Improving Human-Robot Collaboration through Abstraction and Scaffolding in Planning and Programming Interfaces

by

Nathan Thomas White

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2025

Date of final oral examination: 04/30/2025

The dissertation is approved by the following members of the Final Oral Committee:

Bilge Mutlu, Professor, Computer Science, UW–Madison
Aws Albarghouthi, Professor, Computer Science, UW–Madison
Robert Radwin, Professor, Industrial & Systems Engineering, UW–Madison
Christopher Reid, Technical Fellow, Human Factors & Ergonomics, Boeing

For my parents: Jerry & Janice

ACKNOWLEDGMENTS

When I applied to graduate school, I never thought I would be working with robots again. In high school, working with a team in the VEX robotics competition, I grew incredibly interested in artificial intelligence and automation. This led me to pursue an undergraduate degree in Computer Science at the University of Minnesota–Twin Cities, where I learned about artificial intelligence, machine learning, and robotics. I would not have started this PhD journey without the help of Dr. Maria Gini and Dr. John Harwell, who sparked my interest in research through projects in swarm robotics and supported me in pursuing research further.

Completing a PhD is not an easy task. The saying "it takes a village" truly applies, as many people helped to support me in this journey. I thank my committee members: Dr. Bilge Mutlu, Dr. Robert Radwin, Dr. Aws Albarghouthi, and Dr. Christopher Reid for supporting my journey and growth. A huge thank you goes to my advisor, Dr. Bilge Mutlu. After joining the lab, I quickly grew enamored with understanding how to support people's interactions with robots. You embraced my constant curiosity and supported me in exploring multiple research areas and directions, allowing me to experiment across many different interests. I am grateful for your advice and guidance, always pushing me to pursue interesting research topics and collaborations. You have cultivated a truly remarkable culture and group of individuals within the lab, resulting in some of the most interesting and enjoyable work that I have gotten to be a part of.

I would not be where I am now without several of my phenomenal collaborators and friends: Dr. Andrew Schoen, Dr. Pragathi Praveena, Bengisu Cagiltay, and Dakota Sullivan. I have learned so much from them throughout my PhD journey, from how to work with chaotic real-world robot deployments, to developing new systems and exploring novel

interactions. These individuals have given me a plethora of advice, helping me to improve my research skills, but also have lent me their perspectives and insight when I needed it most. I also want to thank my friends in the People and Robots Lab: Dr. Arissa Sato, Hailey Johnson, Callie Kim, Christine Lee, Hui Ru Ho, Yuna Hwang, and Mya Schroder for all the projects, events, and general camaraderie I was fortunate enough to share with you all. You have all made a huge impact on my PhD journey, making for some of the most enjoyable moments!

Finally, I want to thank my close friends and family, who have been with me through it all, both the ups and the downs. A huge thanks goes out to Ismar Mendoza, Randy Alvarez, Fernando Mosqueda, John Schweiger, and Sunny Zheng, I seriously could not have done this without all of your support and words of encouragement. No amount of words can express my gratitude for you all. Finally, my family, Jerry, Janice, Emily, and Austin, thank you for listening to me when I needed it and pushing me to continue to do my absolute best. A special thank you goes to my dad, Dr. Jerry White, who always believed in what I was capable of and pushed me to pursue my dreams.

DECLARATION

The research presented within this dissertation was funded through multiple National Science Foundation (NSF) awards, including 1822872, 1925043, 2026478, and 2152163.

While the research within this dissertation represents the author's own work, there were several additional collaborators on the ideation, implementation, evaluation, and preparation of manuscripts. These individuals include: Andrew Schoen, Dakota Sullivan, Anna Konstant, Curt Henrichs, Amanda Siebert-Evenstone, David Shaffer, Josiah Hanna, Robert Radwin, and Bilge Mutlu. Of these, several individuals contributed significantly to the work presented in this dissertation:

- *Andrew Schoen* and I worked together closely for several pieces of work presented in this dissertation, most notably in the design and implementation of the systems presented in Chapters 3 and 5. He also assisted in the real-world evaluation presented in Chapter 4.
- *Dakota Sullivan* assisted in the brainstorming of methods and evaluation presented in Chapter 4.
- *Anna Konstant* and I worked together in the design and evaluation of the system presented in Chapters 5 and 6.

Within this dissertation, I use the term 'we' to describe our collective work.

CONTENTS

Contents v

List of Tables viii

List of Figures ix

Abstract xiv

1 Introduction 1
- *1.1 Motivation* 1
- *1.2 Thesis Statement* 4
- *1.3 Methodology* 5
- *1.4 Contributions* 6
- *1.5 Dissertation Overview* 7

2 Background 9
- *2.1 Understanding the Demands of Industry* 9
- *2.2 Difficulty of Using Cobots To Fit the Needs of Industry* 11
- *2.3 Addressing Cobot Difficulty* 13
- *2.4 Supporting End-User Programming* 15
- *2.5 Chapter Summary* 15

3 CoFrame: Designing Programming Systems for Human-Robot Collaborations 17
- *3.1 Motivation* 17
- *3.2 Related Work* 19
- *3.3 System Design & Implementation* 21
- *3.4 Case Studies* 27
- *3.5 Discussion* 32
- *3.6 Chapter Summary* 34

4 Evaluating CoFrame and Identifying Supports for Human-Robot Interaction Planning 36
4.1 Motivation 36
4.2 Background 39
4.3 Evaluation Approach 42
4.4 Lab Study 43
4.5 Results of Lab Studies 45
4.6 Real-World Deployment 56
4.7 Approach 58
4.8 Case Study Application 62
4.9 Discussion 74
4.10 Chapter Summary 79

5 Allocobot: Designing Planning Systems for Collaborative Interactions 81
5.1 Motivation 81
5.2 Related Work 83
5.3 Approach 85
5.4 Implementation 90
5.5 Case Study 99
*5.6 Discussion*102
*5.7 Chapter Summary*104

6 Evaluating Allocobot as a Work Allocation and Task Planning Optimization Tool106
*6.1 Motivation*106
*6.2 Related Work*107
*6.3 Method*109
*6.4 Results*113
*6.5 Discussion*122
*6.6 Chapter Summary*124

7 General Discussion126
*7.1 Summary & Significance of Work*126
*7.2 Evaluation of Thesis*128
*7.3 Challenges & Limitations*129
*7.4 Future Work*132
*7.5 Conclusion*134

A Appendix136
*A.1 Links to Systems*136
*A.2 Study Data and Materials*136

References137

LIST OF TABLES

4.1 A timeline of the SME's existing procedure to complete one cycle of their assembly process. 65
4.2 A timeline of process 1 including the steps and timing for a cobot to assist one or two workers in applying silicone. 68
4.3 A timeline of process 2 including the steps and timing for a cobot to assist one or two workers in gathering components and applying silicone. 69
4.4 This table shows several metrics comparing both proposed processes and their direct differences. 71

5.1 Robot error rate for physical interaction tasks, defined as a function of sensing capability, and both location structure and variability. 93
5.2 Robot error rate for visual inspection tasks, defined as a function of sensing capability and skill rating. 94

LIST OF FIGURES

3.1 The original design of the *CoFrame* interface, including components for the Program Editor (G), which includes the Block Drawer (D) and Program Canvas (E), Simulator (B), Expert Frames (A), Contextual Information (C). The Program Canvas (E) contains the program (F) along with implemented skills. Figure from Schoen et al. (2022); Schoen (2023). 22

3.2 The updated design of the *CoFrame* system, extending the original through the addition of timeline (E) to visualize detailed program execution timing, the Task Goals tile (F) presenting high-level guided steps to completing the task, and the documentation window (G) providing supplemental information about program blocks and issues. 24

3.3 Interaction documentation window built in to the *CoFrame* programming environment. Operators can select issues from the Expert Feedback tile to receive information about the problem and how to solve it (Left), or understand block functionality (Top Right) and parameters (Bottom Right). 26

3.4 Three case studies showing the process of evaluating feedback from the system and informing adjustments to the operator's program. The gradient background of the figure denotes the switching between *Expert Frames* by the operator, from *Safety Concerns* (pink), *Program Quality* (blue), *Robot Performance* (orange), and *Business Objectives* (green). In Case Study 1, the operator begins by investigating the task goals and previews the goal (A). The operator then creates a trajectory and addresses a missing trajectory block (B), followed by filling in its parameters (C). The operator then addresses reachability concerns (D) and finishes by addressing issues with robot collision (E). In Case Study 2, the operator begins by addressing joint speed issues, visualizing the speed, and reading documentation to address the issue (F). They transition to solving pinch point issues (G) and finish by addressing issues with the robot's space usage (H). In Case Study 3, the operator begins by investigating the next task and solves issues with machine logic (I). The operator addresses problems with unsafe thing movement (J), and finishes by viewing their completed task (K). 28

4.1 A depiction of the four-phase cobot integration approach proposed within this paper: planning for integration, analyzing workflows, developing simulations, and presenting to the manufacturer. 38

4.2 User Experience Questionnaire results including the average scores of novices and experts and confidence intervals. These results show that both novice and expert participants regard CoFrame favorably with positive average scores across all six UEQ subscales. 46

4.3 An overview of the four-phase approach including the individuals, foci, and milestones involved at each phase. 58

4.4 A subset of the steps involved in the manufacturer's assembly process. A. applying silicone to gaskets. B. applying silicone to pans and beginning the assembly process. C. continuing the assembly process by attaching gaskets. 65
4.5 The program and simulated environment created for process 2. The environment captures the robot, a workspace for the operator, a workstation, a location to switch end effectors, and a source for component parts. 67
4.6 This plot showcases the costs of our processes and their cycle times to assemble two units when one or two workers are assisted by the cobot. 73

5.1 A breakdown of *Allocobot*'s job representation. Each card represents core components defining a job, and includes Agents to be utilized in the job, a set of Tasks or steps to complete the job, Targets to be manipulated throughout the job, Points of Interest (POIs) that represent the job environment, and Primitives that define the actions for each Task. Figure from Schoen (2023). . 86
5.2 Example environment for the kitting task case study. It includes two bins to pick parts from to assemble at the kitting station. . 99
5.3 The user begins by defining all POIs in the environment (A), then specifies the desired *Agents* for the collaboration (B). Then the user defines all *Targets* (C), and the tasks that use them to complete the job (D). Finally, the user simulates the interaction to produce the best collaboration. 100

6.1 Scenario environments for the evaluation of Allocbot. (A) depicts an automotive assembly job where agents will need to move around the environment to grab parts and tools to work on the engine. (B) depicts a small parts assembly job where agents will prepare and assemble parts. For both (A) and (B) areas marked in orange represent the POIs where agents will stand, while the areas marked in red represent hand locations where they will perform work. 110
6.2 The change in work allocation and primitive assignment due to the change in alpha weighting for the Automotive Manufacturing Scenario's four-step process. For each α value, we see the resulting allocation of robot (R), human (H), or human-robot (HR) to the task. Additionally, each allocation specifies which primitives are assigned to each agent, including force (F), position (P), or hold (H). Finally, on the right is the total cycle time for the collaboration. 114
6.3 The ergonomic impact on the human agent measured by Allocobot for each produced allocation in the Automotive Manufacturing scenario. 116
6.4 The total cost and time for task completion for each produced allocation in the Automotive Manufacturing scenario. 117
6.5 The total cost of the task, minus the cost of the task setup, and time for task completion for each produced allocation in the Automotive Manufacturing scenario. 118

