints [39]. Murphy et al. [40], Beller et al. [41], and Perscheid et al. [42] reported that the breakpoint feature is one of the most frequently used features in software development practice. Bednarik [36], Grigoreanu et al. [43], Manfredi et al. [44], Cao et al. [45], on the other hand, investigated the strategy to debug a program by end-user. They found the end-users use a sensemaking model in the debugging process and usually rely on both code and graphical representation of the code or data. For debugging the program with interactive data, Hoffswell et al. [46], Tolksdorf et al. [47] demonstrated a tool that allows developers to replay and inspect information of data logs over time step-by-step. In terms of robotics debugging research, Campusano and Bergel [48] extracted the log traces and visualized the state of the robot using a state machine model. In this dissertation, I provide four debugging features including (i) breakpoint, (ii) node execution monitoring, (iii) execution log, and (iv) robot variables to allow non-expert users to figure out the problems.

Summary

- Related works on HRI and end-user programming focus on the graphical interface for creating a robot program.
- This dissertation not only focuses on the graphical interface for creating the robot behavior program but also focuses on debugging and testing with visualization and simulation.
- This dissertation also provides approaches to evaluate the effectiveness of the robot behavior creator and debugger.

CHAPTER 3

Proposed system

3.1 Overview

This chapter describes a visual programming approach to support non-expert users in creating a robot program without knowledge and experience. The key concept of the proposed system is to allow non-expert users in making pairs of human actions and robot reactions to create a program using drag-and-drop composition through GUI without coding. After creating a program, non-expert users can confirm the result via the simulated convenience store environment with the actions of the human avatars. Moreover, non-expert users can fix issues in the robot behavior program using provided four debugging features in order to speed up debugging process.

This chapter is organized as follows. Section 3.1 shows the overview of the proposed system, challenges, and examples of the system usage. Section 3.2 presents the details of the robot behavior creator with the idea and concept of the robot behavior

program. Finally, Section 3.3 presents the robot behavior debugger with its idea and concept of each feature.

3.2 Robot behavior program

This section introduces the overview of the proposed system. Figure 3.1 shows the working process and users' roles. To use the robot behavior creator, expert users need to prepare sets of human actions and robot reactions as a preliminary process. Then, non-expert users can use those functionalities to create a robot behavior program by making pairs of human actions and robot reactions from the prepared sets. After that, non-expert users can test their created programs via the robot simulator to see how a robot responds to a customer in a simulated convenience store environment. When the non-expert users observe that there is an unexpected result, they can use the proposed debugger to debug the program via four components including breakpoints feature, node execution monitoring feature, node status logging feature, and robot status feature. Examples of my system usages are shown in the following link: https://youtube.com/playlist?list=PLCMeFFUGH8tqqzykNbLwf6uudgF1yLLsN

3.2.1 Concept of a robot behavior program

The concept of a robot behavior program is a mimic of the human decision-making process [49]. The decision-making process starts with humans collecting a piece of information from the past and the present. Then, they use this information to reason and decide what to do in the next step. During the reasoning process, humans have to set some criteria in order to make a decision based on the collected information. By looking at the robot perspective, human action can be used as a condition for deciding which response should be made. Figure 3.2 shows an example of how a human makes a decision in the robot perspective. Human action is considered as the condition while robot reaction is considered as the action of the robot. From this example, if human action is waving, robot should react by saying "Welcome!".

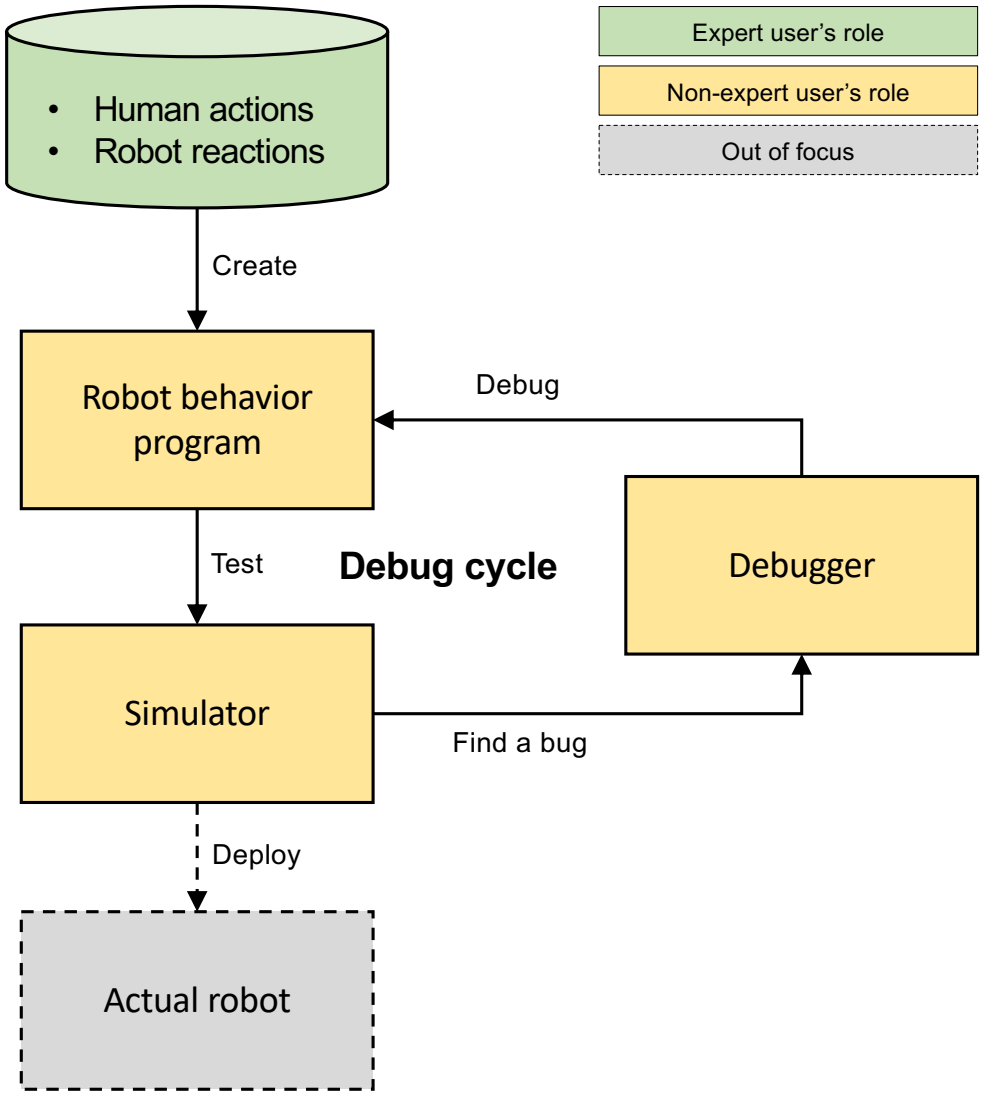


Figure 3.1. Overview of the proposed system shows the working process and users' roles.

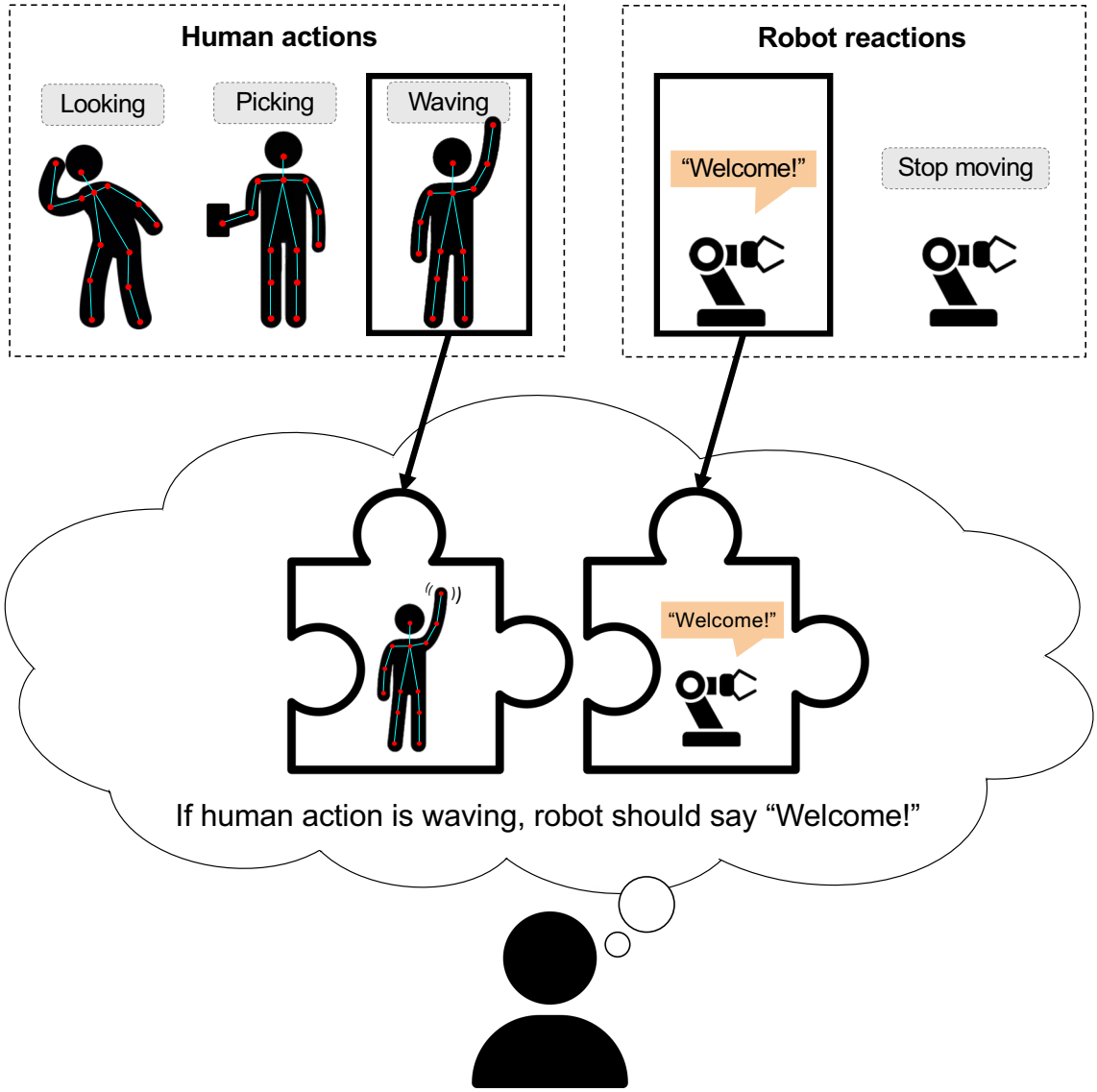


Figure 3.2. Concept of human making decision by paring a human action as condition and a robot reaction as action from the perspective of the robot

3.2.2 Behavior Tree

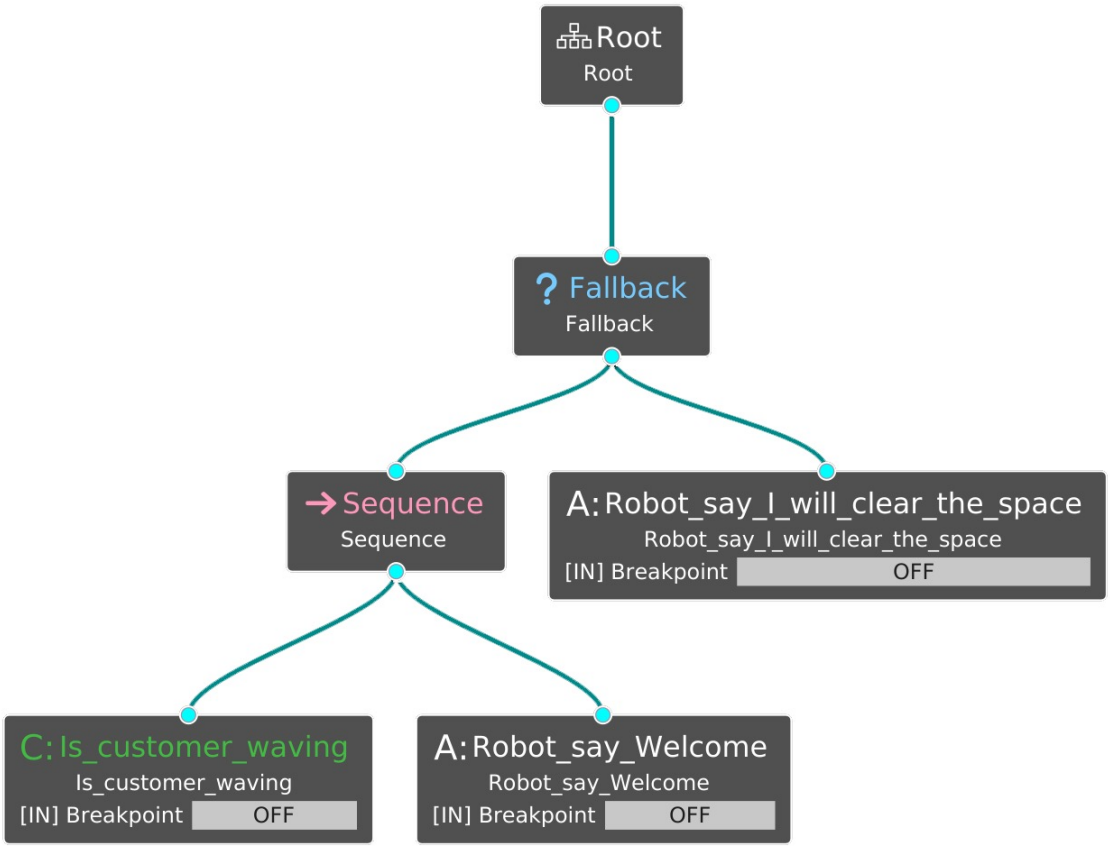


Figure 3.3. Example of a Behavior Tree that represents the robot behavior to check whether or not a human is waving her hand

The Behavior Tree is a tree-like structure representing task switching in the robot under various conditions to construct a robot behavior program [32, 50]. Figure 3.3 shows an example of Behavior Tree which can be interpreted as: (i) the robot will say "Welcome" when a human waves hands, otherwise (ii) the robot will say "I will clear the space." The concept of the Behavior Trees is that the program must be executed step by step from top-to-bottom and left-to-right. Moreover, the Behavior Trees provide human readability, code reusability, and modularity. Therefore, non-expert users can use it to create robot programs by mimicking human thinking processes for

deciding whether to do or skip a task. Note that this dissertation uses the simplified version of the Behavior Tree, which the differences are discussed in Chapter 6.

The proposed system uses four Behavior Tree nodes to represent the robot behavior program, namely, (i) condition node, (ii) action node, (iii) sequence node, and (iv) fallback node. Each node has its own status (i.e., success, running, and failure), which returns after executing that node and may affect the other nodes. The essential nodes for creating a program are the *condition node* and the *action node*, which the robot uses to make a decision and respond to a human. The condition node represents the human actions that the robot uses to make decisions before proceeding to the next node. The action node represents how the robot responds to human actions (e.g., speech, and motion). To make a robot performs complex actions, additional nodes are needed for considering whether to continue or skip a task. Therefore, this dissertation uses the *sequence node* and the *fallback node* because the concepts of these nodes are straightforward to decide whether to continue the task as long as the node status is success, or to try executing the node until its status is success, respectively. The sequence node is a control node that is used to decide how to continue doing the tasks. This node will visit each child node from the leftmost child to the right sequentially until successfully executing every node. If the status of all nodes under the sequence node are success, the status of the sequence node will also be success. Otherwise, the status of the sequence node will be failure. The fallback node is another control node that is used to decide what to do or if a task should be skipped. This node will visit each child node from the leftmost child to the right sequentially until finding the first child node that is successfully executed. If the status of all nodes under the fallback node are failure, the status of the fallback node will be failure too. If at least one of the child nodes succeeds, the status of the fallback node will be success.