6.6 The change in work allocation and primitive assignment due to the change in alpha weighting for the Small Parts Assembly's four-step process. For each α value, we see the resulting allocation of robot (R), human (H), or human-robot (HR) to the task. Additionally, each allocation specifies which primitives are assigned to each agent, including force (F), position (P), or hold (H). The lack of assigned primitives indicates that the task has not been performed after being allocated to the agent. Finally, on the right is the total cycle time for the collaboration, where ∞ indicates the task is not completed. 120
6.7 The ergonomic impact on the human agent measured by Allocobot for each produced allocation in the Small Parts Assembly scenario. 121
6.8 The total cost of the task, minus setup costs, and time for task completion for each produced allocation in the Small Parts Assembly scenario. 122

ABSTRACT

With the introduction of Industry 5.0, there is a growing focus on human-robot collaboration and the empowerment of human workers through the use of robotic technologies. Collaborative robots, or cobots, are well suited for filling the needs of industry. Cobots have a prioritization on safety and collaboration, giving them the unique ability to work in close proximity with people. This has the potential impact of increasing task productivity and efficiency while reducing ergonomic strain on human workers, as cobots can collaborate on tasks as teammates and support their human collaborators.

However, effectively deploying and using cobots requires multidisciplinary knowledge spanning fields such as human factors and ergonomics, economics, and human-robot interaction. This knowledge barrier represents a growing challenge in industry, as workers lack the skills necessary to effectively leverage and realize the potential of cobots within their applications, resulting in cobots often being used non-collaboratively as a form of cheap automation. This presents several research opportunities for the creation of new cobot systems that support users in the creation of cobot interactions.

The goal of this dissertation is to explore the use of abstraction and scaffolding supports within cobot systems to assist users in building human-robot collaborations. Specifically, this research (1) presents updates to the design of systems for planning and programming collaborative tasks, and (2) evaluates each system to understand how it can support user creation of cobot interactions. First, I present the CoFrame cobot programming system, a tool built on prior work, and illustrate how it supports user creation and understanding of cobot programs. Then, I present the evaluation of the system with domain experts, novices, and a real-world deployment to understand in which ways CoFrame does and does not successfully sup-

port users. I then describe the Allocobot system for allocating work and planning collaborative interactions, describing how it encodes multiple models of domain knowledge within its representation. Finally, I evaluate the Allocobot system in two real-world scenarios to understand how it produces and optimizes viable interaction plans.

1.1 Motivation

Collaborative robots (cobots) are robots marketed for their focus on safety, collaborative capabilities, and easy-to-use programming, and have grown in adoption by the manufacturing industry (Grand View Research, 2023). The marketing for cobots (Robots, 2023; fra) promises deployment and usage reconfigurability through their user-friendly programming systems along with increased safety for human work environments. These robots are sold on the promise of ease of use and improved safety, but not at a premium cost.

This marketing of safety and ease of use aligns well with the industrial needs outlined in the concepts of Industry 4.0 and 5.0. One of the key concepts for Industry 4.0 in the manufacturing domain is the flexibility and reconfigurability of manufacturing processes (Lasi et al., 2014; Qin et al., 2016). Manufacturers require adjustable and reconfigurable work cells to accommodate changes to and personalization of products while meeting manufacturing efficiency needs. Traditionally, these adjustments would be accomplished through automation and worker replacement, but Industry 5.0 attempts to address this by focusing on human-robot collaboration and enhancing the capability of the human workforce (Leng et al., 2022; Xu et al., 2021; Akundi et al., 2022).

In addition to cobot capability aligning with these concepts, research also highlights additional positive benefits that these collaborative interactions can provide, including reducing strenuous activity for human workflows, reducing overall cycle times, and reducing costs (Sanneman et al., 2021; Pauliková et al., 2021). Human-robot collaboration can also combine the adaptiveness of humans with the accuracy and precision of cobots, creating workflows that leverage the benefits of all parties in-

volved. This is important to help alleviate worker fears of automation and replacement that exist within the workforce (Golin and Rauh, 2022; Spencer, 2018).

Given the potential of cobots for fulfilling these needs of collaboration and safety, cobots have demonstrated a high potential for application in multiple domains and regions (Jacobs, 2024). As a result of this application potential, cobots have grown in market share with projections increasing through 2030 (Grand View Research, 2023). However, we currently do not see cobots being used in collaborative applications (Michaelis et al., 2020). While adoption is improving, additional requirements and considerations are needed for effective collaboration when compared to traditional robotics and automation approaches. Due to the complexity of these additional requirements, many users struggle to understand new and unfamiliar interaction paradigms of collaborative robots, as much of the workforce has not been adequately trained on or adapted to these needs. Collaboration requires consideration of the individuals involved in the task and the nature of the work being performed. This is further complicated when incorporating cobots due to the difficulty of accounting for changing environments and the non-deterministic actions of human workers. This complexity, coupled with the unpredictability of humans, puts cobots out of reach for many manufacturers.

The programming systems that accompany cobots, such as teach pendants, are meant to assist users in building programs to achieve collaborative interactions, but do not provide adequate support for users (Dong et al., 2021; Zieliński et al., 2021; Byun and Dong, 2023). As a stop-gap, additional formal training is often required for users to supplement system usage and provide a better understanding of how to operate cobots safely while utilizing the collaborative potential of cobots. This lack of user support and difficulty in understanding cobot interactions have been some of the factors that have led to a growing "skills gap" in the industry

(Shipps and Howard, 2013; Holm et al., 2021; Leitão et al., 2020). Additionally, this skills gap has resulted in cobots being mainly utilized as a more cost-effective form of automation (Michaelis et al., 2020; Wallace, 2021), meaning they are used similarly to traditional robots and have their collaborative capabilities underutilized. Due to these factors, manufacturers struggle to specify how they want to interact with and effectively utilize cobot systems, resulting in their use as tools for automation or manufacturers relying on third-party integrators with cobot expertise for work-cell development and implementation.

While the adoption of cobot hardware is growing, it is important to create software solutions that support users, making the collaborative benefits accessible and realizable. Therefore, new systems and supports are needed to help manufacturers integrate and utilize cobots effectively. This is challenging due to the various processes involved with both planning and programming cobot use within work cells, from the planning stages where companies figure out how to incorporate cobots in the manufacturing process and how it will collaborate with a person on the task, to programming it to be able to work effectively with an operator. As these stages could involve different groups of individuals, there are many areas where specialized knowledge of cobots is required, making it difficult to realize their full potential. Thus, to make cobots more accessible and usable, new tools are needed that can better complement users' existing knowledge and support their decision-making. The work in this dissertation presents the design and evaluation of several systems that address the growing need for human-robot collaboration by fulfilling the needs of Industry 5.0 while bridging the skills gap within the existing workforce.

1.2 Thesis Statement

This dissertation explores how we can design systems that support users in creating and understanding effective human-robot collaborations. Towards this goal, my thesis is as follows: **Cobot interfaces that integrate abstraction and scaffolding can facilitate planning and programming human-robot collaborations.**

For this dissertation, we adopt the definition of abstraction by Wing (2008) as "deciding what details we need to highlight and what details we can ignore," meaning that it supports users in focusing conceptually on what something does. Additionally, we adopt the definition of scaffolding by Guzdial (1994) as "support which enables a student to achieve a goal or action that would not be possible without that support," meaning that it serves as the pedagogical strategy for how users engage with or learn a given concept. Put another way, abstraction allows users to think about the problem at a conceptual level, and scaffolding is how those concepts are learned and applied. While these definitions are frequently utilized together, they refer to different ideas. For example, in cobot programming, abstraction might involve focusing on high-level goal-based decisions, such as "move robot" or "grasp object," instead of low-level controls for implementing robot behaviors and motions. At the same time, scaffolding might provide expert knowledge that explains the behaviors and how their usage impacts safe human-robot interactions. In this example, abstraction allows users to think about the human-robot collaboration at a goal-driven level, while scaffolding focuses on how users utilize and interact with these representations.

The system described in Chapter 3 describes the design and implementation of the CoFrame programming system. It highlights design decisions for supporting end-user robot programming by abstracting cobot programming to the conceptual level and by providing scaffolding guidance on program development that incorporates the expertise needed to build

effective collaborative interactions. The chapter discusses these design decisions and the ways they provide support to end-users, highlighting their interaction through illustrative case studies.

Chapter 4 presents the evaluation of the CoFrame system with domain experts and novices, along with the results of a real-world deployment. It provides qualitative insights regarding user perceptions of the system and their use of the abstraction and scaffolding supports. Additionally, it provides insight into the requirements for real-world deployments, identifying new areas of support required and presenting the key outcomes and areas of support needed for small-to-medium enterprise (SME) partners to understand cobot interaction planning and programming.

Chapter 5 presents the design of the Allocobot system, a system built to facilitate the planning of human-robot collaborative interactions by addressing the areas of support described in Chapter 4. It highlights the design decisions for supporting end-user cobot interaction planning by leveraging user domain expertise and abstracting knowledge from the domains of robotics, ergonomics, human factors, and economics to autonomously allocate work and plan collaborative tasks.

Finally, Chapter 6 presents several scenarios to evaluate the use of the Allocobot system on real-world tasks, illustrating how it focuses on existing user knowledge, abstracts the required knowledge to plan and operationalize human-robot collaborations.

1.3 Methodology

The work in this dissertation uses several methodologies to (A) *understand* the needs of end-users, (B) *design and build* the systems that provide abstraction and scaffolding to create human-robot collaborations, and (C) *evaluate* the use and impact of the abstractions and scaffolding in the systems. While each step represents a unique portion of the process, these

are sometimes overlapping or require iteration.

The first step of my approach, understanding the needs of end-users, involved both an understanding of the literature and working with end-users to identify problems and research questions situated in real-world contexts. For example, Chapter 4 involved engaging end-users to understand and identify gaps in user support for creating cobot programs. In addition, this phase involved an understanding of the literature to identify areas for user support, such as illustrated in Chapter 3.

The second step of my approach is the design and development of systems to address the needs identified through my understanding. This step involved designing multiple abstraction and scaffolding supports that address the needs of users to be combined within systems to enable users to create cobot interactions. For example, both Chapters 3 and 5 highlight the design and implementation of both the CoFrame and Allocobot systems that each encapsulate domain knowledge to support users in creating human-robot collaborations.

The final step of my approach is the evaluation of the created systems. To understand how these systems impact the cobot programming and planning process, we employed several techniques, both quantitative and qualitative, to understand how they provide support to users. For example, Chapter 4 incorporates user studies as well as a real-world deployment to understand the use and perception of the CoFrame system holistically. Additionally, Chapter 6 highlights the use and support of the Allocobot system through its application on real-world scenarios.

1.4 Contributions

This dissertation makes several technical, design, empirical, and theoretical contributions. Specifically, this dissertation makes the following contributions:

1. *Survey* – A review of cobot usage within the manufacturing industry and the state of human-robot collaborative systems (Chapter 2)
2. *Technical* – An update to the design and implementation of the CoFrame programming system (Chapter 3, Section 3.3)
3. *Empirical* – An empirical evaluation of the CoFrame programming system with experts, novices, and in a real-world deployment (Chapter 4, Section 4.5)
4. *Theoretical* – The creation of a framework for supporting end-user understanding through the identification and creation of collaborative interaction plans (Chapter 4, Section 4.7)
5. *Theoretical* – Design implications for human-robot collaborative programming systems (Chapter 4, Section 4.9)
6. *Technical* – An update to the algorithmic design and creation of the Allocobot collaborative planning system (Chapter 5, Section 5.3)
7. *Empirical* – The presentation and discussion of several real-world scenarios for the evaluation of the Allocobot collaborative planning system (Chapter 6, Section 6.4)

1.5 Dissertation Overview

The remainder of the dissertation is organized into six chapters. Chapter 2 provides the relevant background related to the use of cobots in industry and systems for human-robot collaboration. Chapter 3 presents an iteration on the design and development of the *CoFrame* system. Chapter 4 presents a user evaluation to understand the perception of the designed programming supports, the utility of such a tool in industry, and the identification of additional supports for use in real-world contexts. Chapter 5

presents the design and development of *Allocobot*, an automated system that attempts to address the problem of collaboration planning identified in Chapter 4 to enable users to create collaborative interactions. Chapter 6 presents an evaluation of the *Allocobot* system using several real-world scenarios, illustrating how the system supports user creation and understanding of collaborative interactions. Finally, Chapter 7 presents a general discussion about the work presented in Chapters 3-6, including the contributions, significance, and limitations of the work, as well as a discussion of future research directions.

2 BACKGROUND

This dissertation argues that the use of cobots for collaborative interactions is currently difficult for users to achieve and that this can be made more accessible through the creation of new systems that provide abstraction and scaffolding supports for end-users. To contextualize the contributions of this work, this chapter provides relevant material for understanding and motivating the needs and desires of industry, why cobots are a well-suited technology to address those needs, the current difficulties in using cobots, and how to support end-user programming.

This chapter provides a general context for understanding and motivating the work of this dissertation. Further background and related works are included in Chapters 3-6, specific to the design and usage of the discussed systems.

2.1 Understanding the Demands of Industry

Industry 4.0 is well underway within the manufacturing domain (Bag et al., 2021; Jamwal et al., 2021). It makes several promises about the change in automation, including increased efficiency in manufacturing and resource management combined with the flexibility for reconfiguring processes to account for a wider variety of products and customization of them (Lasi et al., 2014; Qin et al., 2016). This is highly desirable for manufacturers, as the increased flexibility means that manufacturing processes and lines can change and adjust as needed to varying product designs and requirements. However, this necessitates that any automation and collaboration technologies utilized are also flexible and reconfigurable.

These requirements are compounded with the next wave of industrialization, Industry 5.0 (Leng et al., 2022; Xu et al., 2021; Akundi et al., 2022). Industry 5.0 focuses on the human element of the manufacturing

pipeline, with an emphasis on human-robot collaboration, the empowerment of workers, and the versatility of interactions (Nahavandi, 2019). This promises to merge the precision and accuracy benefits from automation with the versatility and adaptability of humans. Additionally, this focus promises benefits to workers, as work can be allocated based on the strengths of each agent, thus potentially reducing worker injury and harm (Vysocky and Novak, 2016; Pearce et al., 2018).

Cobots have been heavily marketed to the manufacturing industry over the last few decades (Simões et al., 2020), but have also found adoption in logistics (Lappalainen, 2019), and medicine (Ernst and Jonasson, 2020). Marketing material, such as for the UR5 robot (Robots, 2023), includes mentions of ease-of-programming, a focus on safety, and an encouragement for possible human-robot collaborations. This has resulted in a growing adoption rate within the manufacturing domain (Grand View Research, 2023). Cobots have been seen as one way of fulfilling this flexible and collaborative need (Knudsen and Kaivo-Oja, 2020), as they can be interconnected with other systems within the factory, allowing them to be more easily reconfigured and adjusted as needed to suit these changing processes (Tamas and Murar, 2019). Additionally, their focus on safety features allows them the capability to work alongside people, with prior work exploring how work can be structured to enable humans and robots to work together (Shi et al., 2012; Pearce et al., 2018), and how cobots can be integrated into production lines (Wojtynek et al., 2019; Horst et al., 2021). Depending on implementation, cobots can be used to collaborate on tasks with humans, acting as a teammate and enhancing human capability (Christiernin, 2017; Michalos et al., 2015).

Other work demonstrates the additional benefits that are provided when using cobots, such as the improvement of physical ergonomic working conditions for operators (Peshkin and Colgate, 1999; Cardoso et al., 2021; Vysocky and Novak, 2016) and the reduction of cycle time (Enrique

et al., 2021). The benefits of integrating cobots, alongside their usability, are some of the main factors for adoption (Simões et al., 2019, 2020); and have resulted in cobots having a demonstrated history of being successfully integrated into various manufacturing processes, providing companies various productivity benefits (Gil-Vilda et al., 2017).

2.2 Difficulty of Using Cobots To Fit the Needs of Industry

Recent work has found that there are still many scenarios where cobots are being utilized as a cheaper alternative to traditional manufacturing robots, resulting in these companies not using their collaborative capabilities due to the additional difficulties cobots bring (Michaelis et al., 2020; Guertler et al., 2023; Wallace, 2021). Cobots are fundamentally different from traditional robotics used in automation. Collaboration introduces many new ideas and concepts that require new ways of thinking. When planning interactions, while there are still considerations for system performance and throughput, there are additional considerations for how robots should interact with individuals (Galin and Meshcheryakov, 2020; Khalid et al., 2017; Christiernin, 2017; Shi et al., 2012). This includes social aspects such as physical characteristics and team dynamics (Sauppé and Mutlu, 2015; Goetz et al., 2003), new capabilities such as being able to adapt to changing environments and conditions (El Zaatari et al., 2019), and the management of trust in robots due to fear of job loss (Kopp et al., 2021). Additionally, organizations have released specific standards that outline how to ensure safe operation and collaboration across a wide range of cobot configurations and work cell designs (ANSI/RIA R15.06-2012, 2012; ANSI/RIA R15.08-1-2020, 2020; ISO/TS 15066:2016, 2016). These regulations highlight critical and tangible design aspects that must be met for collaborations, including the operation speed of the cobot, the max-

imum force it can exert under several scenarios, and additional sensing and capabilities required for proximal user interaction.

This is a lot of knowledge required for individuals building collaborative interactions. Even after knowing this information, there are additional difficulties as users must find appropriate tasks to fit cobot capabilities and determine how to utilize them effectively (Kadir et al., 2018). After finding an appropriate task, increasing task efficiency can require making adjustments to the robot's program, which can be difficult as the responsibility of reprogramming the interaction has shifted from engineers to operators who are not necessarily robotics experts (Massa et al., 2015). Further complications arise as operators find the task of reprogramming to be intimidating, due to feeling like they are not trained to be able to perform such tasks and that it requires years of training to be able to do (Giannopoulou et al., 2021).

Other challenges beyond programming hinder the usage of cobots, such as how to have the cobot operate safely within the environment (Kildal et al., 2018; Malm et al., 2019), and how to balance control between the users and robot to reduce overall stress (Pollak et al., 2020). These concerns are less prevalent in students but are still a major concern amongst current operators (Kildal et al., 2018). Thus, planning for cobot integration can be difficult and includes thinking about factors such as people being in unexpected areas, whether the cobot is holding an object, and how fast it is moving (Bi et al., 2021). Improper movements while holding objects can create uncollaborative environments, as collaborative environments as dictated in part by the task and are not inherent in the application of cobots (Guertler et al., 2023). Therefore, more consideration is required early on when determining how to integrate cobots into the environment and assign them tasks. This is partially mitigated through giving the operator control over the robot and the interaction, as they can adjust behaviors and prevent potential issues (George et al., 2023). However, determining

what controls to present to the operator and when to do so requires extra consideration before a user ever begins to physically interact with a cobot.

2.3 Addressing Cobot Difficulty

These challenges have demonstrated that existing paradigms are not sufficient and that new, simpler systems and paradigms are needed that allow for the various end users to be able to more effectively use the cobot (Hentout et al., 2019; Weiss et al., 2021; Djuric et al., 2016; Emeric et al., 2020). While existing systems, such as the teach pendant for the UR cobots, allow users to build programs on the fly, prior work has shown that there are multiple ways in which they are insufficient for supporting end-user programming (Dong et al., 2021; Zieliński et al., 2021; Byun and Dong, 2023). To address this issue, several programming interfaces and systems have been developed in prior work (Schoen et al., 2022; Steinmetz et al., 2018; Schou et al., 2013; Pieskä et al., 2018; Senft et al., 2021a) that try to bridge the gap in understanding for cobots by abstracting different characteristics of the programming process or providing the user with contextual information. Several different methods have been explored for supporting the programming process, such as the use of block-based programming to make the programming process more accessible, understandable, and logic focused (George et al., 2023), as well as through the use of flow charts to represent moving through the steps within a process (Dmytriyev et al., 2022). In regards to providing additional information whilst programming, recent work has explored the concept of a digital twin, a paradigm where the real-world environment is simultaneously modeled within a digital environment to understand and measure the impact on the worker, predict robot actions, and understand the state of environment (Rivera-Pinto et al., 2023; Malik and Brem, 2021). Other systems have looked at using new modalities for cobot programming,

such as using augmented reality (AR) to improve collaboration between users and allow for multiple types of users to be able to program the robot (Kapinus et al., 2023). This variety of systems and paradigms has begun to be explored to try and address the need for simplified programming of cobots.

This has resulted in the identification of a number of areas for improving the use of cobots, such as system usability, trust, and safety (Chowdhury et al., 2020), with suggestions for new systems focusing on the design of more intuitive interfaces (El Zaatari et al., 2019). Prior work has also laid out a complete set of guidelines for how these new systems need to support users and the information they require to be better informed during the interaction (Frijns and Schmidbauer, 2021), but emphasizes that the interfaces should generally include information about context and human awareness to make the interaction feel more safe (Kildal et al., 2018) and alleviate the mental workload of the operator (Faccio et al., 2023).