3.3 Robot Behavior Creator

The system consists of a Behavior Tree GUI editor and a robot simulator. Figure 3.4 shows the robot behavior creator. This system allows non-expert users to create a robot behavior program via GUI and test it with the robot simulator.

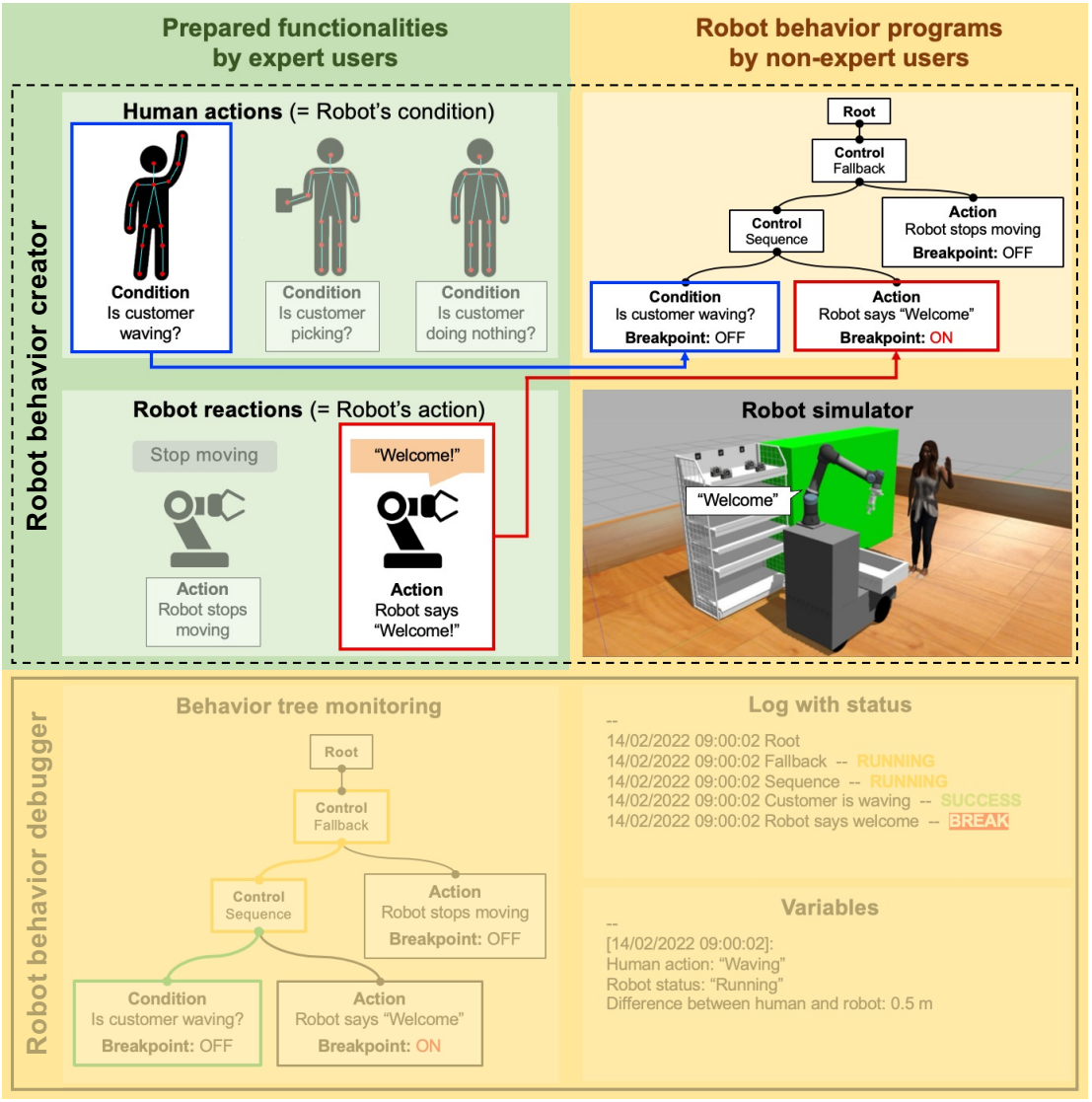


Figure 3.4. Proposed robot behavior creator to assist non-expert users in creating a robot behavior program.

3.3.1 Behavior Tree GUI editor

I provide a Behavior Tree GUI to allow non-expert users to create robot behavior programs by constructing nodes and edges visually. First, expert users need to

prepare a set of human actions and robot reactions as Behavior Tree condition nodes and action nodes, respectively. Then, non-expert users can choose desired Behavior Tree nodes in the panel using drag-and-drop into the creative space to create the program. This system uses Groot[1] as the Behavior Tree GUI.

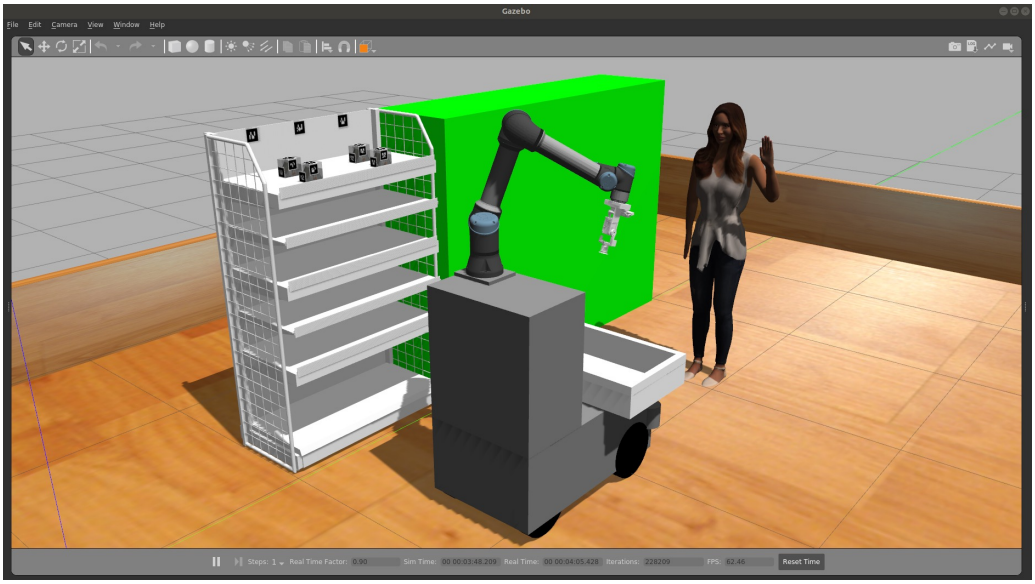
3.3.2 Robot simulator

A robot simulator that allows non-expert users to test their created robot behavior programs. The simulator is a supporting tool for non-expert users that provides a graphical simulated robot working in a convenience store environment. After creating a robot behavior program, non-expert users can test their program via the provided robot simulator to see how the virtual robot interacts with the actions of human avatar. Figure 3.5 shows examples of human actions in simulator which are waving and picking. Therefore, non-expert users can visualize how the robot responds to human actions. The simulator is not only for testing the program but can support users to find bugs in the program. When non-expert users observe any issues in the simulation (e.g., a robot responds to the human with a wrong reaction), they can use that information to fix the bugs via the Behavior Tree GUI editor. This system uses Gazebo[2] as the robot simulator.

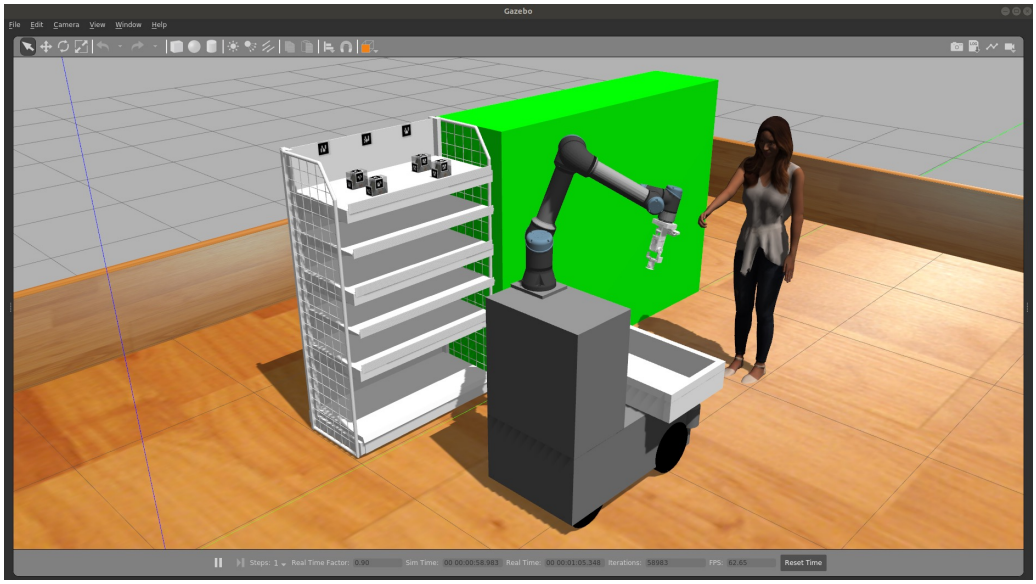
3.4 Robot Behavior Debugger

In addition to the robot behavior creator, I propose a robot behavior debugger to assist non-expert users to fix issues in the robot behavior program. Figure 3.6 shows my current proposed system in the solid frame to add debugging abilities. The robot behavior debugger has four features: (i) breakpoint, (ii) node execution monitoring, (iii) execution log, and (iv) robot variables.

[1]Groot, https://github.com/BehaviorTree/Groot
[2]Gazebo, http://gazebosim.org/



(a) Simulator with a human avatar waving.



(b) Simulator with a human avatar picking.

Figure 3.5. A comparison of the simulator with a human avatar between the case where a robot behavior is successfully executed. (a) shows the case that robot interacts with a human avatar waving and (b) shows the case that robot interacts with a human avatar picking

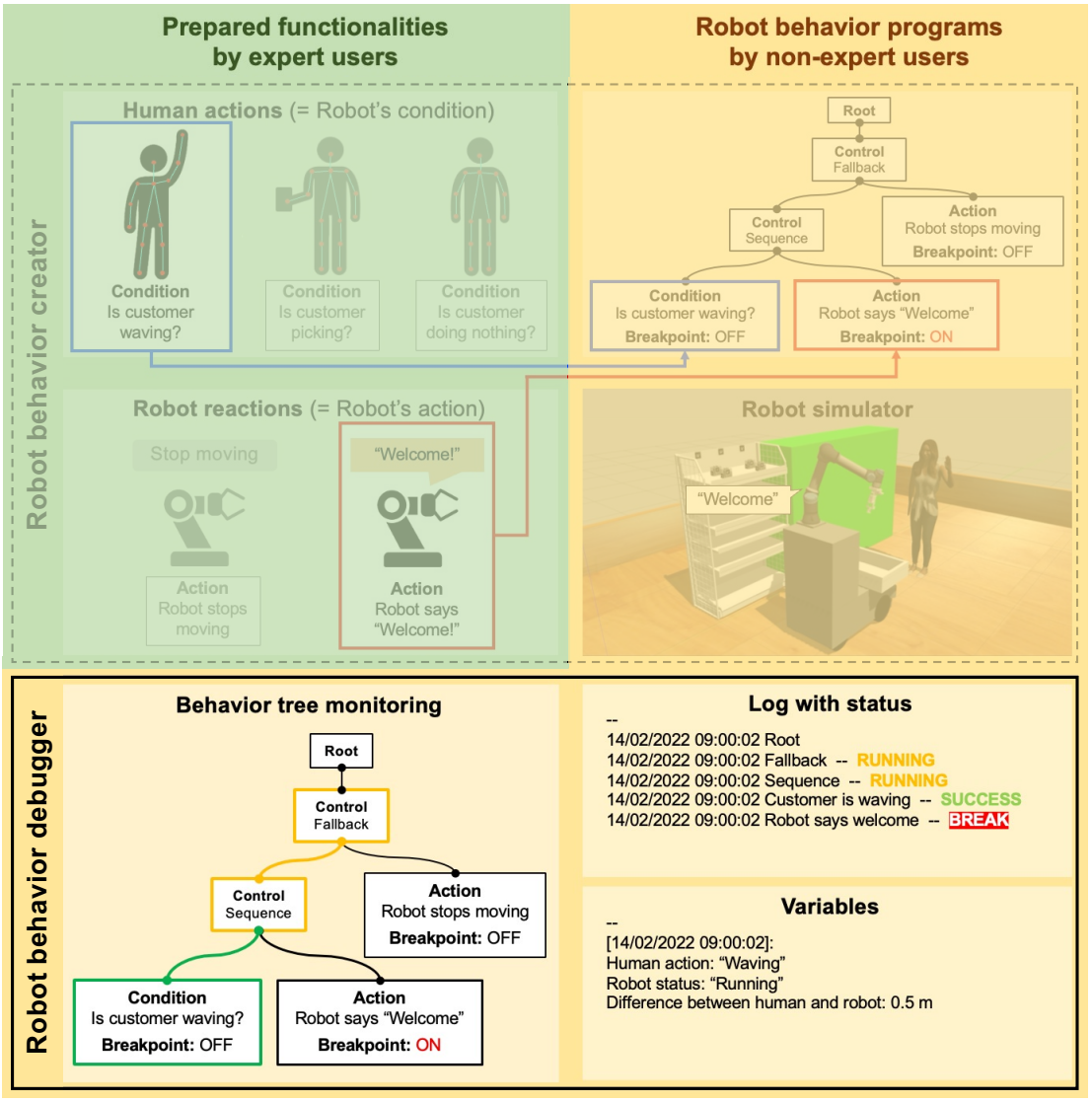


Figure 3.6. Proposed robot behavior debugger to assist non-expert users in finding and fixing issues in the robot behavior program.

3.4.1 Breakpoint

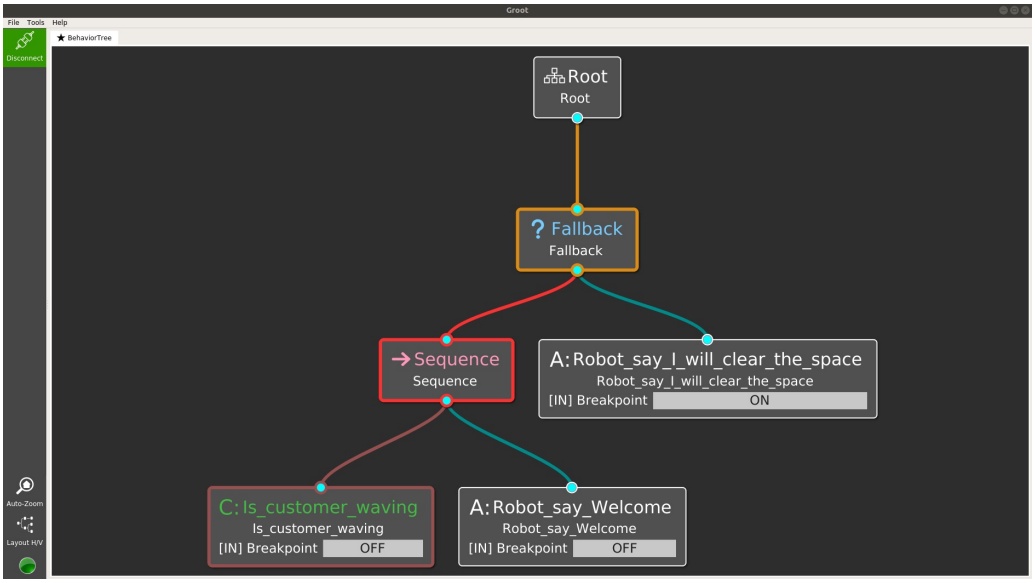
A breakpoint is a feature that can be set to suspend the execution of the running program [51]. Breakpoints help developers to pause the program and check the status of the working environment to find the root cause of issues in their code, as shown in Figure 3.7. With the similar nature of creating the robot behavior program and writing the code, I proposed the breakpoint feature to pause the robot behavior in the program at the specific nodes. Moreover, my system allows users to pause and continue the program with multiple breakpoints. The example of breakpoint usage is shown in Figure 3.7. When the program finds the first breakpoint, it will pause and continue from the specific node until it reaches the next breakpoint. By using breakpoints, non-expert users can observe the robot behavior in the simulated environment through the following features that I propose.