While it is important to develop these new systems and interfaces, recent work has emphasized the importance of exploring the human side of this problem (Salvatore et al., 2021), with one avenue being the inclusion of end users in the design process using methods such as participatory design (Kaasinen et al., 2020; Bounouar et al., 2022). Other studies have tried engaging makerspace communities as a way of having the community get engaged with the creation of these new tools and interfaces (Ionescu and Schlund, 2019; Ionescu, 2020). Through this incorporation of end users in the early phases of the system design process and making the developed tools more accessible to different communities, prior work seeks to give them a stake in the development of the tools; thereby creating systems that are both more usable and better support the user in their tasks.

2.4 Supporting End-User Programming

One way to help users in programming is by providing support to simplify or ease the programming process. One common practice has been through the introduction of block-based programming systems (Schoen and Mutlu, 2024; Fraser, 2015; Resnick et al., 2009) that abstract low-level code implementation. Block-based programming has been shown to help students with learning to program (Mladenović et al., 2018), especially when compared to text-based programming (Weintrop and Wilensky, 2017, 2015). Due to the benefits and accessibility that these block-based systems provide, several robot programming tools have begun to explore the benefits of block-based robot programming (Mayr-Dorn et al., 2021; Weintrop et al., 2017; Bachiller-Burgos et al., 2020).

Another form of support provided to users is the encapsulation of expert robotics knowledge within the system (Schoen et al., 2020; Sanders et al., 2018). These systems can support user understanding of how to make program adjustments that accommodate the robot's capabilities or outright correct problems. This form of support attempts to have the system supplement user understanding through abstraction (Janjanam et al., 2021; Buchanan and Smith, 1988), thereby reducing the barrier of having users fully understand robots and their capabilities before using them. This aligns with other work in understanding the effect of abstraction in program learning, demonstrating that having multiple levels of abstraction helps to facilitate increased learning (Waite et al., 2018; Devathasan et al., 2022).

2.5 Chapter Summary

As trends in the manufacturing industry seek to increase flexibility and throughput while simultaneously empowering and supporting their human workforce, cobots are particularly well suited to address this need.

Cobots have demonstrated the potential to assist workers while providing safe environments for collaboration. However, current implementation techniques for cobots are difficult due to the required deep understanding of collaborative principles and the inaccessibility of support for programming approaches. While new systems are improving, existing systems do not adequately support users, resulting in cobot systems being used for cheap automation instead of collaboration. This highlights a need for new systems that better support users in the creation of effective cobot programs, allowing them to more fully meet the needs and promises of Industries 4.0 and 5.0.

This chapter provided background relevant to contextualizing and understanding the work presented in this dissertation. Specifically, this chapter highlights the current state of cobot usage in the manufacturing industry and the difficulties in current implementations, while highlighting ways of approaching and addressing this shortcoming. The work presented in the following chapters leans on the understanding presented in this chapter and generally builds on the work in the area of cobot programming and planning.

3 CoFrame: Designing Programming Systems for Human-Robot Collaborations

This chapter describes the design and implementation of the *CoFrame* programming system. It addresses the challenge of supporting the end-user robot programming process by abstracting core robotics knowledge and scaffolding user interaction to build effective cobot programs. In this chapter, we discuss the motivation of the work, review relevant prior work to understand the design and application of the system, provide details about our system, and reflect on the design process through an illustrative workflow. This chapter includes work from a manuscript in progress (White et al., 2025b), building on our previously published work in Schoen et al. (2022); Schoen (2023).

3.1 Motivation

Industry 5.0 concepts focus on human-robot collaboration within manufacturing (Nahavandi, 2019). This focus attempts to enable human workers, merging the precision and accuracy of automation with the versatility and adaptability of humans. This focus on the empowerment of the workforce comes at a time when the industry faces labor shortages (Autor, 2021), which further necessitates the incorporation of new technologies. Collaborative robots (cobots) are one such technology that can address these needs by combining the skill sets of both humans and robots for collaboration (Pearce et al., 2018), and have seen an increase in usage within industry (Miller, 2021).

Cobots are uniquely equipped to address the needs of manufacturers, with a high potential for impact in the manufacturing industry (Liu et al., 2022a; Knudsen and Kaivo-Oja, 2020). Their focus on safety features allows them to work effectively alongside people, with the potential to

interact as a teammate and enhance human capability (Christiernin, 2017; Michalos et al., 2015). This ability to collaborate and interact with human workers, while performing precise and repetitive actions, allows for cobot integration in many areas such as assembly, machine tending, or kitting. These safety features, coupled with the collaborative capability of cobots, allow for them to assist users in strenuous tasks, improving physical ergonomic working conditions for operators (Peshkin and Colgate, 1999; Cardoso et al., 2021; Vysocky and Novak, 2016) and reducing overall task cycle time (Enrique et al., 2021).

However, programming cobots is non-trivial, meaning that these benefits are difficult for users to achieve. This has resulted in a growing "skills gap" (Michaelis et al., 2020; Wingard and Farrugia, 2021; Holm et al., 2021), in which workers in industry do not have the capability to successfully integrate cobots into collaborative tasks. The existing workforce is trained on traditional automation skills and is not trained to develop cobot programs, resulting in feelings of intimidation when attempting to do so (Giannopoulou et al., 2021). Cobot interactions require specialized knowledge, planning for non-deterministic human behavior, and the consideration of changing work cell environmental factors. Additionally, it requires operators to consider and account for many critical safety and performance factors that are not always obvious.

One method to address this gap is the development of new systems to support users in the creation of cobot programs. In this chapter, we present an update on the design and implementation of the *CoFrame* digital programming environment and discuss how we leverage design to build a programming environment that supports end users in learning and building collaborative programs.

The contributions described in this chapter include[1]:

[1]The research in this chapter is derived from a manuscript in progress meant as an update to the published work by Dr. Andrew Schoen, myself, Curt Henrichs, Dr. Amanda Siebert-Evenstone, Dr. David Shaffer, and Dr. Bilge Mutlu.

- An update to the design and implementation of the *CoFrame* human-robot collaboration programming environment (see Appendix A for a link to the system);
- The presentation and discussion of an updated set of case studies demonstrating the system's support for end-users building human-robot collaborative programs.

3.2 Related Work

Cobots have been growing in adoption by the manufacturing industry (Simões et al., 2020; Grand View Research, 2023), given their safety features enabling the safe operation and the potential for collaboration to work with people (Shi et al., 2012; Pearce et al., 2018; Christiernin, 2017; Michalos et al., 2015). This collaborative potential has allowed cobots to take on more strenuous physical labor in collaborative tasks, allowing for the improvement of physical ergonomic working conditions for operators (Peshkin and Colgate, 1999; Cardoso et al., 2021; Vysocky and Novak, 2016) and the reduction of cycle time (Enrique et al., 2021). However, despite these potential benefits, there is a growing skills gap in industry resulting from the lack of skills necessary to effectively use cobot technology within the existing workforce (Ras et al., 2017; Michaelis et al., 2020; Wingard and Farrugia, 2021; Shmatko and Volkova, 2020; Giffi et al., 2018).

While educational and training programs have been proposed as a means of addressing this gap (Chrisinger, 2019), prior work has found a lack of emphasis on critical technical and non-technical skills in these educational programs (Andrew et al., 2020). In response, cobot manufacturers have provided training programs that target the specific skills they deem necessary to utilize their products, and are being used in vocational training (Słowikowski et al., 2018). However, recent work shows that

these programs are not sufficient, as industry primarily uses cobots for automation due to the lack of necessary skills (Michaelis et al., 2020).

This has led to several attempts at teaching robot programming concepts to the potential future workforce, through high school (Dagdilelis et al., 2005) and college students (Ziaeefard et al., 2017), as well as attempting to increase cobot accessibility within the community (Ionescu and Schlund, 2019; Ionescu, 2020). Additional work has explored contextualizing cobot skills through hands-on, real-world learning environments, such as the "Teaching Factory" that offers a factory-like classroom environment (Mavrikios et al., 2013; Chryssolouris et al., 2016), or virtual and augmented reality platforms that enable users to perform work tasks and processes (Matsas and Vosniakos, 2017).

This has left the programming of cobots for new tasks to be one area where the skills gap is significant (El Zaatari et al., 2019). While prior work has explored several robot programming systems for skill demonstrations (Steinmetz et al., 2018, 2019), human-robot task allocations (Schoen et al., 2020), and AR-based interfaces (Perzylo et al., 2016; Gao and Huang, 2019; Senft et al., 2021b,a), they do not adequately support user understanding for the programming and debugging of cobot applications.

One technique for supporting the cobot programming process is through the use of visual programming languages (Alexandrova et al., 2015; Huang and Cakmak, 2017), including block-based programming. Block-based programming has been shown to help students with learning to program (Mladenović et al., 2018), especially when compared to text-based programming (Weintrop and Wilensky, 2017, 2015). This has been explored within the context of programming of industrial robots (Weintrop et al., 2017) and has been shown to be more effective and preferable to the existing cobot programming interfaces (Weintrop et al., 2018). Additional systems have explored the encapsulation of expert robotics knowledge (Schoen et al., 2020; Sanders et al., 2018), an abstraction technique shown

to support user understanding (Janjanam et al., 2021; Buchanan and Smith, 1988).

Our work builds on this body of knowledge, attempting to design a cobot programming system that addresses the skills gap through a combination of introducing and supporting cobot concepts, situating learning within hands-on real-world problems, and abstracting cobot knowledge that can support users.

3.3 System Design & Implementation

In this section, we summarize the initial implementation of the *CoFrame* system and then highlight the changes and additions made to the system.

Original Implementation

CoFrame is a programming system designed to support users in the creation of collaborative robot programs. Figure 3.1 showcases the original implementation of the system. The interface contains a four-tile layout with several key features for users. This includes a built-in simulator (B), a programming environment (G), an expert feedback panel (A), and an area to view and interact with contextual information (C).

CoFrame includes a block-based visual programming environment that is inspired by Blockly (Fraser, 2015) and Scratch (Resnick et al., 2009). It aims to simplify the programming paradigm for novice users, leveraging a drag-and-drop-based design, allowing users to assemble programs using a set of predefined programming blocks. These blocks are color-coded according to specific categories of actions and environmental objects, and organized within a program drawer according to these categories. These categories are:

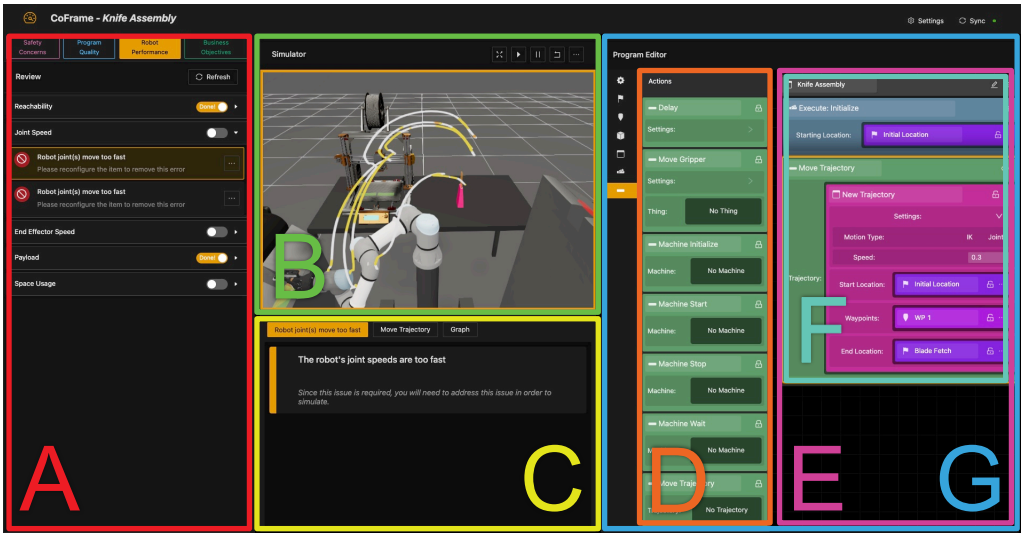

Figure 3.1: The original design of the *CoFrame* interface, including components for the Program Editor (G), which includes the Block Drawer (D) and Program Canvas (E), Simulator (B), Expert Frames (A), Contextual Information (C). The Program Canvas (E) contains the program (F) along with implemented skills. Figure from Schoen et al. (2022); Schoen (2023).