3.4.2 Node execution monitoring

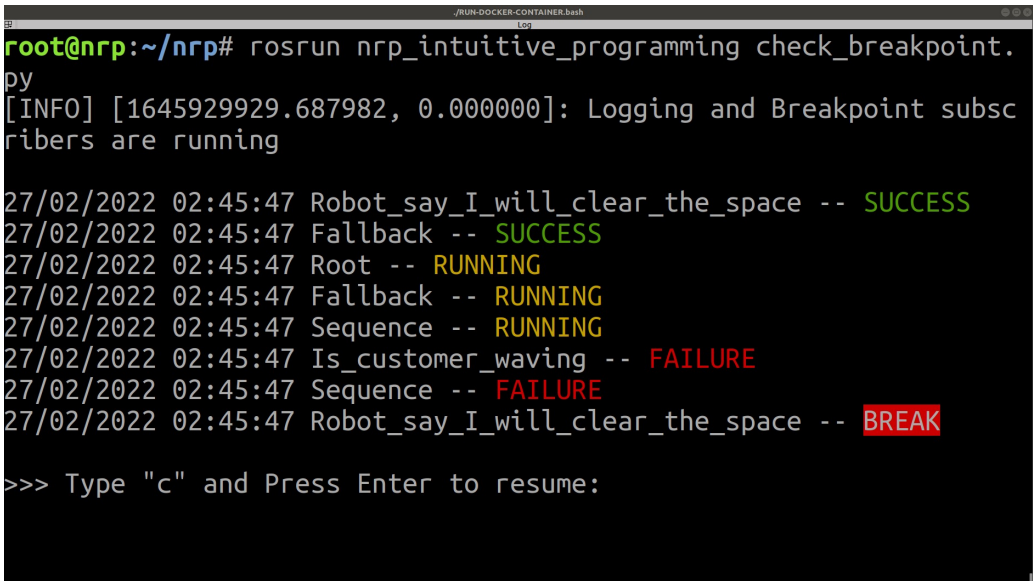
To visually show the progress of robot behavior, I can visually show it on the Behavior Tree directly. Hence, I provide the feature to monitor the status of node execution within the Behavior Tree by highlighting nodes and edges in different colors. The example of the execution monitoring as shown in Figure 3.8. The status color codes are the following: (i) Orange for running, (ii) Green for success, and (iii) Red for failure.

3.4.3 Execution log

Logging information is used to observe how the events occurred or how the program runs. It is important to understand the current status of the robot behavior to address bugs in the robot behavior program. In this work, I provide the information of node status that is being processed in the program with a timestamp. The example of execution log is shown in Figure 3.9. The status that I provide consists of (i) running, (ii) success, (iii) failure, and (iv) break (i.e., when reached a breakpoint).

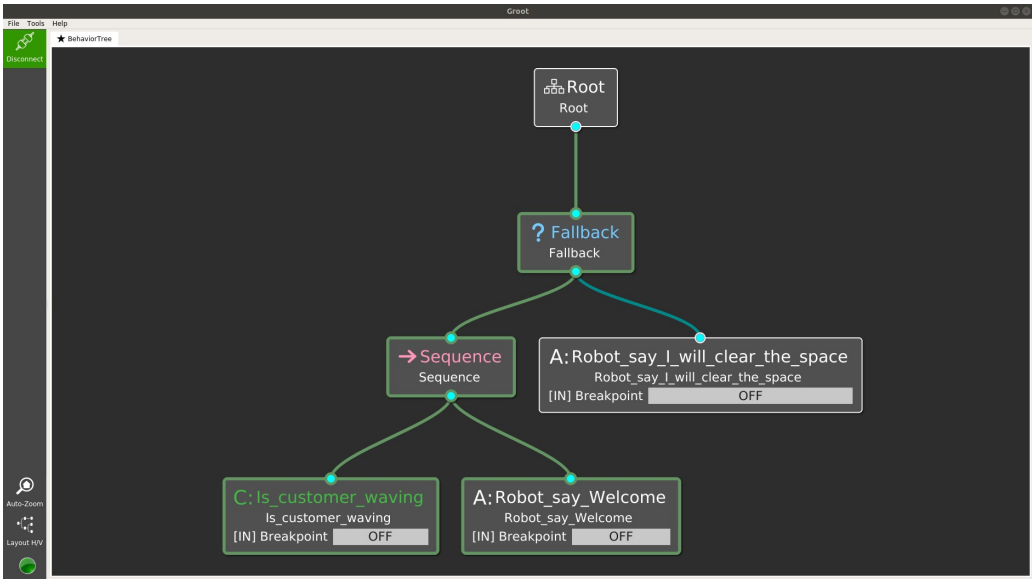


(a) Behavior Tree GUI after turning ON the breakpoint

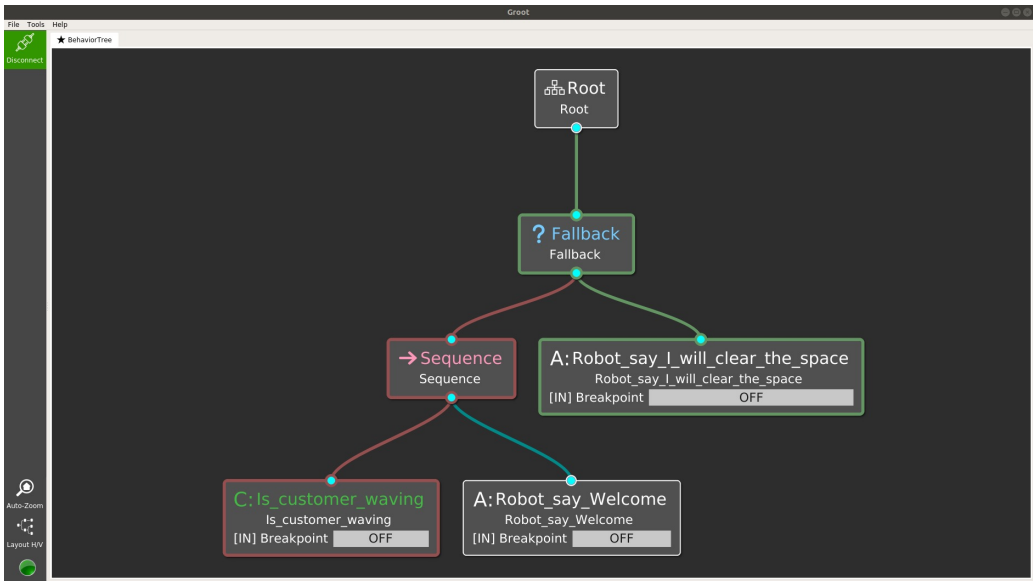


(b) Log after turning ON the breakpoint

Figure 3.7. Example of breakpoint feature usage. Non-expert users can turn ON a breakpoint at any node to pause the executed program intentionally.

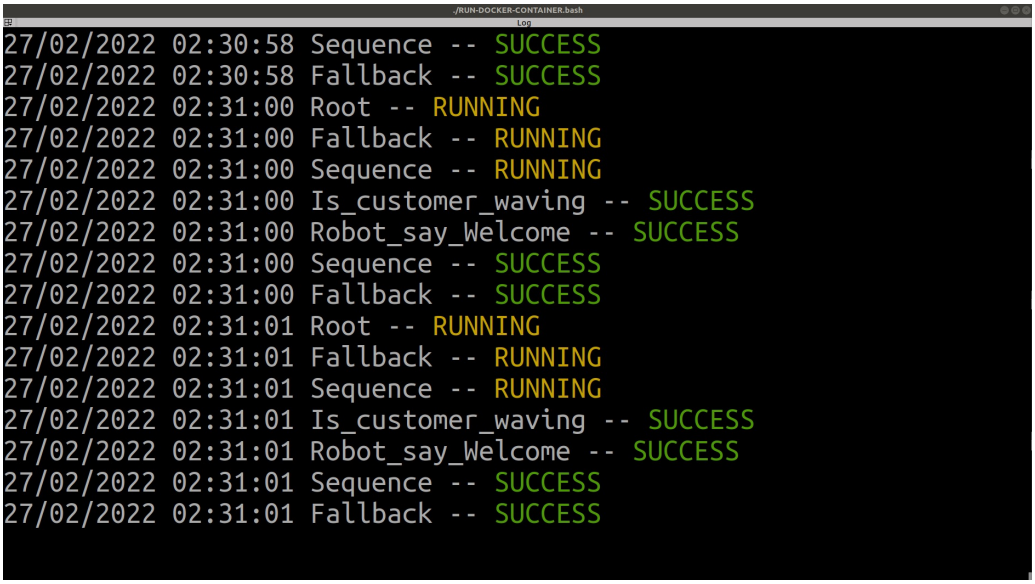


(a) Behavior Tree monitoring with all success status (i.e., all green nodes).

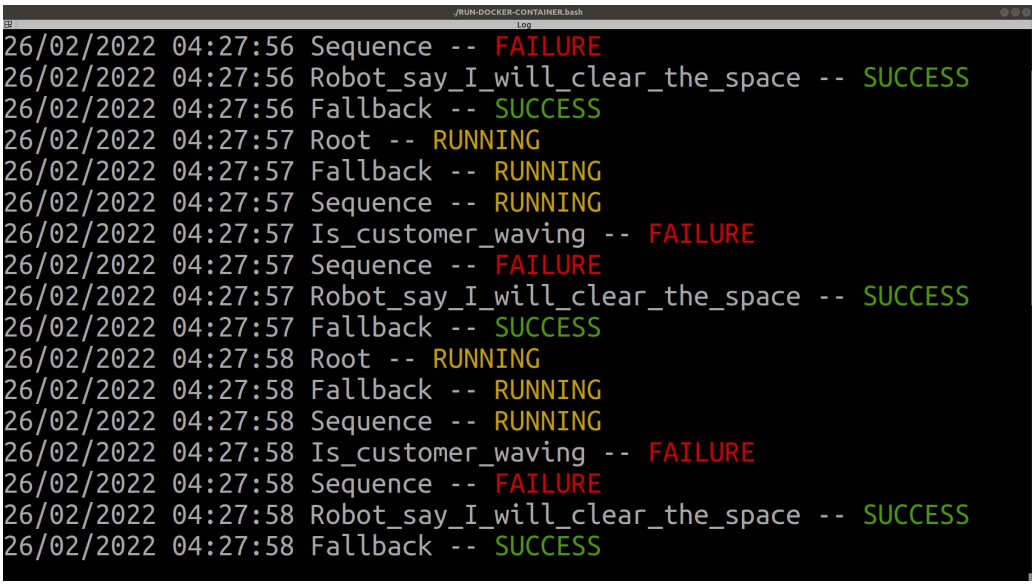


(b) Behavior Tree monitoring with some failure status (i.e., some red nodes).

Figure 3.8. A comparison of the node execution monitoring feature between the case where a robot behavior is successfully executed. (a) shows the case that robot is able to detect a human waving, otherwise, (b) shows issues occur



(a) Log status with success status.



(b) Log status with failure status.

Figure 3.9. A comparison of the node status logging feature between the case where a robot behavior is successfully executed. (a) shows the case that robot is able to detect a human waving, otherwise, (b) shows issues occur

3.4.4 Robot variable

As mentioned by the subjects in my preliminary study, it is hard to understand the robot and human status in the simulator. To fill this gap, I propose a debugging feature to show the robot status, current human action, and distance between them which are gathered from the simulator.

Summary

- The key concept of the proposed system is to allow non-expert users in making pairs of human actions and robot reactions to create a robot behavior program using drag-and-drop composition through GUI.
- The robot behavior program is represented by Behavior Trees.
- The robot behavior creator allows non-expert users to create the robot behavior program and test it in the simulated convenience store environment.
- There are four features for the robot behavior debugger including (i) breakpoint, (ii) node execution monitoring, (iii) execution log, and (iv) robot variables.

CHAPTER 4

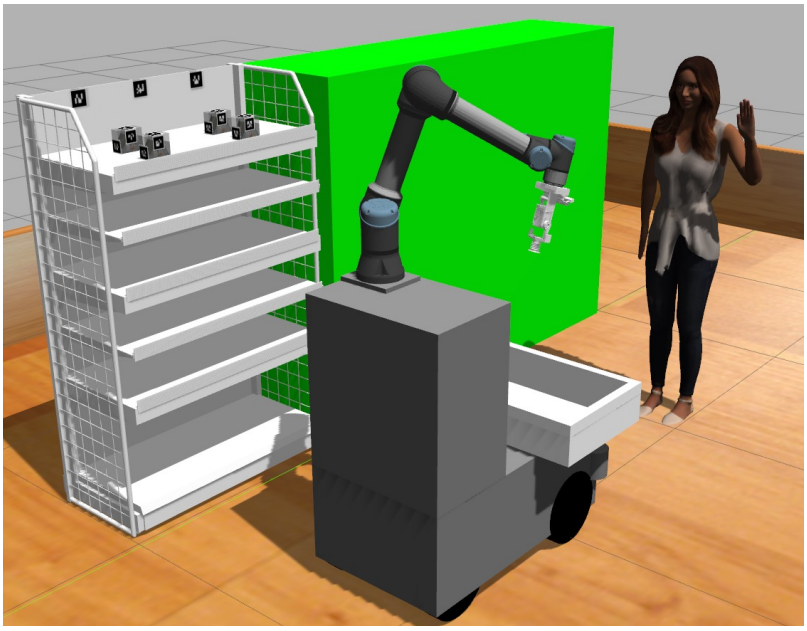
Experiments

4.1 Experimental Setup

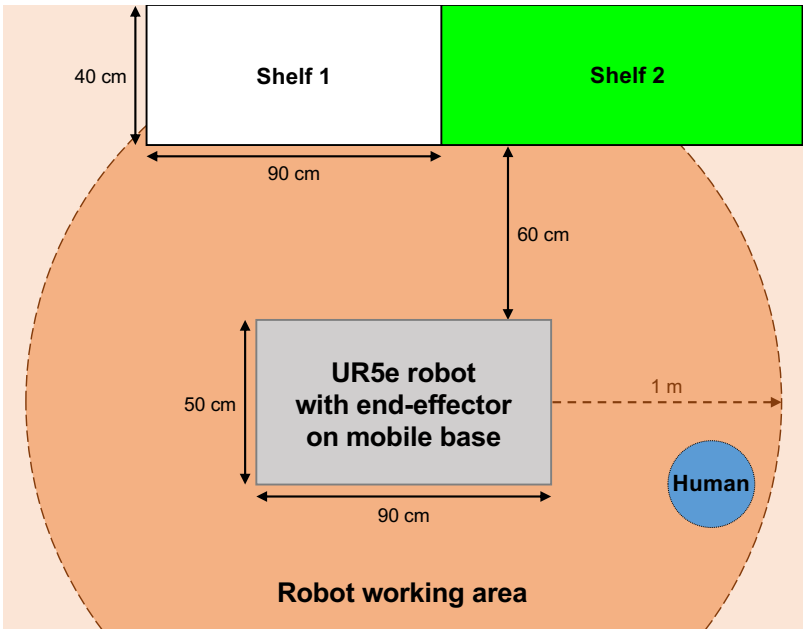
This chapter describes the experimental setup, the details of the subjects, tasks, and subjective evaluation of the robot behavior creator and the robot behavior debugger. Specifically, the details of a convenience store in the simulator and the experimental procedure are explained below.