1. Things and Tools, which represent the physically interactable objects in the environment.

2. Machines and Processes, which represent the manufacturing machines that produce and consume parts, as well as their associated processes.

3. Locations and Waypoints, which are 3D points of interest in the environment.

4. Actions, such as “Move Gripper”, “Move Trajectory”, and “Machine Start”, which affect robot or machine states.

5. Containers, such as “Hierarchical” and “Trajectory,” which cluster and organize related actions for modularity and clarity.

6. Skills, analogous to functions in traditional programming, which define reusable sequences of parameterized actions.

CoFrame also includes a feedback panel, or Expert Frames, which helps users to identify and visualize issues during the program creation process through four frames: Safety Concerns, Program Quality, Robot Performance, and Business Objectives. Each of these frames addresses unique issues pertaining to cobot programming. Under Safety Concerns, the system detects rapid motions in dangerous directions (End Effector Pose), checks for the grasping of unsafe objects (Thing Movement), highlights unsafe areas to place hands near the robot (Pinch Points), identifies collisions between the robot and the environment or the robot itself (Collisions), and monitors for the robot approaching or entering the human work zone (Occupancy). For Program Quality, the system identifies structural and logic issues in the user-generated program, including missing blocks, missing parameters, issues in the order of operations for using machines (Machine Logic), unused skills and features, and empty program blocks. Within Robot Performance, the system flags reachability issues for locations or waypoints (Reachability), trajectories that exceed safe speed thresholds in joint space (Joint Speed) or end effector space (End Effector Speed), situations where payload capacity is exceeded during grasps or trajectories (Payload), and trajectories resulting in excessive workspace usage (Space Usage). Finally, the Business Objectives category includes metrics that are graphed as users iterate on the program: changes in cycle time, idle time, and expected return on investment.

Each issue is presented as either a warning or an error. Errors are presented as must-fix problems for the user, while warnings can be manually acknowledged and dismissed or addressed. As users interact with the frames and issues, feedback is presented to the user through the Contextual Information tile. This tile includes definitions of key terms, usage suggestions, and graphical breakdowns of the selected issue. For example,

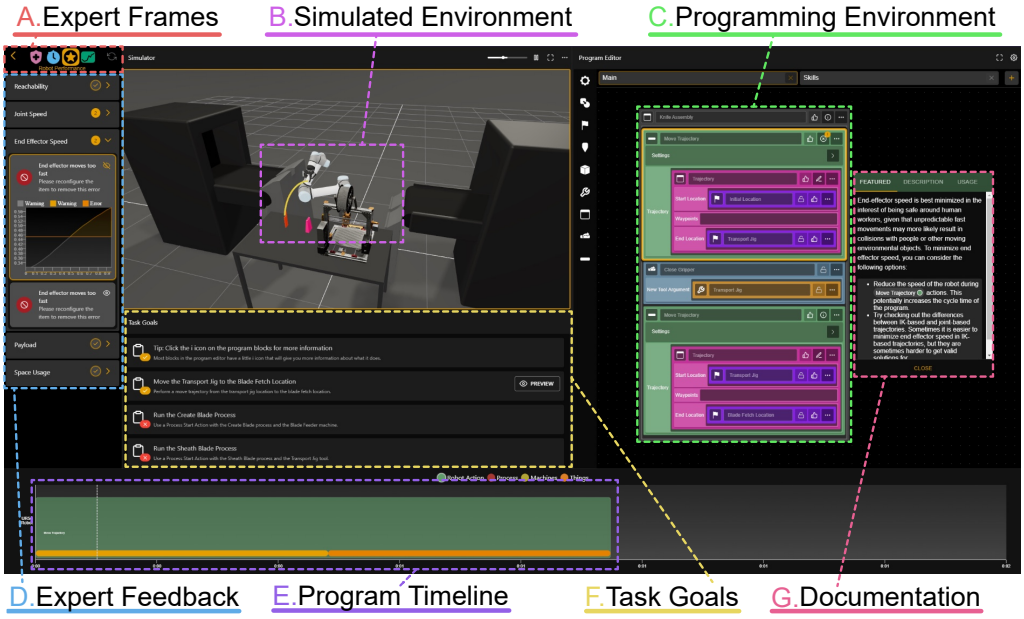

Figure 3.2: The updated design of the *CoFrame* system, extending the original through the addition of timeline (E) to visualize detailed program execution timing, the Task Goals tile (F) presenting high-level guided steps to completing the task, and the documentation window (G) providing supplemental information about program blocks and issues.

if a Joint Speed issue is selected, it informs the user that the highlighted trajectory is moving too fast and presents the user with a graph detailing the robot's speed over time for each joint.

CoFrame is designed for users to build programs in the Program tile, generate issues, and address them one by one. As users build their program or select issues, the Simulator tile is updated to visualize and animate the robot's behavior, along with visually highlighting and illustrating the selected issues.

Updated Design & Implementation

Several updates and changes have been made to the *CoFrame* system since the original implementation (Figure 3.2). Namely, the replacement of the Contextual Information tile, the provision of additional textual information, and the addition of more visualizations to support user understanding.

While *CoFrame* still features a four-tile design, the Contextual Information tile was replaced with Task Goals. The new Task Goal tile (Figure 3.2 F) provides operators with a detailed task breakdown and was introduced as a way to support users through a series of tasks to learn cobot programming. Tasks can be manually specified by users, allowing them to be as specific or general as needed. Each step of the task breakdown adds support for previews, allowing for the visualization of objectives users need to accomplish, as well as automatically verifying whether the step has been achieved by the user's program. This automatic verification is accomplished using linear temporal logic properties (Rozier, 2011), and checks whether the user's program satisfies the property.

Information from the original Contextual Information tile has been redistributed across several new and existing components within the *CoFrame* interface. Previously, when a user selected a detected issue from the Expert Frames panel, a graph would appear in the Contextual Information tile to explain the issue. This graph is now embedded directly within the issue itself (Figure 3.2 D), more clearly communicating to users what the information is referring to. To further aid user understanding, *CoFrame* highlights the corresponding programming block that causes the issue and opens a new documentation window (Figure 3.2 G).

The documentation window provides users with block and issue-specific information. Information about issues includes why the issue exists, what it means, why it is an issue roboticists care about, and also provides high-level suggestions regarding potential ways to address the

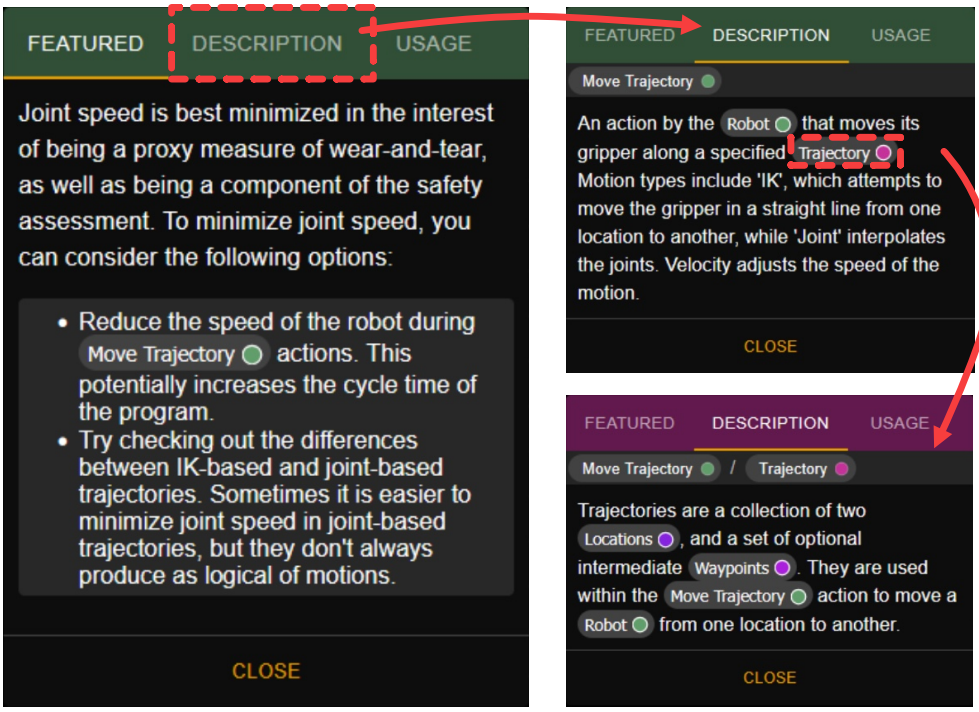

Figure 3.3: Interaction documentation window built in to the *CoFrame* programming environment. Operators can select issues from the Expert Feedback tile to receive information about the problem and how to solve it (Left), or understand block functionality (Top Right) and parameters (Bottom Right).

issue. In addition to issue-specific information, users can select program blocks in the program editor to view documentation about the use of the block. This provides users with information about what the block does and what parameters it accepts (Figure 3.3).

To help users better understand program execution, a timeline was added to the interface (Figure 3.2 E). The timeline component is initially not visible to the user, only appearing when program blocks are selected to run in the Simulator tile. It visualizes the timing of actions for the robot and the machines in the environment. If an issue is selected, such as

Joint Speed, *CoFrame* will highlight the corresponding programming block, visualizing the motion and showing the timeline to the user. Additionally, issues add a simplified visual of the accompanying graph onto the timeline, allowing users to see at a glance the problematic areas of their program.

A new issue detector was added under the Program Quality Expert Frame called Thing Flow. Thing Flow refers to the order of operations in producing and using Thing objects in *CoFrame*. This new issue detector checks user programs to ensure that, before interacting with a Thing object, the Thing is produced earlier in the program, allowing it to exist in the simulated environment. Building on *CoFrame*'s issue and frame dependencies, users must first address Missing Parameter and Missing Block issues before addressing Thing Flow issues.