4.1.1 Convenience store setup in the simulator

As shown in Figure 4.1, the convenience store environment setup in the simulator has one robot with two shelves and one human avatar that is in the robot's working area. The UR5e robot arm which has 7 degrees of freedom mounted on a mobile base in front of shelves (i.e., 60 centimeters) is used for the experiments.



(a) Human, robot, and convenience store environment in Gazebo simulator



(b) Top-view diagram of human, robot, and convenience store.

Figure 4.1. Convenience store environment for the simulation in my experiment.

During the experiment, the human avatar can change position within or outside the robot working area (i.e., 1 meter). This human avatar can perform actions depending on the given tasks. As shown in Table 4.1, there are four human actions and eight robot reactions that can be used within the proposed system.

Table 4.1. The list of human actions and robot reactions available in my experiment.

Human actions	**Robot reactions**	
	Speech	**Motion**
- Waving their hands	- Saying "Please go ahead"	- Halting the current task
- Looking for an item	- Saying "Welcome"	- Continue the current task
- Picking an item	- Saying "Thank you"	
- Approaching a robot within 1 m	- Saying "I will clear the space"	
	- Saying "Please wait a moment"	
	- Saying "I will resume my work"	

4.2 Experiment for the robot behavior creator

The objective of this experiment is to evaluate whether or not non-expert users can create and test robot behavior programs by using the concept of pairing human action and robot reaction. The details of the procedure, participants, and tasks are shown in the following sections.

4.2.1 Procedure of the experiment

The experiment for assessing the effectiveness of the robot behavior creator took around 90 minutes for each subject. The procedure of the experiment is detailed as follows:

1. Each subject was tutored for up to 30 minutes individually. I instructed the subjects as follows: (i) introducing HRI in convenience store, (ii) introducing the robot behavior program, (iii) showing an overview of the robot behavior creator with the simulator. Participants also got five minutes to practice with an example task and ask for clarification during the tutoring process.

2. Each participant spent approximately 50 to 60 minutes completing three given tasks. The details of the tasks are stated in Section 4.2.3.

3. Information from the experiment and feedback from subjects were collected for the evaluation detailed in Section 4.4.

4.2.2 Participants

As this experiment is an exploration to find whether non-expert users understand the concept of the robot behavior program, I recruited graduate students who did not have IT-related backgrounds as the experimental subjects. Table 4.2 shows the demographics of the subjects. My participants consist of four females and six males. The age range of the participants is from 25 to 31 years old (The median of the participants is 26 and the standard deviation is 2.39). There are six participants who are studying in the biological science division and four participants who are studying in the material science division. For the experience in programming, all participants do not have any experience in robot programming. There is one participant who has experience in visual programming (e.g., Scratch: block programming). Five participants have experience in at least one programming language, but mainly from university courses (e.g., C#, Java, and Python).

Table 4.2. Demographic of the experimental subjects for the first experiment (ten graduate students).

Number of subjects	10
Age Range	25 to 31 years old (Median = 26, SD = 2.39)
Gender	Female = 4 Male = 6
Division	Biological Science = 6 Material Science = 4
# participants who have experience in programming	Robot programming = 0 Visual programming = 1 Other programming = 5

4.2.3 Tasks

The first task of this experiment is to fix a provided, buggy robot behavior program to comply with the given situation. This task aims to confirm that subjects understand the basic concept of the robot behavior program. As shown in Figure 4.2, the given situation is detailed as follows:

- If a human is waving at the robot, the robot should say "Waving."
- Otherwise, the robot should say "Please go ahead."

For the initial buggy robot behavior program in Figure 4.2 (left), there are three mistakes (i) wrong control node, (ii) wrong action node, (iii) misplaced condition node. The correct robot behavior program is shown on the right side of the figure.

The second task is to create a simple robot behavior program with only a single condition. Non-expert users have to create the program from scratch (i.e., no initial program). Figure 4.3 is the example solution for the second task. The situation in this task is:

- If the robot detects human waving, then the robot should say the following sentences:
  1. "I will clear the space."
  2. "Please wait a moment."
- Otherwise, the robot should say "Thank you."

The third task is to create a more complex robot behavior program with multiple conditions. Figure 4.4 is the example solution for the third task. The situation in this task is:

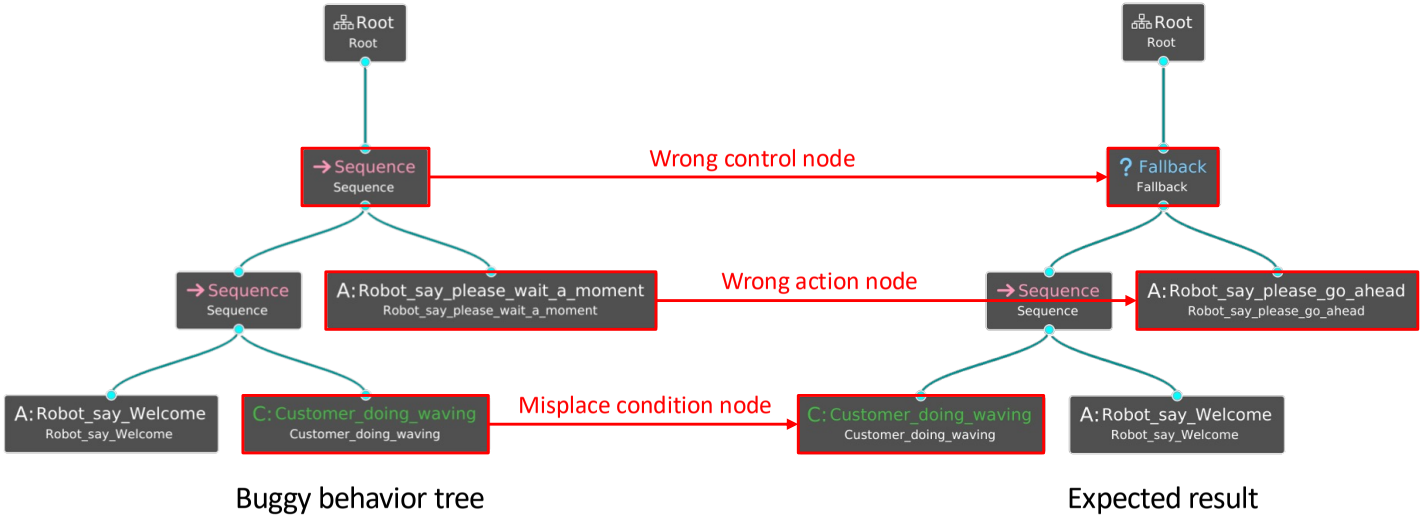


Figure 4.2. The given robot behavior program of the first task. A buggy tree with bugs is highlighted in red. To solve this task, the non-expert users have to understand the conditions and actions that the robot will use to make a decision and the rule of the robot behavior program that always executes a tree from top-to-bottom and left-to-right

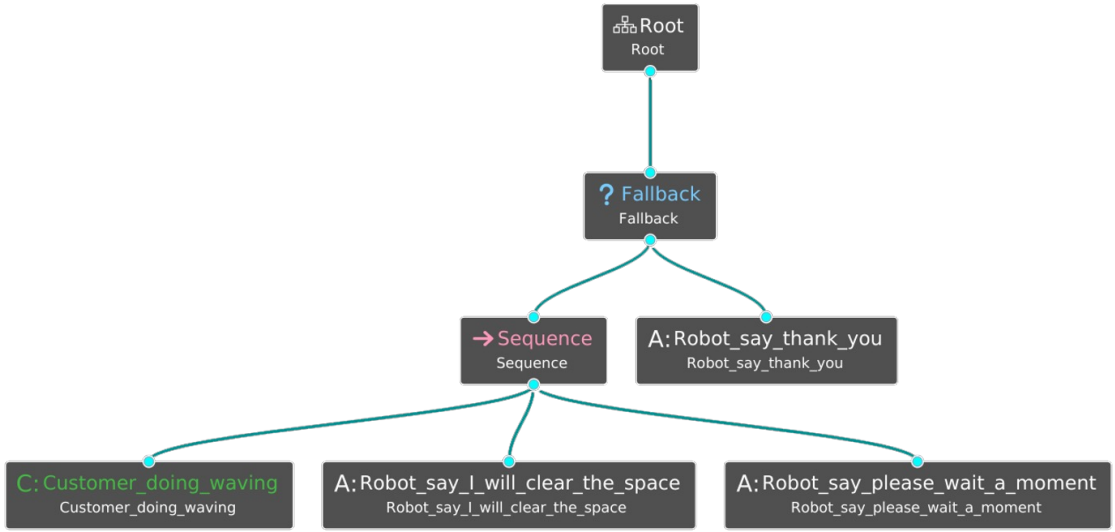


Figure 4.3. Example of solutions for the second task.

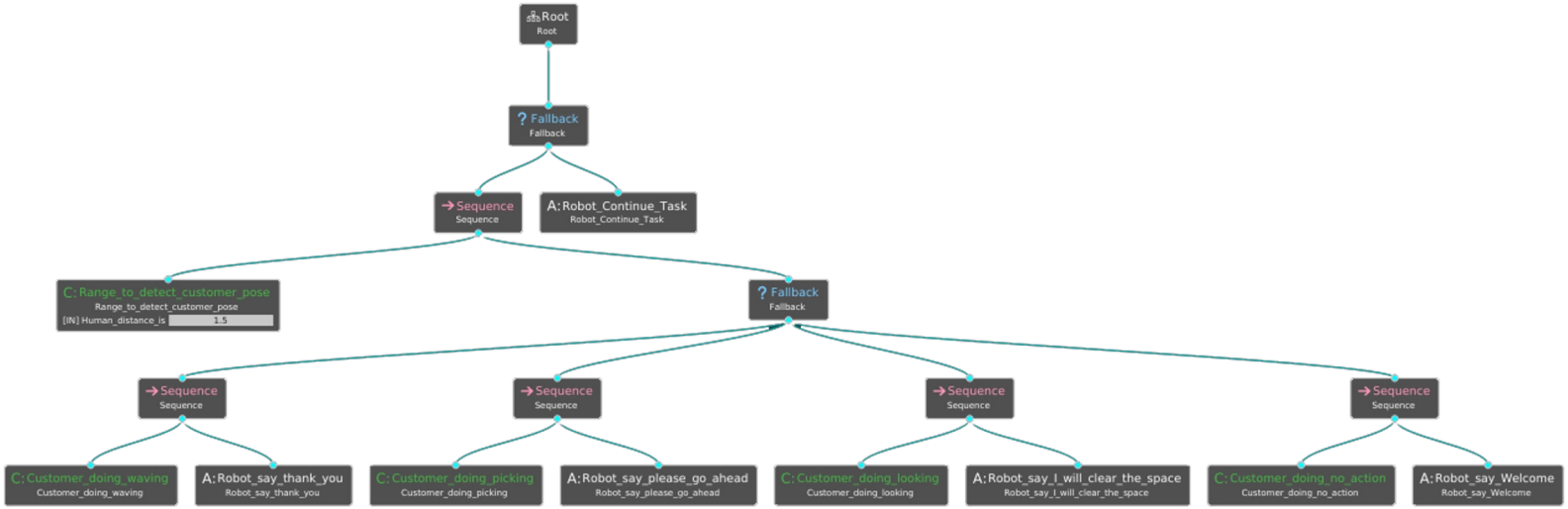


Figure 4.4. Example of solutions for the third task.

- If a human is in the robot working area (i.e., within one meter from the robot), the robot should behave as following conditions:
  - The robot should say "Welcome" if it detects a human standstill (i.e., no action).
  - The robot should say "I will clear the space" if it detects a human looking for items.
  - The robot should say "Please go ahead" if it detects a human picking some items.
  - The robot should say "Thank you" if it detects a human waving their hand.
- Otherwise, the robot should continue the task.

4.3 Experiment for robot behavior debugger

The objective of this experiment is to evaluate whether or not non-expert users can utilize the breakpoint feature to fix issues in robot behavior programs.

4.3.1 Procedure of the experiment

The experiment for evaluating the effectiveness of the robot behavior debugger took approximately 90 minutes to complete per subject. The procedure of the experiment is detailed as follows:

1. Divided 14 subjects into two groups equally. For the first group, the breakpoint feature was introduced to subjects (i.e., with-breakpoint group). On the other hand, the breakpoint feature was not introduced to the second group (i.e., without-breakpoint group). The hypothesis is that the breakpoint feature can decrease the number of attempts and time to finish the task.
2. Each subject was tutored for up to 30 minutes individually. I instructed participants on the following steps: (i) introducing HRI in convenience store, (ii)

showing an overview of the system, (iii) introducing the robot behavior program, and (iv) explaining the debugging features. Participants also got five minutes to practice using the system and ask for any clarification during the tutoring process.

3. Each participant spent approximately 50 to 60 minutes completing three given tasks by using the proposed system with the robot behavior debugger (i.e., with or without breakpoint feature based on their group). The details of the tasks are stated in Section 4.3.3. Note that the subjects from with-breakpoint group were encouraged, but not forced, to use the breakpoint feature.

4. Information from the experiment and feedback from subjects were collected for the evaluation detailed in Section 4.4.

4.3.2 Participants

As this experiment is an extension of the first experiment, I recruited real non-expert users who are not graduate students and are more likely to satisfy the requirement of a convenience store staff job. Table 4.3 shows the demographic details of the subjects in this experiment. I recruited 14 subjects with ages ranging from 22 to 61 years old (the median of the subjects' age is 43, and the standard deviation is 13.4). I believe that the subjects that I recruited to do experiments have a similar background to the actual convenience store staff in Japan and possess the requirements of the convenience store staff such as performing basic computer operations (no programming background is needed). The occupations of the test subjects in this study were diverse but unrelated to technology, such as non-IT-related college students, secretaries, plumbers, kindergarten teachers, and housewives.