The final update to the *CoFrame* system was the change to a fully web-based implementation. This allowed *CoFrame* to be accessed via browser, increasing system accessibility as it removed the need to set up a complicated backend for additional simulators.

3.4 Case Studies

Schoen et al. (2022); Schoen (2023) includes an initial set of case studies to illustrate how the system supported users in the creation of effective programs 3.4. These case studies were created in the context of a knife assembly task, based on the comments of experts (Siebert-Evenstone et al., 2021), where a cobot is programmed to move about an environment to assemble a knife from base components. Here we present an update to those case studies, highlighting the abstraction and scaffolding provided to the users through the updated design of the system.

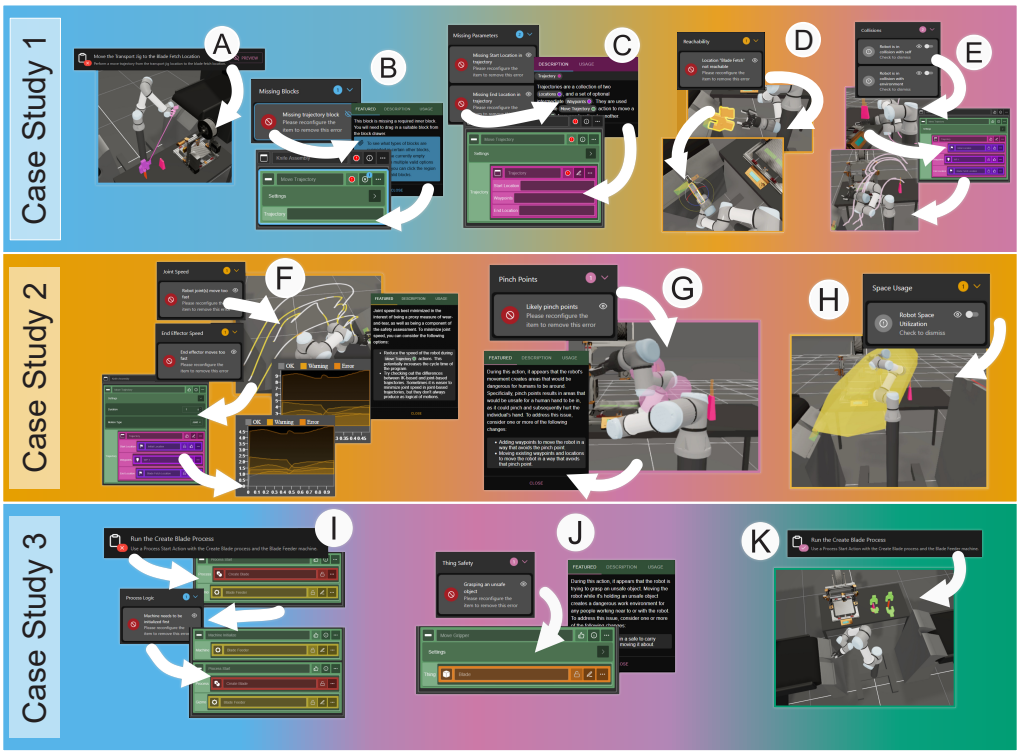

Figure 3.4: Three case studies showing the process of evaluating feedback from the system and informing adjustments to the operator's program. The gradient background of the figure denotes the switching between *Expert Frames* by the operator, from *Safety Concerns* (pink), *Program Quality* (blue), *Robot Performance* (orange), and *Business Objectives* (green). In Case Study 1, the operator begins by investigating the task goals and previews the goal (A). The operator then creates a trajectory and addresses a missing trajectory block (B), followed by filling in its parameters (C). The operator then addresses reachability concerns (D) and finishes by addressing issues with robot collision (E). In Case Study 2, the operator begins by addressing joint speed issues, visualizing the speed, and reading documentation to address the issue (F). They transition to solving pinch point issues (G) and finish by addressing issues with the robot's space usage (H). In Case Study 3, the operator begins by investigating the next task and solves issues with machine logic (I). The operator addresses problems with unsafe thing movement (J), and finishes by viewing their completed task (K).

Case Study 1: Defining a trajectory

Opening the interface, the operator first looks at the *Task Goals* to understand the task they are trying to complete. They see the first action calls for having the robot move through the environment, requiring them to create a trajectory to move the robot. The operator clicks on the preview button to see the desired action in the context of the environment. As they observe the animation playing out, they inspect each machine to determine where the robot needs to move first. They see visually that they need to move to the "Blade Receive" machine, which catches the blades as they arrive from the conveyor. The operator cancels the preview and notices the additional text accompanying the task, seeing that it confirms the location they need to move the robot to. Then they open the program editor's block drawer and drag a "Move Trajectory" block into the program. As they release the block into the environment, they see it is empty and has a red X icon. The operator hovers their cursor over the X and sees that it tells them the block is missing a required parameter. The operator moves to the Expert Frames tile, clicking the "Refresh" button, showing a number of errors with their program. As they move through the frames, they investigate *Program Quality* and see that under the Missing Blocks issue, it says the trajectory is missing. The user clicks on the visibility toggle to investigate the issue further. *CoFrame* highlights the "Move Trajectory" action, displaying a documentation window informing the operator that the block needs a "Trajectory" block from the drawer. As the operator adds the block, they see a new red X appear, informing them that the required parameters are missing. The operator decides to open the documentation window for the block, seeing that they need to specify two locations in the "Trajectory" block to have the robot move. They refresh the feedback and confirm that they need to parameterize the trajectory with a start and end location. The operator adds the location blocks to the trajectory, but notices that one of the locations has a triangle icon on it. As they hover their mouse, they see

that, unfortunately, their end location is unreachable. They click on the issue to bring up the location for editing and notice the robot got stuck in a joint state, preventing it from reaching the location. The operator adjusts the location. *CoFrame* then displays the new joint state that aligns the gripper with the location. At this point, the operator refreshes the Expert Frames tile again and sees that the trajectory has been fully specified. However, under the Safety frame, they see that the trajectory causes collisions with both the environment and itself. They add a waypoint, guiding the robot above the table while also avoiding colliding with itself. After a last refresh, the collision issues have been downgraded to warnings. Finally, the operator looks back at the *Task Goals* to see that the goal has been checked off, marking the task complete.

Case Study 2: Debugging a movement

After creating a valid trajectory, the operator notices there are a number of issues still present in the Expert Frames tile. They consult the *Robot Performance* frame showing active issues for joint and end effector speeds. Clicking the joint speed issue, the operator sees lines for each joint position over time appear within the Simulator. The operator also sees a more explicit graph depicting the speed of each joint over time under the issue. Then, the operator investigates the end effector issue, observing the corresponding graph in the issue. With the issue still selected, the operator looks over to the programming environment, where a documentation window has opened to inform them about the end effector issue caused by the move trajectory block. In the documentation, they see a suggestion for reducing the speed of the trajectory, so they open the motion's settings menu to increase the duration of the movement. Afterward, the operator switches back to the *Safety* frame to view pinch point issues. They see that the documentation window has been updated, suggesting that they add additional waypoints to the trajectory to better coax the robot to a joint

state that prevents the issue. After adding a waypoint, they click the feedback "Refresh" button to update the trajectory visualization. The operator is now prompted to address robot space usage for the trajectory. They notice that the robot extends out into the workspace more than necessary, so they again modify the waypoints, iterating over speed, collision, and pinch point concerns.

Case Study 3: Working with machines

Happy with the trajectory, the operator looks at the *Task Goals* tile to understand the next task. They see it tells them to use the blade feeder machine to run the create blade process. The operator revisits the scene to consider the machines' operations. They click on the blade feeder and see that it "produces blades," which they want to move to the assembly jig. They open the block drawer and place a "Machine Start" action after their trajectory, parameterizing the action with the machine's item block. They then place a "Machine Wait" action to force the robot to wait until the machine completes its action. Refreshing the feedback, they see a machine logic error informing them that the machine needs to be initialized before use. The operator adds the necessary action block, then adds a "Move Gripper" action parameterized with the blade, followed by a second "Move Trajectory" action. They iterate over the new trajectory in a similar fashion to the first one, though they add and adjust waypoints before seeking frame feedback.

Refreshing the feedback, the operator encounters a thing movement issue, which informs them that the program causes the robot to grasp an unsafe object. Clicking on the issue, the operator sees *CoFrame* has highlighted the problem in their program and focuses their attention on the simulation to find the transport jig, which they can use to safely carry the blade. They add additional sets of movements and grasps to move the transport jig to the blade feeder to get a blade that is safe to carry. Finally,

the operator checks the *Task Goals* tile to see that the task is marked as complete.

3.5 Discussion

While multiple industries face a growing skills gap that prevents the effective utilization of cobots, new systems can support cobot programming by bridging the gap in user knowledge and guiding user learning and exploration. *CoFrame* illustrates this potential, highlighting the multiple ways it supports learning and guides users through the process of creating cobot programs. Designing cobot applications requires specialized knowledge that is encoded in *CoFrame* through the *Safety First Expert Model* extracted from work by Siebert-Evenstone et al. (2021). This translation of the model into Expert Frames provides novice users with structured support in addressing cobot programming issues while highlighting key concepts experts pay attention to.

The update to *CoFrame*'s design provides users with more information in a targeted and focused manner, where they can learn and understand how to build cobot programs. It adds more cohesion across interface components, supporting the user's learning experience and their understanding as they move from one component to another to build programs and address issues.

As illustrated in our case studies, as the operator begins working through the process of creating a program, there are multiple sources of information and feedback for them to rely on to develop safe and effective cobot programs. The operator leverages expert feedback from the Expert Frames tile, watches the simulator to see issues visualized, sees highlighted parts of their program alongside contextualized feedback, and is provided with information for how to address the problem. Additionally, *CoFrame* slowly introduces the user to cobot concepts and programs by

guiding them through the steps of building programs, breaking down the task into steps in the Task Goals tile, and then through the Expert Frames tile to iterate and improve their program. This approach helps to structure and guide the cobot programming process for novice users, supporting their learning and exploration of cobot concepts and programming.

As cobot expertise in industry is difficult to find, this presents several challenges for the effective integration of cobots for working alongside human workers. We believe that new systems can increase the accessibility of cobot technology and bridge the skills gap by supporting the existing workforce in learning and understanding cobot concepts.

Limitations & Future Work

The work presented in this chapter has a number of limitations. Primarily, the current work discusses the design of the *CoFrame* system as it supports cobot programming, and does not include an empirical evaluation. An evaluation of the system should be conducted with both experts and novices to assess the perception and use of programming supports created through the translation of the *Expert Model* into *Expert Frames*. While we demonstrate the behavior of the system and illustrate how it supports the creation of cobot programs through several case studies, future work should compare the efficacy of cobot programming systems, such as *CoFrame*, to traditional methods.