4.3.3 Tasks

The first task of my experiment is to fix a provided, buggy robot behavior program to comply with the given situation. This task aims to confirm that subjects understand the condition and action nodes of robot behavior programs. As shown in Figure 4.5, the given situation is detailed as follows:

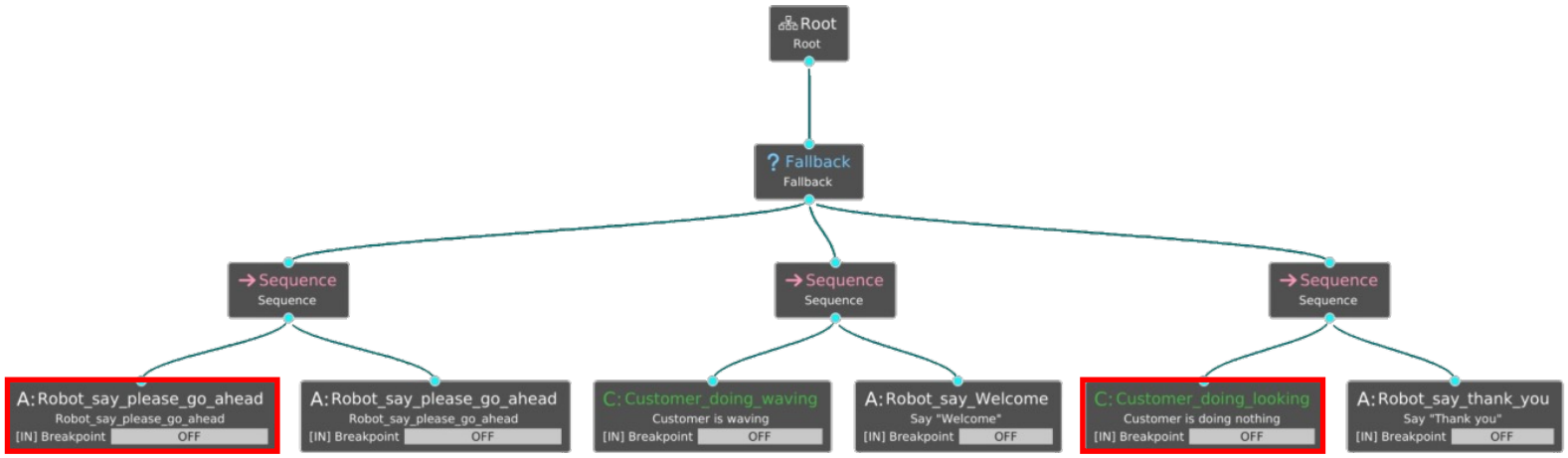
Table 4.3. Demographic of the subjects in the second experiments. (u-student = undergraduate student).

Number of subjects	14
Age range	22 - 61 years old (Medium = 43, SD = 13.4)
Gender	Female = 12 Male = 2
Programming experience	Robot programming = 0 Visual programming = 0 Other = 0
Field of study (occupation)	Economy (u-student) = 1 Psychology (u-student) = 1 Food management (u-student) = 1 Language education (u-student) = 1 Education (housewife) = 1 Sediment disaster (plumber) = 1 English education (office worker) = 1 English literature (office worker) = 7

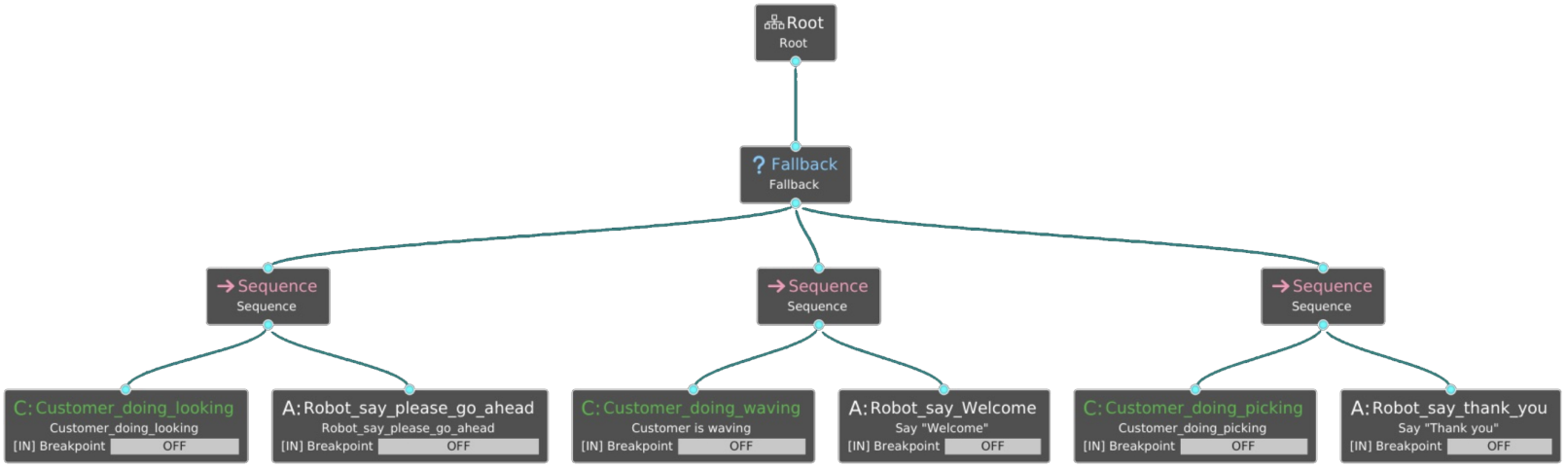
- If a human is looking at the robot, the robot should say "Please go ahead."
- If a human is waving at the robot, the robot should say "Welcome."
- If a human is picking an item, the robot should say "Thank you."

For the initial buggy robot behavior program in Figure 4.5(a), there are two wrong condition nodes that subjects have to replace. The example solution for this task is shown in Figure 4.5(b).

The second task is also to fix a robot behavior program, but with more complicated bugs than the first one. This task aims to confirm that subjects understand the control nodes (i.e., sequence and fallback nodes). As shown in Figure 4.6, the given situation is detailed as follows:

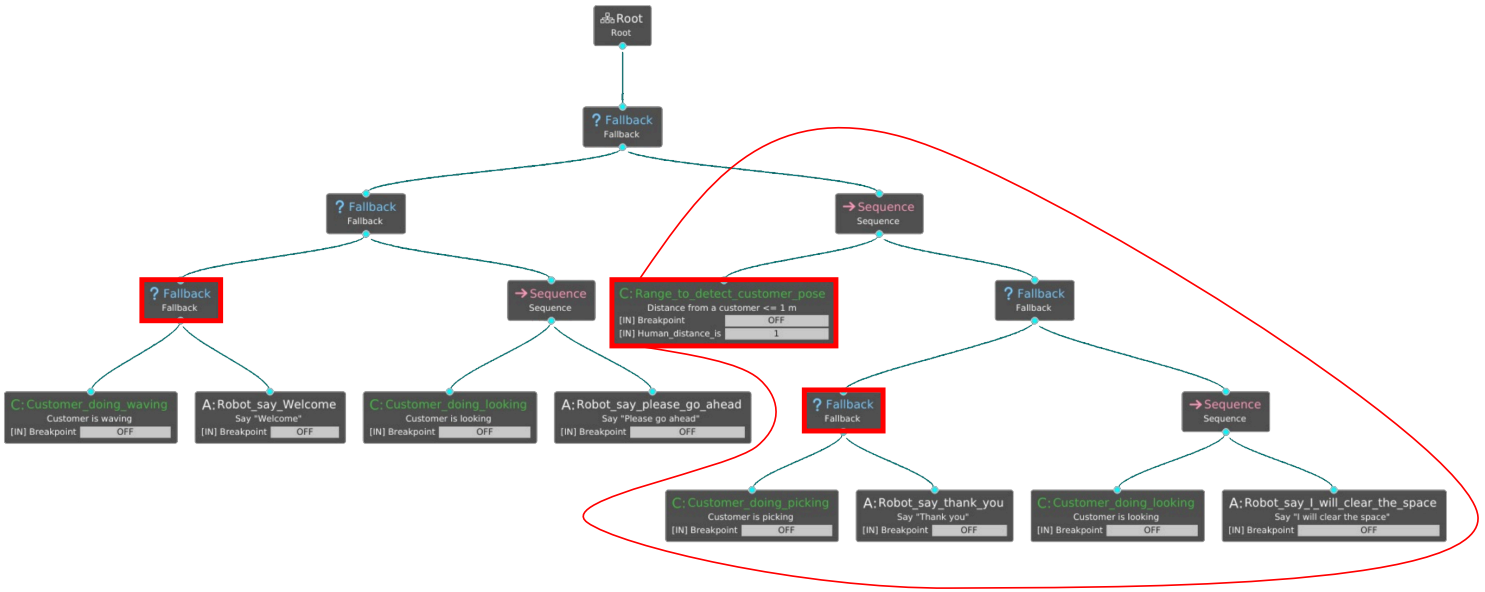


(a) A buggy tree with the bugs highlighted in red. To solve this task, the non-expert users have to understand the conditions and actions that robot will use to make a decision and the rule of the robot behavior programs that always execute a tree from top-to-bottom and left-to-right

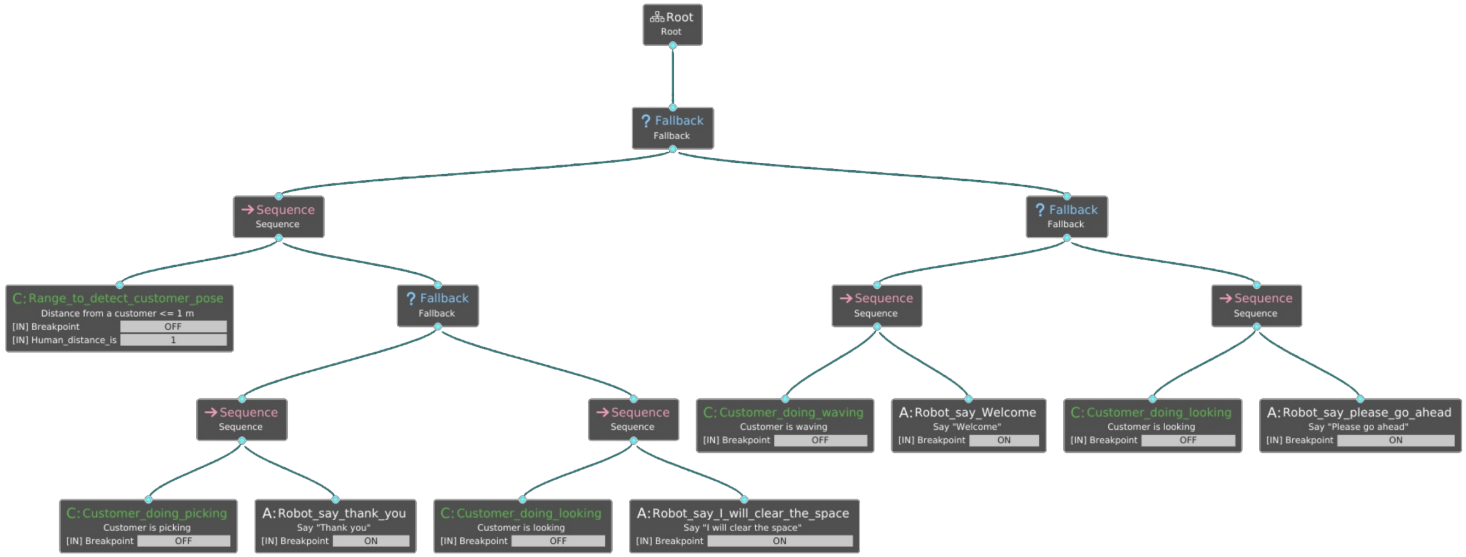


(b) An example solution.

Figure 4.5. Robot behavior program for the first task.



(a) A buggy tree with the wrong nodes and wrong order highlighted in red. To solve this task, the non-expert users have to know the concept of the control nodes (i.e., sequence and fallback node) and the rule of robot behavior programs that always execute a tree from top-to-bottom and left-to-right.



(b) An example solution.

Figure 4.6. Robot behavior program for the second task.

- If a human is in the robot range (i.e., one meter) and:
  - If a human is picking an item, the robot should say "Thank you."
  - If a human is looking at the robot, the robot should say "I will clear the space."
- Otherwise:
  - If a human is waving at the robot, the robot should say "Welcome."
  - If a human is looking at the robot, the robot should say "Please go ahead."

For the initial buggy robot behavior program in Figure 4.6(a), there are two wrong control nodes to be replaced and one wrong order of nodes. The example solution for this task is to move the right subtree to the left side and replace two fallback nodes with sequence nodes as shown in Figure 4.6(b).

The third task is to create the robot behavior program from scratch (i.e., without a given buggy tree or, simply, an empty tree) that works correctly in the given situation. This task is inspired by the task in the Future Convenience Store Challenge (FCSC)[1]. Unlike the two previous tasks, this task aims to examine whether non-expert users are able to utilize my proposed system and debugging features. Figure 4.7 is the example solution for the third task. The situation in this task is:

- If a human is in the robot working area (i.e., within one meter from the robot), the robot should say the following sentences and halt the current task:
  1. "Please wait a moment."
  2. "I will clear the space."
- Once the human leaves the robot's working area, the robot should say "I will resume my work," then continue the task.

[1]FCSC, https://wrs.nedo.go.jp/en/wrs2020/challenge/service/fcsc.html

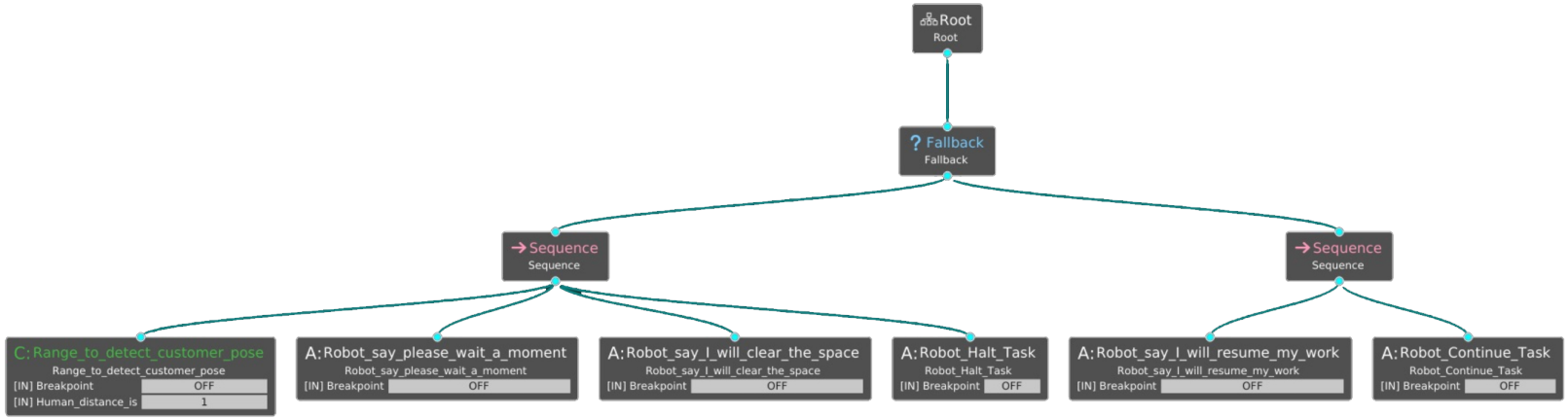


Figure 4.7. Example of solutions for the third task. To solve this task, the subjects have to understand the concept of the robot behavior programs for creating the robot behavior program and use the debugging functionalities to solve issues based on the situation.

4.4 Evaluation

To evaluate the effectiveness of the proposed system, I consider four criteria during the experiment with each subjects: (i) the correctness of the robot behavior program, (ii) the number of attempts to finish the task, (iii) the time to finish the task, and (iv) the usability of the system interface (i.e., SUS score). First, I consider that the robot behavior program is correct if it can be executed exactly the same as stated in each task situation. Note that it is possible to have different solutions to fix the robot behavior programs. If the robot behavior program is correct, I count the number of attempts and measure the time that non-expert users spent to run the simulator before finishing a task.

Table 4.4. Ten questions of the System Usability Scale for evaluating a robot behavior creation system. Participants have to give a score from 1 (strongly disagree) to 5 (strongly agree).

	SUS Question list
1	I think that I would use this interface frequently.
2	I found this interface unnecessarily complex.
3	I thought the interface was easy to use.
4	I would need the help of a technical person to be able to use this interface.
5	I found the various functions provided by this interface to be well integrated.
6	I thought the interface design was too inconsistent.
7	I imagine most people would learn to use this interface very quickly.
8	I found this interface cumbersome to use.
9	I feel very confident in using this interface.
10	I need to learn a lot of things before I could be an effective user of this interface.

For evaluating the usability of the system interface, I use the System Usability Scale (SUS) [52]. The SUS contains a set of ten questions with five-points Likert scale (Strongly disagree: 1, Disagree: 2, Neutral: 3, Agree: 4, Strongly agree: 5) to obtain feedback about the system as shown in Table 4.4. After each subject answered all of the questions, I calculated the SUS score as follows:

- For odd questions: subtract 1 from the response.

- For even questions: subtract the response from 5.
- Calculate the sum of scores from ten questions and then multiply it by 2.5.

In the end, I get the SUS score for each subject in the range of 0 to 100. The interpretation of the SUS score provided by Bangor et al. [53, 54] can be found in Figure 5.6. Additionally, I asked subjects to give their opinions and suggestions for improving my proposed system.

For the experiment for robot behavior debugger, I also calculated the unpaired t-test [55] to find whether the results of the three criteria and SUS score for with breakpoint and without breakpoint groups have significant differences. Note that I consider that the results are significantly different when the p-value is less than 0.01.

Summary

- The objective of the first experiment is to evaluate whether non-expert users understand the concept of the robot behavior program with graduate students who do not belong to the information science division and do not have an experience with visual programming.
- The objective of the second experiment is to evaluate whether the proposed debugging features help non-expert users to identify issues in the robot behavior program, especially the breakpoint feature.
- The second experiment is an extension of the first one as recruited non-expert users are not graduate students and are more likely to satisfy the requirement of a convenience store staff job.
- There are four criteria for the system evaluation with each subject: (i) the correctness of the robot behavior program, (ii) the number of attempts to finish the task, (iii) the time to finish the task, and (iv) the usability of the system interface (SUS score).

CHAPTER 5

Results

This chapter presents the results of the experiments with non-expert users in two sections. Section 5.1 shows the results of the experiment for evaluating the robot behavior creator and the robot simulator. Section 5.2 shows the results of the experiment for evaluating the robot behavior debugger with a comparison of breakpoint usages.

5.1 Experimental results of robot behavior creator

Table 5.1 shows the number of subjects who can fix and create the Behavior Trees correctly for each task. For the evaluation, most of non-expert users can finish every tasks within 1 attempt. In average, they spent around 2 minutes to finish first and

Table 5.1. Number of subjects that are able to fix and create the Behavior Trees correctly for each task.

Tasks	# Participants				Average time [m's"]
	# Attempts of pass			# Attempts of fail	
	1	2	3		
Task1 - Fix	8	1	1	0	2' 4"
Task2 - Create (single)	10	0	0	0	2' 5"
Task3 - Create (multiple)	8	1	0	1	7' 10"

second tasks, while took around 7 minutes for the third tasks. Note that there is only one participant that failed to finish the third task during the experiment time.

Figure 5.1 shows the number of attempts that participants took for finishing our experiment (i.e., fixing a buggy Behavior Tree and creating Behavior Trees from scratch). I found that most of the participants were able to finish three tasks with only one attempt within approximately five minutes. For the first task about fixing a buggy Behavior Tree, there are two participants who took more than one attempt to finish the task. For the third task about creating a Behavior Tree from scratch, only one participant took two attempts to finish the task. The reasons behind these mistakes are due to the confusion with the question statements and the GUI of Groot. However, after carefully testing and debugging with the provided simulation, they were able to finish the given tasks. These results indicate that participants could utilize the robot simulation for debugging their Behavior Trees.

Figure 5.2 shows the distribution of SUS scores from all participants. I found that the average SUS score (i.e., Median) is 77.5 out of 100 with a standard deviation of 6.26. According to SUS score interpretation from Bangor et al. [54], our proposed system is considered to have a "good" usability.

From the interview with participants about the usage of the proposed system, participants commented that the simulation helps them to test and find the issue of the desired robot behavior. This is because our simulation can show a reaction of a robot when it detects a human based on the created behavior. For example, one participant said that "*figuring the issue point from previously created Behavior Tree can be supported by the simulation.*" The other participant also supported that the simulation "*makes me more confident for the Behavior Tree that I had created.*"

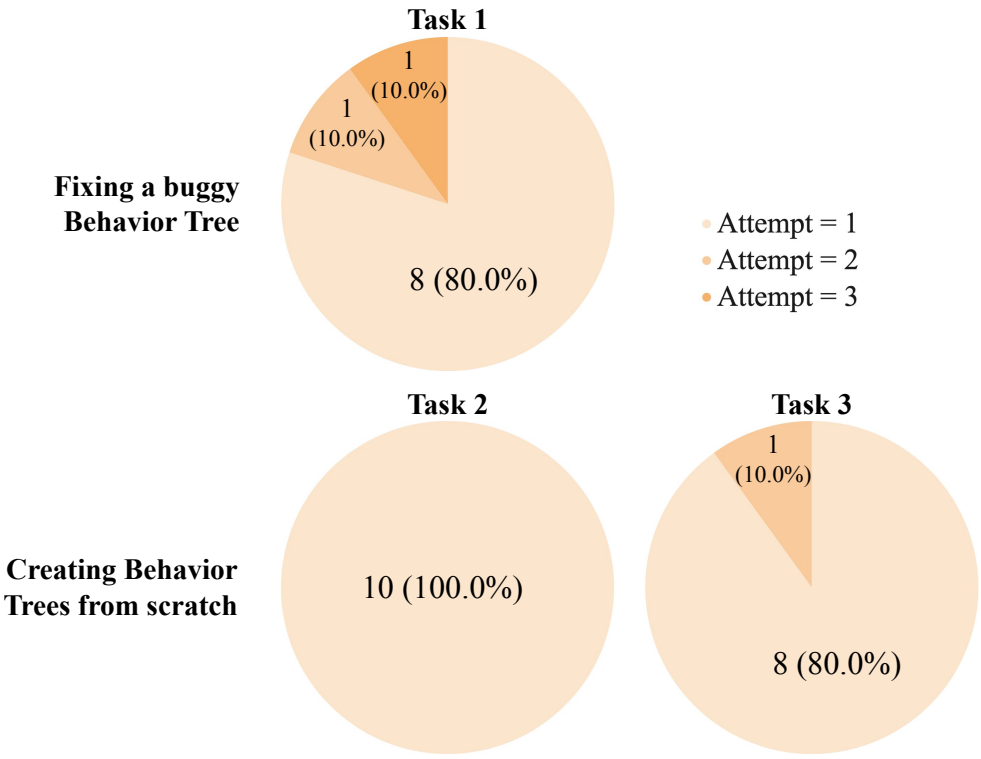


Figure 5.1. Numbers of attempts that users spent during the experiments

Additionally, participants also commented that the simulation helps them to understand the concept of robot behavior creation. This is because participants can check the order of robot reaction described in the Behavior Tree from the simulation. For example, they always "*got confused with other computer programs or computer languages*" and *"need time for understanding a new concept*", so they thought that "*simulator greatly helps to check what happens to program.*"

However, the system still lacks some features as suggested by participants. The first point is that the system lacks a proper logging system to show what is happening in the simulation. For example, the proposed system should have the "*error warning*" in the case that the robot cannot work properly in the simulation (e.g., cannot generate sounds or collides with the shelf). By providing the logging system, non-expert users can understand the situation inside the simulation through the logs.

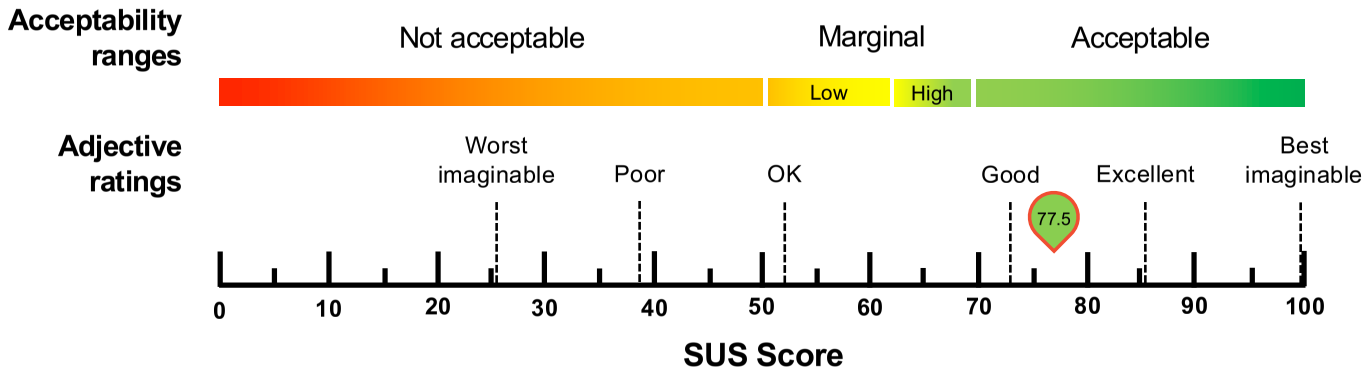


Figure 5.2. Distribution of the result from SUS score.

The second point is that the system lacks the proper connection between the robot behavior system and the robot simulation. Specifically, the system cannot show the executing node in the simulation in the Behavior Tree. For example, participants suggested that the system should "*shows the current executing node in the Behavior Tree along with the simulation*" and should be able to "*stop the loop*" of the actions in the Behavior Tree to see how the robot reacts to the human step-by-step. By providing such a feature, non-expert users can easily know the current status of robot behaviors in the simulation.

In summary, our proposed system helps non-expert users to create robot behaviors by using Behavior Tree and to test created robot behaviors by using the robot simulation. I found that the proposed system is considered to have good usability. From the participant feedback, the simulation can help non-expert users to confirm the correctness of robot behaviors and to learn the Behavior Tree. For improvement, I plan to implement a logging system to show errors and warning messages from the simulation. I also plan to implement the feature to track the current executing node in the Behavior Tree.

5.2 Experimental results of robot behavior debugger

Table 5.2 shows the number of subjects who can fix and create the Behavior Trees correctly for each task. I find that every subject from both with breakpoint and without breakpoint groups can finish the first task correctly. However, only subjects from the without breakpoint group are able to finish the third task without failing.

Table 5.2. Number of subjects that are able to fix and create the Behavior Trees correctly for each task.

	Task 1	**Task 2**	**Task 3**
With breakpoint	7	5	6
Without breakpoint	7	5	7

The results show that the differences between with breakpoint group and without breakpoint group are not significant in terms of the number of attempts and time to finish the task. Figure 5.3 shows that subjects from both groups spent only one attempt to finish the first task and a maximum of three attempts for the second task. The reason for this result is that some subjects identify issues in the programs by manually checking the Behavior Trees instead of using breakpoints, so they required more than one attempt to complete the tasks. Figure 5.4 shows that subjects from without breakpoint group spent slightly less time than the with breakpoint group in the first task (i.e., fixing condition nodes) and the third task (i.e., creating a new Behavior Tree). However, in the case of the second task (i.e., fixing control nodes and reordering nodes), subjects from with breakpoint group spent less time to finish the task (i.e., 823 $<$ 1,022 seconds). According to the subjects, the second task is more complicated than others, so the breakpoint is useful for tracing the issue in the robot behavior program.

Figure 5.5 shows the distribution of SUS scores of subjects from with breakpoint and without breakpoint groups. I find that the mean SUS score from with breakpoint group is less than without breakpoint group (i.e., 66.07% $<$ 71.43%). According to Figure 5.6, the acceptability ranges of our proposed system with breakpoint feature belong to high marginal level of usability and are acceptable for without breakpoint feature.