Secondly, *CoFrame* makes several simplifications to support and ease user learning of cobot programming. *CoFrame* was built to support level-one (e.g., start-stop shared-space, time-separated) collaborations commonly found in industry (Michaelis et al., 2020), and future work should consider additional levels to support more complex forms of human-robot collaboration. Additionally, *CoFrame* assumes a simplistic model of physics for simulating programs, instead focusing on teaching users cobot concepts rather than having users spend time accounting for discrepancies in

physical interactions, such as grasping, that introduce additional considerations for program design. Finally, *CoFrame* is not designed to support real-time control, nor allow users an easy means of adjusting the tasks and simulated environment. However, integrating these features will allow for more general-purpose implementations and use cases, and should be explored in future work to assess how systems can support users.

While these simplifications are useful for a simulated learning environment, future work will need to explore how to bridge the simulated world to the physical while supporting user understanding across both. Currently, *CoFrame* does not account for real-world issues such as grasp detection or I/O management, nor does it integrate with physical robot systems. These are features that are desired by users and may assist in understanding cobot concepts. Future work should consider how to incorporate these features while supporting user understanding.

3.6 Chapter Summary

As industries face a growing skills gap, there is an increasing need for the development of new systems that support the existing workforce in creating effective human-robot collaborations. Current systems rely on specialized user knowledge and a deep understanding of collaborative concepts, which are currently lacking among users in industry. This need for deep understanding makes current systems insufficient for supporting the creation of cobot programs.

CoFrame represents the first step towards a solution to this problem. *CoFrame* encodes expert knowledge by translating the model created by Siebert-Evenstone et al. (2021), abstracts low-level robot implementation details so users can interact with high-level block-based representations, and highlights key issues and problems that can support user understanding of robot concepts. Additionally, it scaffolds user learning, allowing for

gradual introduction and exploration of concepts that support cobot program development through the use of Task Goals and Expert Frames that guide users on what it means to create safe and effective cobot programs.

This chapter presents an update to the design of the *CoFrame* system, leveraging abstraction and scaffolding supports to facilitate effective learning and cobot program creation. This is illustrated through a series of case studies that highlight how a user can leverage the system to learn about multiple concepts and create a cobot program.

4 EVALUATING COFRAME AND IDENTIFYING SUPPORTS FOR HUMAN-ROBOT INTERACTION PLANNING

This chapter presents our evaluation of the CoFrame programming system, in a lab study with both novice and expert cobot users to understand system perceptions and use of supports, and in a real-world deployment to understand CoFrame's capability to assist users in real-world applications. In this chapter, we discuss the motivation of the work, describe our approach to evaluating the system, present empirical observations and findings, and discuss both the use of the CoFrame system as it pertains to supporting end-users and the identification of new required supports to assist users in creating collaborative interactions. This chapter includes work from a manuscript in progress (White et al., 2025b) as well as previously published work in Sullivan et al. (2024).

4.1 Motivation

Cobots are increasingly utilized across various tasks and domains (Javaid et al., 2022) and hold particular potential within manufacturing settings (Liu et al., 2022a). This potential comes from the versatility and flexibility that cobots provide, as they are relatively easy to reprogram and repurpose without an integrator. Given their ability to work in conjunction with human workers and perform precise, repetitive actions, cobots possess a skill set that makes them very effective in tasks such as assembly, palletizing, packaging, kitting, and tool use for caulking, gluing, and sanding. However, most prior efforts to analyze cobot integration have focused on the associated engineering challenges that emerge following the choice to integrate.

Although there has been significant progress in the development of technical approaches to integration, several key questions remain: "*How*

can a cobot complement existing human-only work processes;" "*are there subtasks that human workers prefer the cobot perform*;" and "*is it possible to integrate a cobot while maintaining these preferences?*" These questions are important to address to ensure that a cobot is operating safely, being utilized effectively, meeting worker preferences, and has a positive impact on business outcomes, as these factors affect cobot adoption (Simões et al., 2019; Silva et al., 2022; Berx et al., 2022; Aaltonen and Salmi, 2019). Additionally, as these questions are not fully considered in the existing integration paradigm, organizational leaders, including owners and managers of small and mid-size enterprises (SMEs), may lack the knowledge or understanding required to make informed decisions about cobot integration in their workspaces. When cobots are integrated under these circumstances, the result may be poor utilization of their collaborative capabilities, disruption in existing worker processes, and only partial realization of potential improvements in overall business outcomes (Michaelis et al., 2020; Paliga, 2022).

In this chapter, we present an evaluation of the CoFrame programming system both in a lab and real-world setting[1]. For the lab setting, we focus on understanding the current issues and trends within industry as perceived by cobot experts, and the perceptions of system use and difficulties for novice robot programmers. For the real-world setting, we present the deployment of the CoFrame system and the process we employed to bridge the gap of supporting end users through the process of human-robot interaction conceptualization and planning. For this, we examine the decisions that occur prior to program implementation and propose an approach for collaborating with manufacturers. Our proposed approach includes four phases: **planning** for integration, **analysis** of existing workflows, **development** of new human-cobot workflows, and **presentation**

[1]The research in this chapter is derived from a manuscript in progress as well as previously published work by Dakota Sullivan, myself, Dr. Andrew Schoen, and Dr. Bilge Mutlu.

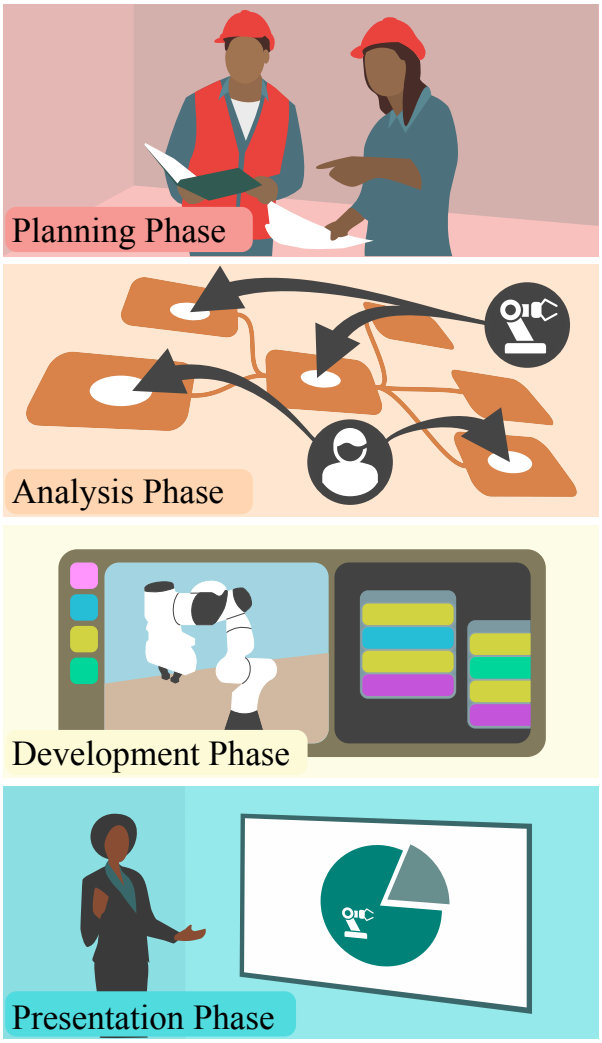

Figure 4.1: A depiction of the four-phase cobot integration approach proposed within this paper: planning for integration, analyzing workflows, developing simulations, and presenting to the manufacturer.

of results to stakeholders (Figure 4.1). This process allows stakeholders within manufacturing settings to make informed decisions about cobot integration, address questions related to worker and business preferences, and consider practical engineering constraints. To illustrate our approach, we discuss each phase within the context of our collaboration with an SME manufacturer. Following this discussion, we examine feedback from our collaborator.

4.2 Background

Cobot Usage

SMEs are increasingly using cobots in their processes, in part due to their marketed usability and benefits for collaboration (Simões et al., 2019, 2020), and the potential for reduction in cycle time of their processes (Enrique et al., 2021). Cobots can help to reduce repetitive tasks for operators (Marvel, 2014) and assist them in their tasks, such as by holding objects the operator is working on (Munzer et al., 2018). This practice of the cobot assisting operators is well explored within the research community (Peshkin and Colgate, 1999; Colgate et al., 2003; Bi et al., 2021).

However, prior work has noted that the usage of cobots by SMEs has primarily been as a cheaper alternative to traditional manufacturing robots, resulting in SMEs not fully utilizing their collaborative capabilities (Michaelis et al., 2020; Guertler et al., 2023; Wallace, 2021). In part, this under-utilization of the collaborative aspect of cobots can be attributed to the difficulty of finding appropriate tasks applications, misunderstanding how to utilize cobots effectively (Kadir et al., 2018), and a lack of knowledge regarding cobots by SMEs (Boucher et al., 2022). These findings illustrate the difficulty of successfully integrating cobots into existing manufacturing processes.

Factors for Cobot Integration

When beginning to integrate cobots into manufacturing facilities, there are a number of factors that must be considered. Existing work has identified the need to better understand work environments such that cobots can safely operate within them (Kildal et al., 2018; Malm et al., 2019). Maintaining safe operation requires consideration of factors such as crossover between cobot and worker work zones, cobot handling of objects, and cobot movement speeds, as these can create unsafe conditions for operation (Bi et al., 2021). Furthermore, certain cobot actions (*e.g.*, handling hazardous materials, moving quickly, or moving unintuitively) can create non-collaborative environments. These examples show that collaboration is dictated in part by a given task and is not inherent to the application of cobots themselves (Guertler et al., 2023).

Once a task is selected and initial workspace factors have been considered, additional interaction considerations must be made. Integrators must consider the ways individuals will interact with cobots to complete a task and utilize their knowledge of a cobot's capabilities to develop a collaborative process that optimizes operator needs and task outcomes (Grahn and Langbeck, 2014; Simões et al., 2020). To make a process collaborative, existing work has documented a set of guiding considerations (Malik et al., 2021), such as workspace configuration, ergonomic impact, types of interaction and collaboration that occur between the operator and robot, and understandability of cobot actions to the operator.

Task scheduling is well explored in automation (Lewandowski and Olszewska, 2020; Yin et al., 2018; Wang and Li, 2019), and cobots provide new variables that integrators need to consider, as they pose new ways of dividing, sharing, and collaborating on tasks between the worker and cobot based on the type of interaction (Christiernin, 2017). While algorithms and approaches for addressing this challenge exist (Sadik and Urban, 2017; Tsarouchi et al., 2017), it is important to consider the ways in

which a cobot can assist the human operator more directly, as it is commonplace for operators to adjust their own workflows to work with cobots (Wurhofer et al., 2015). While cobots can improve an operator's physical working conditions (Peshkin and Colgate, 1999; Cardoso et al., 2021), this capability is dependent on which tasks are selected for the cobot and operator to perform. It is important to consider the operator's preferences and trust between the operator and cobot, as these are important factors in determining resulting task performance (Kopp et al., 2021).