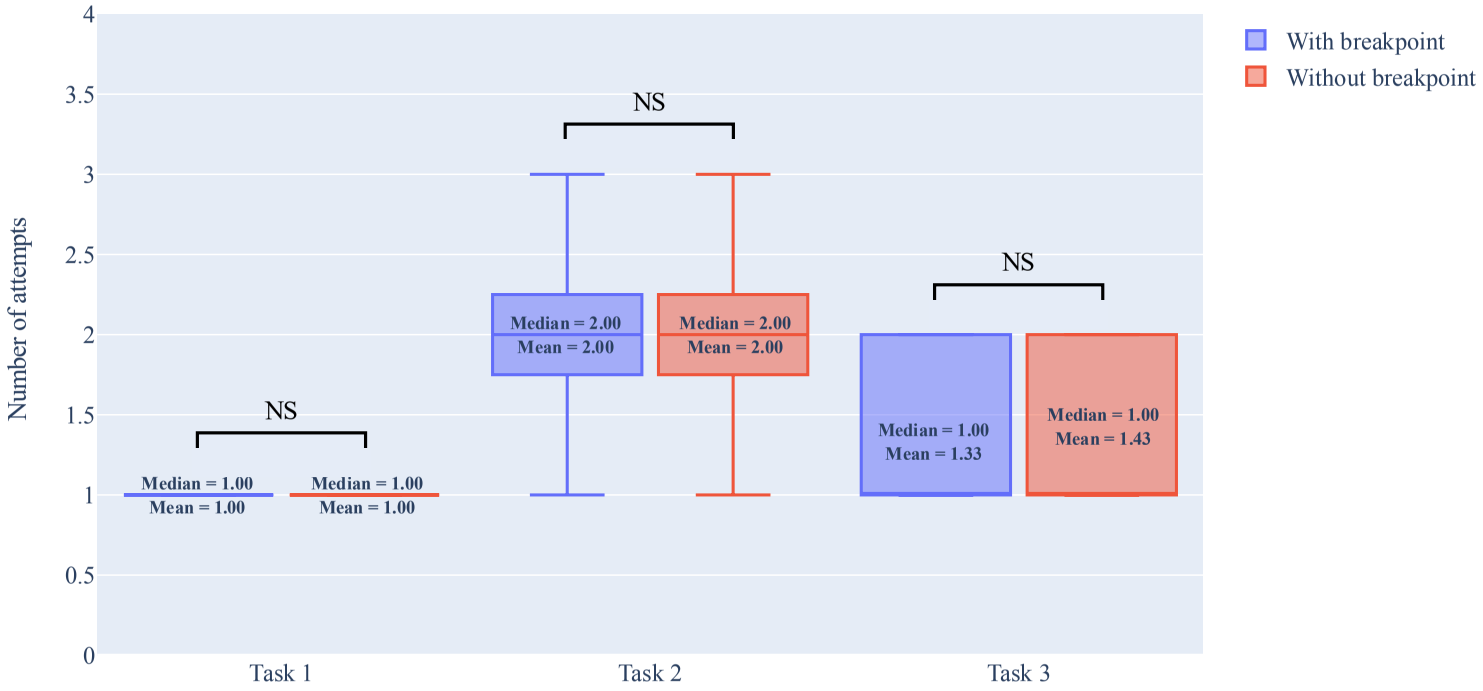


Figure 5.3. Number of attempts to finish each task from both experiments

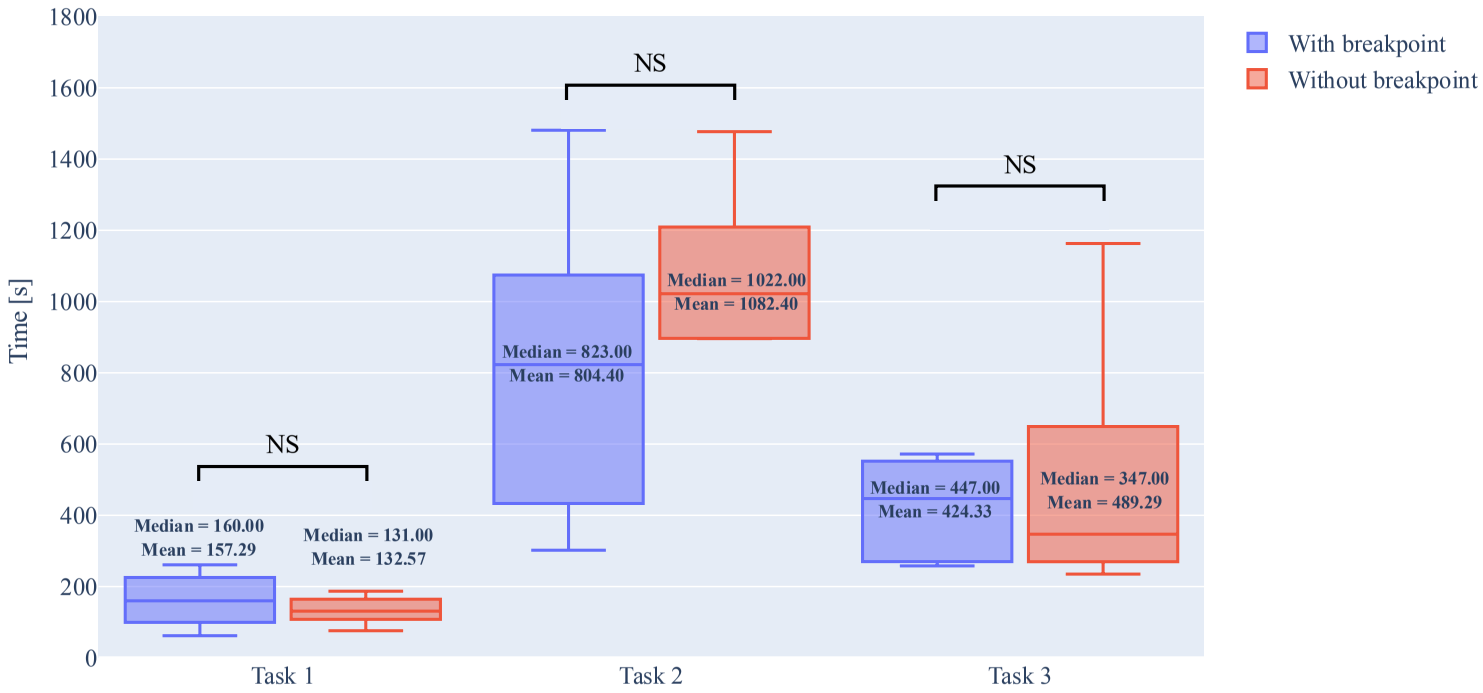


Figure 5.4. Time to finish each task from both experiments

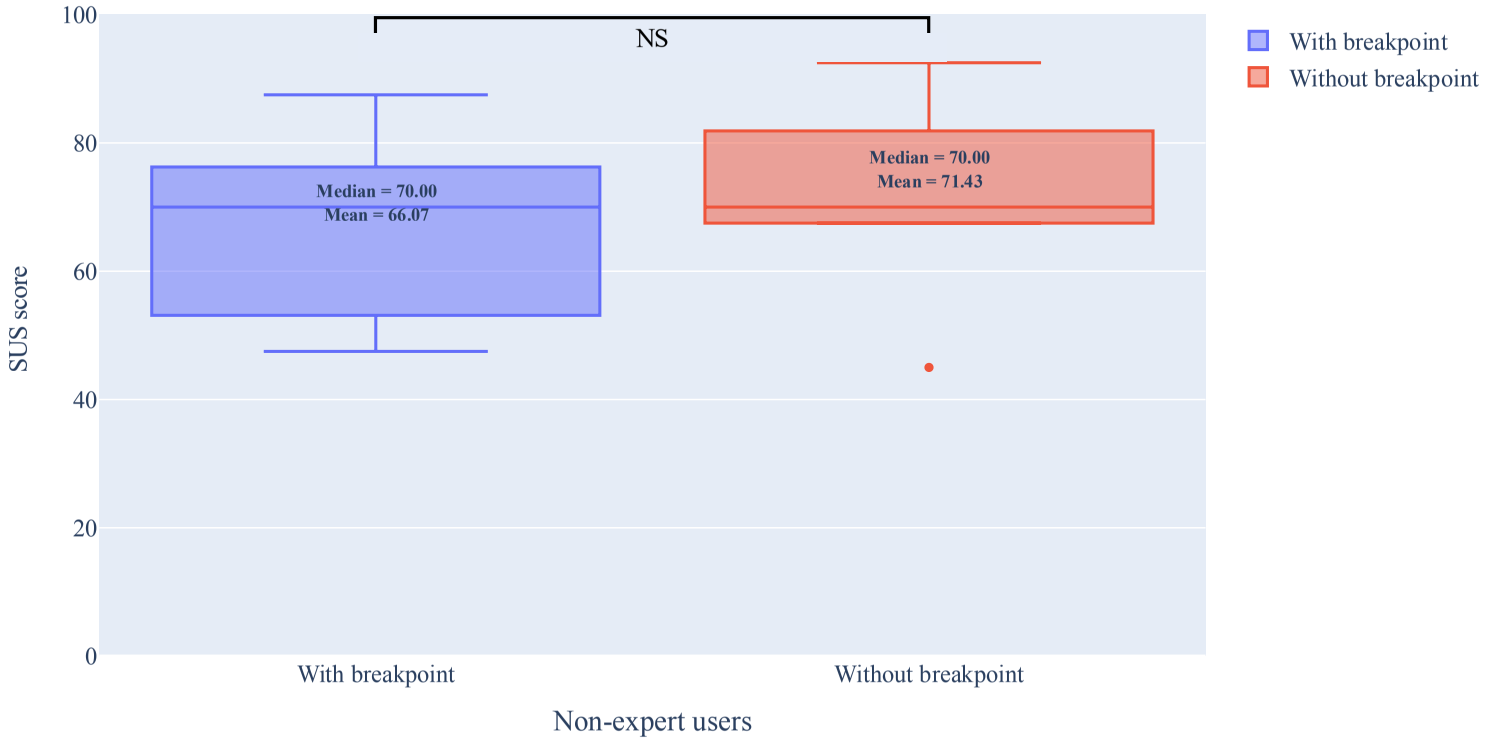


Figure 5.5. Distribution of the result from SUS score from both experiments

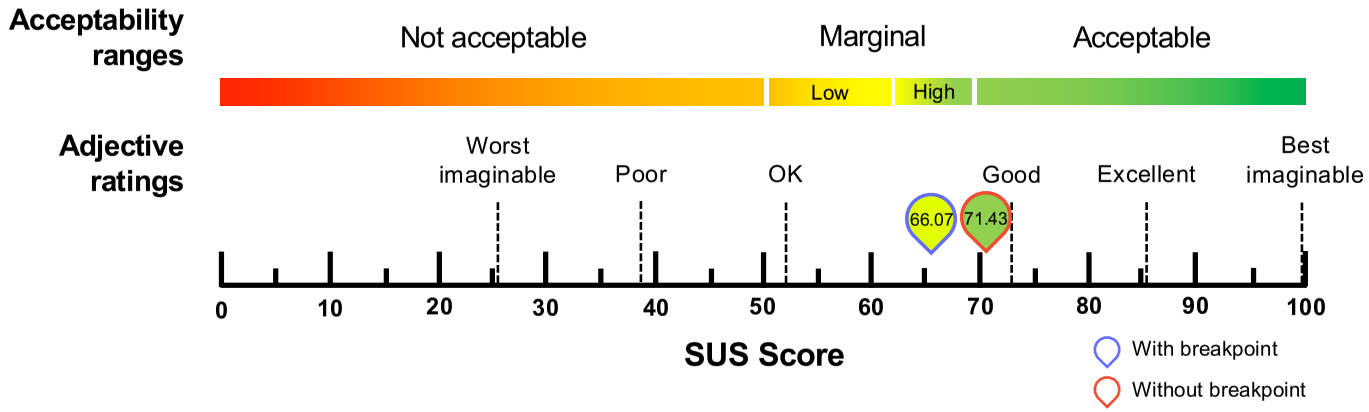


Figure 5.6. Mapping of the SUS score and the adjective rating.

Summary

- Non-expert users are able to create and fix the robot behavior program with only one subject failing to complete tasks by using the robot behavior creator.
- The robot behavior creator has good usability based on the SUS score.
- Only few subjects failed to identify and fix the robot behavior program by using the proposed debugger.
- The robot behavior debugger has high marginal usability based on the SUS score.

CHAPTER 6

Discussion

This chapter addresses a discussion and lessons learned from the experimental results and interviews with non-expert users. The discussion is separated into five points as follows.

Non-expert users are able to utilize our proposed system. Based on the experimental results, there are only a few subjects that failed to create and fix issues within the robot behavior program. As shown in Tables 5.1 and 5.2, only one subject failed to finish tasks for the first experiment and four subjects failed to finish tasks for the second experiment (i.e., two subjects per group). This confirms that non-expert users could understand the concept of the Behavior Trees and use our proposed system to create and modify the robot behavior program within the time limit. Moreover, the comments from the subjects also indicated that using our system is *like playing a video game*, so it seems to be attractive when the program is visually displayed. Hence, we conclude that the interactive debugging functionality such

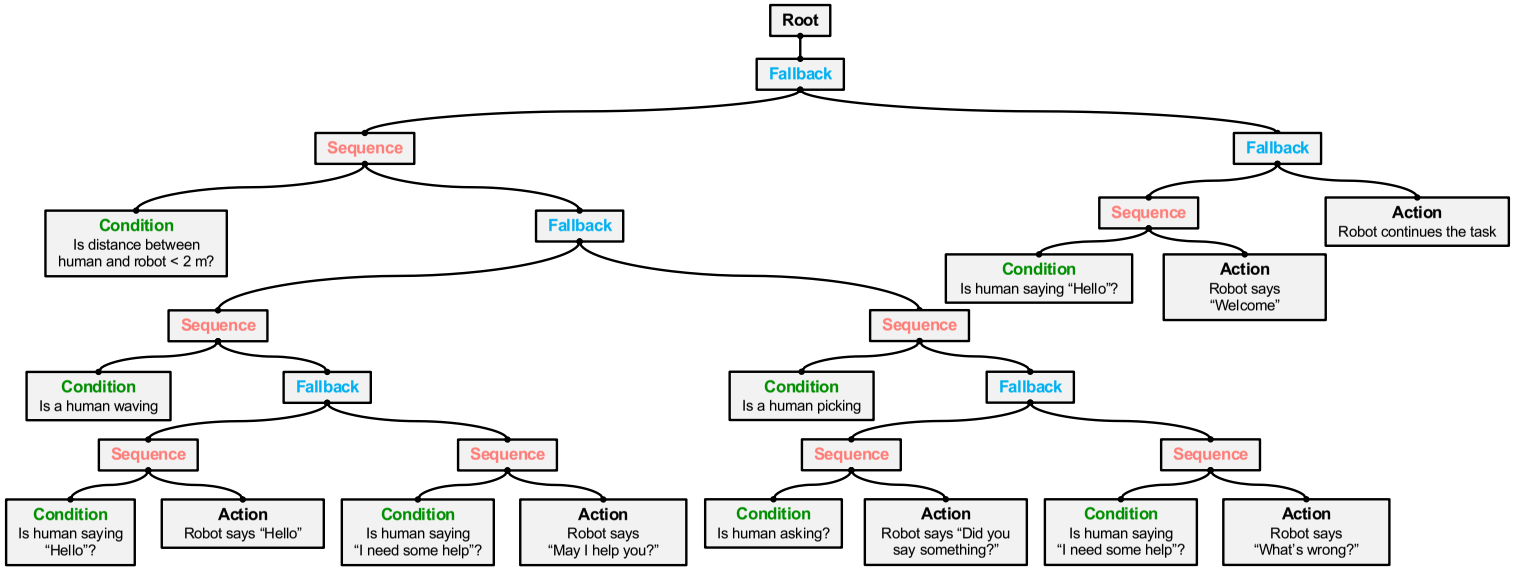


Figure 6.1. Examples of a complex robot behavior program that contains various nodes and branches.

as node execution monitoring with the proposed robot behavior creator is suitable for non-expert users who do not have experience with any programming languages. Despite the effectiveness of the proposed robot behavior creator and debugger, one subject raised a concern related to the interface of the proposed system, In the case of complex behavior, the robot behavior program may contain lots of nodes and branches, which may confuse non-expert users as shown in Figure 6.1. One potential future work to address this issue is by improving the interface of the creator by allowing users to group several nodes to reduce the complexity of the program.

Non-expert users did not utilize the breakpoint feature to fix Behavior Tree issues. From the experimental result of the robot behavior debugger, I found that the breakpoint feature does not significantly improve the effectiveness of debugging the robot behavior program. As shown in Figure 5.4, the difference in the time to finish each task between the with-breakpoint and without-breakpoint groups is not significant. Based on the SUS score from Figure 5.5, the score of the with-breakpoint group is slightly less than the without-breakpoint group (i.e., 66.07 % $<$ 71.43 %). This indicates that the breakpoint feature may confuse non-expert users, even though it is considered a high marginal level of usability. From the observations and interviews with subjects, I found that only one was able to use the breakpoint feature to debug the robot behavior program. One subject mentioned that they could solve the issues by only checking the Behavior Tree monitoring interface, so they did not try to use the breakpoint feature. Another example comes from subjects who were unable to finish the tasks. They commented that, in the second task, the order of nodes is confusing. However, they did not know how to use the breakpoint feature to identify the issue. Hence, we conclude that the non-expert users did not understand how to utilize the breakpoint feature to fix the issues within the Behavior Trees. We suggest that giving more examples on how to use the breakpoint feature usage could help non-expert users to use this feature. We also suggest that simplifying the breakpoint feature can also improve the usability of the system.

Visualization helps non-expert users address the issue. As mentioned by one subject from the robot behavior debugger experiment, the log status is hard to see because *there are so many lines*. Others subjects commented that they mostly used the information from the monitoring feature and the simulation during the ex-

periment. The reason is that these features are easier to understand and observe than the log which is only text information. Hence, we conclude that visual information is easier to understand than textual information. The future direction of the research is to increase the effectiveness of the debugging functionality by using visual information.

Impact of the subjects' characteristics. From the experimental results shown in Figures 5.2 and 5.6), the average SUS score from the robot behavior creator experiment is greater than the robot behavior debugger one (i.e., 77.50 % > 66.07 % for with-breakpoint and 71.43 % without-breakpoint). I suspect that the background and characteristics of subjects impact how they solve tasks in the experiments. For the robot behavior creator experiment, the subjects are graduate students (i.e., master's and doctoral students), so their logical thinking might be stronger than other people. While the subjects from the robot behavior debugger experiment are a mixture of different jobs. From the interviews, subjects also have different hobbies which might impact their thinking process. For example, the participant who plays Shogi (i.e., Japanese chess) is the only person that could finish all tasks within one attempt. Another example from the without-breakpoint group is the participant who plays sports (e.g., American football and baseball) could finish all tasks in the shortest time (i.e., 21'36") while the average time in the same group is 29'28". Hence, they might have better logical thinking than other subjects which results in less time or fewer attempts to finish the task as shown in Figure 5.3 and 5.4. To confirm the impact of subjects' backgrounds, further investigation is needed.

Some types of Behavior Tree nodes are not required in the robot behavior program. From Marzinotto et al. [32], there are six types of Behavior Tree nodes that can be used to create a program including (i) condition node, (ii) action node, (iii) sequence node, (iv) fallback node (i.e., selector node), (v) parallel node, and (vi) decorator node. The most essential nodes for creating a program are the condition node and the action node, which the robot uses to make a decision and respond to a human. To make a robot performs more complex actions, some additional nodes are needed for considering whether to continue or skip a task. Hence, in the dissertation, I decided to use the sequence node and the fallback node because the concepts of these nodes are straightforward to decide whether to continue as long as

the node status is successful or to try until the node status is successful, respectively. The concepts of the parallel node and the decorator node are redundant to the sequence node and the fallback node. The parallel node allows the robot to consider multiple actions and conditions at the same time, while the decorator node allows the robot to behave based on certain conditions or the result of actions. In conclusion, in this dissertation, I selected four types of nodes with the least complexity in order to make it easier for non-expert users to create their own programs.

Summary

- The simulator greatly helps non-expert users understand how the robot behavior program works.
- The concept of breakpoint confuses non-expert users and does not significantly improve the effectiveness of the proposed system.
- Visualization is more preferable for debugging than monitoring logs.
- The background and characteristic of subjects affect how they use the proposed system.
- Parallel and decorator nodes are not required in the robot behavior program as four other nodes can represent the robot's decision-making.

CHAPTER 7

Conclusion

7.1 Summary

This dissertation proposes a system that can support non-expert users in creating, testing, and debugging robot behavior programs without robot programming experience. The proposed system consists of (i) a robot behavior creator with a robot simulator and (ii) a robot behavior debugger. To support non-expert users in making a program, it is crucial to provide an easy-to-use user interface and an easy-to-understand concept to make a program. Therefore, this research proposes an approach to create a program mimicking the human decision-making concept [49]. This approach requires considering conditions for making decisions and corresponding actions. The robot behavior creator provides visual information through Behavior Trees GUI which is a tree-like structure to represent the conditions and actions of the robot. Moreover, the system provides drag-and-drop composition to assist the

users in creating the program by making pairs of conditions and actions. The users can test their robot behavior program by using the robot simulator. From the evaluation with ten non-expert users, I found that the proposed robot behavior creator and robot simulator can help non-expert users to create and test the robot behavior program. The robot behavior creator is evaluated with ten non-expert users to find whether those subjects can create the robot behavior program with the GUI. The robot behavior debugger consists of four features, including (i) breakpoint, (ii) node execution monitoring, (iii) execution log, and (iv) robot variables. From the evaluation with 14 non-expert users, I found that the proposed debugger can help them to identify and fix the robot behavior program. However, a breakpoint was highlighted by non-expert users as they were confused about its usage.

7.2 Opportunities for Future Research

This dissertation proposes a Human-Robot Interaction system for non-expert users to create, test, and debug robot behavior using visual programming. However, there are still a lot of research aspects that can be done in order to improve usability for non-expert users. The potential research opportunities are addressed as follows:

An investigation of the impact of non-expert user background to the visual programming. Based on the results in Chapter 5, non-expert users with different backgrounds tend to have different ways of thinking to create, test, and debug the robot behavior program. However, the impact of non-expert user backgrounds on programming efficiency has yet to explore. I suggest that future researchers should investigate this point in order to understand how non-expert user program a robot and how to improve the proposed system in the case that users do not have strong logical thinking skills.

An adaptation of the proposed system in different robot working environments. As detailed in Chapters 1 and 3, the current version of the proposed system focuses on the robot that operates in the convenience store. The proposed system also provides a simulated convenience store environment that includes human avatars and robots. In fact, both robot behavior creator and robot behavior debugger can be adapted and used in other HRI scenarios such as different robot

working environments. Hence, one of the immediate future works is to implement the environment of the simulator for various HRI scenarios.

An improvement of the interface of the robot behavior creator. From the discussion with non-expert users in Chapter 6, they have a concern in the case that when the robot behavior program is very complex, it can be very hard to read the Behavior Tree in the proposed robot behavior creator. One possible solution to this issue is by improving the interface of the robot behavior creator to hide some parts of the Behavior Tree. Non-expert users should be able to toggle the desired nodes for showing or hiding sub-trees.

An enhancement of the visual debugger. Based on the comments from non-expert users in Chapter 6, they agree that visual debugging features such as node execution monitoring is more preferable than textual logs. In order to make a tool that is more simple for non-expert users, I suggest that future researchers can design and convert all text-based debugging features to more interactive visual-based debugging features.

A simplified version of the breakpoint feature. From the results in Chapter 5, there is only one non-exert user who can fully utilize the breakpoint feature to debug the issue in the robot behavior program. Based on other comments in Chapter 6, non-expert users reveal that they do not understand the mechanism of the breakpoint and do not know how to use it in debugging. From the expert point of view, the breakpoint feature is one of the most used debugging features for typical software engineering projects [40]. I still believe that the breakpoint feature is still useful for non-expert users. However, simplification is needed in order to attract those users who totally have no experience in programming. Hence, the immediate future work is to design and implement a simplified version of the breakpoint feature. This also includes more subjective evaluations with non-expert users.

References

[1] Keiichiro Hamaguchi. How have japanese policies changed in accepting foreign workers? *Japan Labor Issues*, 3(14):2–7, 2019.

[2] Eric L Hsu. Robots as means to address the challenges of an ageing population. *The Routledge Social Science Handbook of AI*, page 96, 2021.

[3] Daniel Belanche, Luis V. Casaló, Carlos Flavián, and Jeroen Schepers. Service robot implementation: a theoretical framework and research agenda. *The Service Industries Journal*, 40(3-4):203–225, October 2019.

[4] Hiroyuki Okada, Tetsunari Inamura, and Kazuyoshi Wada. What competitions were conducted in the service categories of the World Robot Summit? *Advanced Robotics*, 33(17):900–910, September 2019.

[5] Kazuyoshi Wada. New robot technology challenge for convenience store. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, December 2017.

[6] Pattaraporn Tulathum, Bunyapon Usawalertkamol, Gustavo Alfonso Garcia Ricardez, Jun Takamatsu, Tsukasa Ogasawara, and Kenichi Matsumoto. Human-robot interaction system for non-expert users in convenience stores using behavior trees. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 1072–1077, 2022.

[7] Pattaraporn Tulathum, Bunyapon Usawalertkamol, Gustavo Alfonso Garcia Ricardez, Jun Takamatsu, Tsukasa Ogasawara, and Kenichi Matsumoto. Robot behavior debugger for non-expert users in convenience stores using behavior trees. *Advanced Robotics*, 36(17-18):951–966, 2022.

[8] Adriana Tapus, Cristian Ţăpuş, and Maja J. Matarić. User—robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy. *Intelligent Service Robotics*, 1(2):169–183, February 2008.

[9] Yusuke Kato, Takayuki Kanda, and Hiroshi Ishiguro. May i help you?-design of human-like polite approaching behavior. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 35–42, 2015.

[10] Tian-Miao Wang, Yong Tao, and Hui Liu. Current researches and future development trend of intelligent robot: A review. *International Journal of Automation and Computing*, 15(5):525–546, April 2018.

[11] Hiroaki Masuzawa, Jun Miura, and Shuji Oishi. Development of a mobile robot for harvest support in greenhouse horticulture — person following and mapping. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, December 2017.

[12] Vinh Nhat Lu, Jochen Wirtz, Werner H. Kunz, Stefanie Paluch, Thorsten Gruber, Antje Martins, and Paul G. Patterson. Service robots, customers and service employees: what can we learn from the academic literature and where are the gaps? *Journal of Service Theory and Practice*, 30(3):361–391, April 2020.

[13] Gabriele Obermeier and Andreas Auinger. Human-computer interaction in physical retail environments and the impact on customer experience: Systematic literature review and research agenda. In *HCI in Business, Government and Organizations. eCommerce and Consumer Behavior*, pages 51–66. 2019.

[14] Chao Shi, Satoru Satake, Takayuki Kanda, and Hiroshi Ishiguro. How would store managers employ social robots? In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2016.

[15] Yoha Oishi, Takayuki Kanda, Masayuki Kanbara, Satoru Satake, and Norihiro Hagita. Toward end-user programming for robots in stores. In *Proceedings of the Companion of*

the 2017 ACM/IEEE International Conference on Human-Robot Interaction, March 2017.

[16] Amy J Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, et al. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, 43(3):1–44, 2011.

[17] Barbara Rita Barricelli, Fabio Cassano, Daniela Fogli, and Antonio Piccinno. End-user development, end-user programming and end-user software engineering: A systematic mapping study. *Journal of Systems and Software*, 149:101–137, 2019.

[18] Barbara Rita Barricelli, Daniela Fogli, and Angela Locoro. Eudability: A new construct at the intersection of end-user development and computational thinking. *Journal of Systems and Software*, 195:111516, 2023.

[19] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. Addressing accessibility barriers in programming for people with visual impairments: A literature review. *ACM Transactions on Accessible Computing (TACCESS)*, 15(1):1–26, 2022.

[20] Chris Paxton, Andrew Hundt, Felix Jonathan, Kelleher Guerin, and Gregory D. Hager. CoSTAR: Instructing collaborative robots with behavior trees and vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[21] Chris Paxton, Felix Jonathan, Andrew Hundt, Bilge Mutlu, and Gregory D. Hager. Evaluating methods for end-user creation of robot task plans. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[22] David Weintrop. Block-based programming in computer science education. *Communications of the ACM*, 62(8):22–25, July 2019.

[23] Pierre A. Akiki, Paul A. Akiki, Arosha K. Bandara, and Yijun Yu. EUD-MARS: End-user development of model-driven adaptive robotics software systems. *Science of Computer Programming*, 200:102534, December 2020.

[24] Christoph Mayr-Dorn, Mario Winterer, Christian Salomon, Doris Hohensinger, and Harald Fürschuss. Assessing industrial end-user programming of robotic production cells: A controlled experiment. *Journal of Systems and Software*, 195:111547, 2023.

[25] Anthony Savidis. Programming experience requirements for future visual development environments. In Mutlu Cukurova, Nikol Rummel, Denis Gillet, Bruce M. McLaren, and James Uhomoibhi, editors, *Proceedings of the 14th International Conference on Computer Supported Education (CSEDU)*, pages 284–292. SCITEPRESS, 2022.

[26] Enrique Coronado, Fulvio Mastrogiovanni, and Gentiane Venture. Development of intelligent behaviors for social robots via user-friendly and modular programming tools. In *2018 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, September 2018.

[27] Stephen Balakirsky, Zeid Kootbally, Craig Schlenoff, Thomas Kramer, and Satyandra Gupta. An industrial robotic knowledge representation for kit building applications. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1365–1370. IEEE, 2012.

[28] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. A survey of behavior trees in robotics and ai. *Robotics and Autonomous Systems*, 154: 104096, 2022.

[29] Miguel Nicolau, Diego Perez-Liebana, Michael O' Neill, and Anthony Brabazon. Evolutionary behavior tree approaches for navigating platform games. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):227–238, 2016.

[30] Ryan Marcotte and Howard J Hamilton. Behavior trees for modelling artificial intelligence in games: A tutorial. *The Computer Games Journal*, 6(3):171–184, 2017.

[31] Francesco Rovida, Bjarne Grossmann, and Volker Krüger. Extended behavior trees for quick definition of flexible robotic tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6793–6800. IEEE, 2017.

[32] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ogren. Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014.

[33] Kevin French, Shiyu Wu, Tianyang Pan, Zheming Zhou, and Odest Chadwicke Jenkins. Learning behavior trees from demonstration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7791–7797, 2019.

[34] David John Barnes, Michael Kölling, and James Gosling. *Objects First with Java: A practical introduction using BlueJ*. Pearson/Prentice Hall, 2006.

[35] Joseph Lawrance, Christopher Bogart, Margaret Burnett, Rachel Bellamy, Kyle Rector, and Scott D Fleming. How programmers debug, revisited: An information foraging theory perspective. *IEEE Transactions on Software Engineering*, 39(2):197–215, 2010.

[36] Roman Bednarik. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies*, 70(2):143–155, 2012.

[37] Bryce Ikeda and Daniel Szafir. An AR Debugging Tool for Robotics Programmers. 2021.

[38] Bryce Ikeda and Daniel Szafir. Advancing the design of visual debugging tools for roboticists. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*, pages 195–204, 2022.

[39] Eclipse desktop & web ides | the eclipse foundation. https://www.eclipse.org/ide/, 2022. (Accessed on 03/18/2022).

[40] G.C. Murphy, M. Kersten, and L. Findlater. How are java software developers using the eclipse ide? *IEEE Software*, 23(4):76–83, 2006.

[41] Moritz Beller, Niels Spruit, Diomidis Spinellis, and Andy Zaidman. On the dichotomy of debugging behavior among programmers. In *Proceedings of the 40th International Conference on Software Engineering*, pages 572–583, 2018.

[42] Michael Perscheid, Benjamin Siegmund, Marcel Taeumel, and Robert Hirschfeld. Studying the advancement in debugging practice of professional software developers. *Software Quality Journal (SQJ)*, 25(1):83–110, 2017.

[43] Valentina Grigoreanu, Margaret Burnett, Susan Wiedenbeck, Jill Cao, Kyle Rector, and Irwin Kwan. End-user debugging strategies: A sensemaking perspective. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(1):1–28, 2012.

[44] Ren Manfredi, Margherita Andrao, Francesco Greco, Giuseppe Desolda, Barbara Treccani, and Massimo Zancanaro. Toward a better understanding of end-user debugging strategies: A pilot study. In *Proceedings of the 3rd International Workshop on Empowering People in Dealing with Internet of Things Ecosystems co-located with International Conference on Advanced Visual Interfaces (AVI)*, volume 3172, pages 31–35, 2022.

[45] Jill Cao, Kyle Rector, Thomas H Park, Scott D Fleming, Margaret Burnett, and Susan Wiedenbeck. A debugging perspective on end-user mashup programming. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 149–156, 2010.

[46] Jane Hoffswell, Arvind Satyanarayan, and Jeffrey Heer. Visual debugging techniques for reactive data visualization. In *Computer Graphics Forum*, volume 35, pages 271–280, 2016.

[47] Sandro Tolksdorf, Daniel Lehmann, and Michael Pradel. Interactive metamorphic testing of debuggers. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 273–283, 2019.

[48] Miguel Campusano and Alexandre Bergel. VizRob: Effective visualizations to debug robotic behaviors. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, February 2019.

[49] Artificial intelligence and human decision making. *European Journal of Operational Research*, 99(1):3–25, 1997.

[50] Michele Colledanchise and Petter Ögren. *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

[51] Cheng Zhang, Dacong Yan, Jianjun Zhao, Yuting Chen, and Shengqian Yang. Bpgen: an automated breakpoint generator for debugging. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 271–274, 2010.

[52] J. Brook. SUS: A 'quick and dirty' usability scale. In *Usability Evaluation In Industry*, pages 207–212. CRC Press, June 1996.

[53] Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human- Computer Interaction*, 24 (6):574–594, 2008.

[54] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of Usability Studies*, 4(3): 114-123, May 2009.

[55] Karen K Yuen. The two-sample trimmed t for unequal population variances. *Biometrika*, 61(1):165–170, 1974.