Integration Frameworks and Approaches

There are many key factors that need to be considered when approaching cobot integration. One of these factors is the selection of candidate workcells and processes for cobot assistance. This step can be completed by identifying any manual processes that may be a bottleneck to other processes (Cohen et al., 2019), or through analyzing return on investment over long-term usage (Gil-Vilda et al., 2017). Another step that must be completed is the configuration of an effective workcell (*i.e.*, developing a simple, modular, and safe design for workers) (Malik et al., 2021) while taking into account productivity (Gil-Vilda et al., 2017) and interactions between the cobot and worker (Malik et al., 2021). These workcell designs need to consider the potential for the cobot to work in parallel with the worker, either by having the cobot work in a separate area of the cell or by collaborating with the worker directly (Andronas et al., 2020). Additionally, an appropriate cobot must be selected for integration based on the context in which it will be situated. Prior work has investigated how to make this decision, based on the requirements of the task, the properties of the robot, and its potential performance (Cohen et al., 2019; Ghorabaee, 2016).

Several of the above steps have been encapsulated by the National Institute of Standards and Technology (NIST) within their set of guidelines for

cobot integration, based on discussions with robotics experts (Horst et al., 2021). In their work, they present several concrete methods for identifying candidate workcells for integration, metrics for selecting a cobot, and metrics for determining the viability of a given integration plan. Overall, NIST provides several steps to begin the integration process, as well as metrics and considerations to use in the decision-making process. Other work has explored a method of integration which begins at a general level, by understanding the task context, and then considers specific elements such as the workcell, cobots and other machines, and, finally, the workers (Djuric et al., 2016).

However, recent work has acknowledged the technological focus that exists in prior approaches to cobot integration, as well as the recent shift towards incorporating a socio-technical perspective that considers the worker and cobot a partnership rather than as individuals (Adriaensen et al., 2022). While these technological approaches have defined many important factors and methods for integration, they fall short of incorporating both manufacturer and worker considerations while also demonstrating system feasibility.

4.3 Evaluation Approach

To understand how CoFrame addresses the challenges in industry and how novice users experience such systems, we conducted three studies, one lab study with industry roboticists knowledgeable in cobots, which we denote as *Expert Evaluation*, another lab study with novice users, which we denote as *Novice Evaluation*, and a real-world deployment case study where we deploy CoFrame with a SME partner, which we denote as *Real-World Deployment*. Sections 4.4-4.5 discuss the lab studies used to evaluate CoFrame, while sections 4.6-4.8 discuss the real-world deployment of CoFrame. Finally, section 4.9 presents a general discussion on the design

and deployment of programming systems.

4.4 Lab Study

Participants

For the two lab studies, we recruited a total of 3 industry experts with knowledge of cobots (3 male, age $M = 37.67$, $SD = 10.66$), and 13 novices (9 male, 3 female, 1 no response, age $M = 23.15$, $SD = 5.39$). The experts were recruited by contacting cobot integration companies. The novices were recruited in person on a University campus, through flyers posted in common areas, as well as through university mailing lists. Novices were recruited from several disciplines, including mechanical engineering, electrical engineering, industrial & systems engineering, and computer engineering, and were required to affirm they have some knowledge of robots. Studies lasting around 1 hour in length for which participants were compensated for their participation, with experts receiving $\$48/hour$, while novices received $\$12/hour$.

Measures

Within this study, we capture participants' perceptions of CoFrame through the User Experience Questionnaire (UEQ) Schrepp et al. (2014) and semi-structured interviews. The UEQ is a 26-item questionnaire composed of six subscales (*i.e.*, attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty). Each item within the questionnaire is a seven-point semantic differential scale that allows users to characterize their experience. While a benchmark is provided by the authors of this questionnaire Schrepp et al. (2017), we do not include it within our analysis and results. CoFrame, while intended for both novices and experts, is used for a specialized application (*i.e.*, programming of robots), unlike

many of the products included within the benchmark (*e.g.*, web services, social networks, mobile applications, and household appliances).

During our semi-structured interviews, we asked participants questions regarding the features in CoFrame they liked, disliked, and would have liked to see modified. For example, we asked participants, "*What did CoFrame do well*," "*What did it not do well*," and "*How would you suggest changing the system to address that?*" These questions allow us to better understand what specific features are most and least helpful to participants and why. We then conducted Thematic Analysis (TA) on the semi-structured interviews using the guidelines proposed by Clark and Braun Clarke and Braun (2017). This process required transcription of interviews, familiarization with the data, iterative coding, and development of themes.

Lab Study Procedure

Expert Evaluation. Participants were first informed about the purpose of the study and provided consent to participate. Participants then engaged in a semi-structured interview ($10-15$ minutes) about the use of cobots in industry, the different skills they believe to be important for using cobots, what tools and processes they currently utilize for cobots, and what some of the biggest issues that need to be addressed are. Following the interview, participants were then shown a brief video tutorial on the CoFrame system that explains the multiple tiles and program blocks, as well as demonstrating how to use the system. Participants were then given 15 minutes to use the CoFrame system. They were told they could explore the system freely or follow the set of predefined task goals to build a program. At the end of their interaction, participants were asked to fill out a UEQ survey, as well as demographics. Next, participants engaged in another semi-structured interview ($10-15$ minutes) regarding their experience with and perception of CoFrame, how CoFrame addresses some of the issues they presented in the first interview, as well as its

applicability in solving the problems faced in industry. Example questions include ”What did you like and dislike about CoFrame?”, and ”Do you see CoFrame as being applicable at any stage of the cobot integration process?” Finally, participants were compensated for their participation.

Novice Evaluation. Participants were first informed about the purpose of the study and provided consent to participate. Participants were then shown a brief video tutorial on the CoFrame system, explaining the multiple tiles and program blocks, as well as demonstrating how to use the system. Participants were then given 30 minutes to use the CoFrame interface. During this time, participants were asked to follow a predefined set of task goals within the system, completing as much as they could within the timeframe. These task goals had the participant move the robot around the simulated environment, interact with various objects, and combine those objects to produce some final product. Participants were asked to think aloud while they built their programs, and after they either finished building their program or reached the time limit, they were asked to fill out a UEQ questionnaire and demographics. Next, participants engaged in a semi-structured interview $(10-15$ minutes$)$ regarding their experience with the interface. Example questions include ”What did you like or dislike about using CoFrame?”, and ”Why did you do X?” where X is some action they took in building their program. Finally, participants were compensated for their time.

4.5 Results of Lab Studies

Here we present the findings of our study. First, we present the quantitative results of the UEQ questionnaire, discussing initial high-level user perceptions of the CoFrame system and its comparison to the benchmark. Then we present the results of our Thematic Analysis, highlighting expert

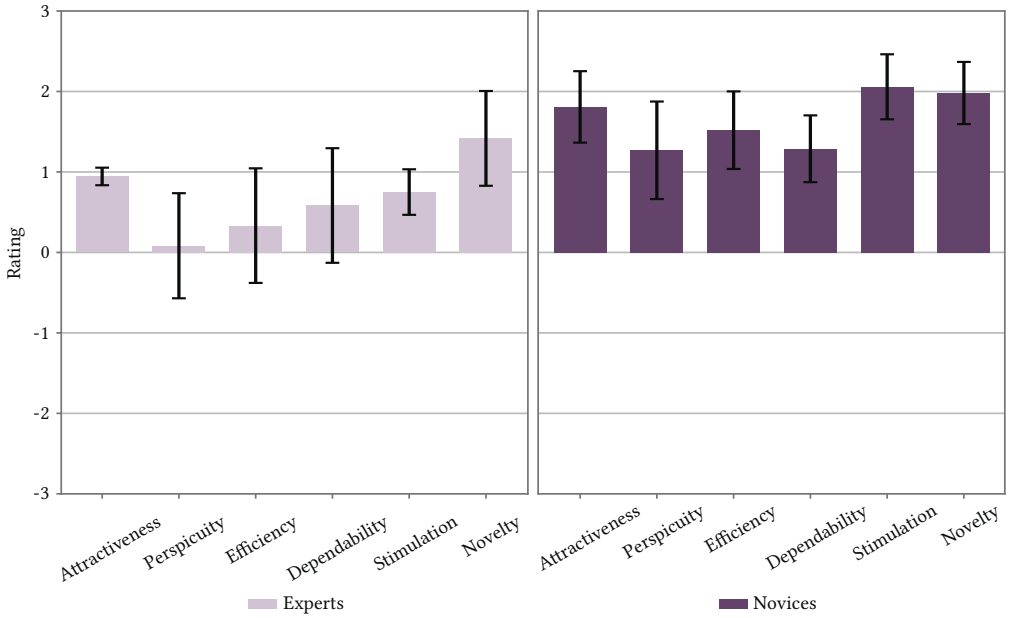

Figure 4.2: User Experience Questionnaire results including the average scores of novices and experts and confidence intervals. These results show that both novice and expert participants regard CoFrame favorably with positive average scores across all six UEQ subscales.

perceptions of difficulties in industry and the perceptions from novice's experience interacting with the CoFrame system.

UEQ Results

Following their exploration of CoFrame, participants completed the User Experience Questionnaire. The results of all six usability subscales of the UEQ can be seen in Figure 4.2. Novice participants rated all six UEQ usability aspects positively with stimulation receiving the highest average score ($2.06, SD = 0.74$), followed by novelty ($1.98, SD = 0.71$), attractiveness ($1.81, SD = 0.82$), efficiency ($1.52, SD = 0.89$), dependability ($1.29, SD = 0.76$), and perspicuity ($1.27, SD = 1.12$). Similarly, expert participants also rated all aspects positively, with novelty receiv-

ing the highest average score ($1.47, \text{SD} = 0.52$), followed by attractiveness ($0.944, \text{SD} = 0.096$), stimulation ($0.75, \text{SD} = 0.25$), dependability ($0.583, \text{SD} = 0.629$), efficiency ($0.333, \text{SD} = 0.629$), and perspicuity ($0.083, \text{SD} = 0.577$).

Experts Viewed Cobots as Difficult And Inaccessible

When discussing the current state of cobots and cobot support, experts discussed cobots as being complicated, with E1 describing there being "so many areas where an impact could be made." E1, familiar with CNC machining, framed the current state of robotics as where CNC was 30 years, needing tools that better support users in using cobots without requiring them to work at the lowest levels.

> "*30 years ago, most people were programming CNC machines by writing the machine code by hand, or by using some kind of assisted features at the CNC control to teach the machine how to do what you wanted it to do. And now, almost nobody does that, you got to design the part in CAD, you go over to CAM, which is computer aided manufacturing, and you design the toolpath* [...] *press Cycle Start, and then it just makes a thing for you* [...] *the world of robotics is very much where the world of CNC was 30 years ago*" – E1

E3 expressed a similar concern over cobot difficulty, stating that "there are a lot of limitations and there are things that humans intuitively will do," resulting in experts having to "really think hard" about how they'll incorporate the robot within tasks. E2 explained that to try and overcome this, they work in teams so they can rely on each other's expertise to "keep pointing things out" and that "it's really just trying to get them to look at the scenarios in the right way and to identify a risk. And then again, determine if you need to do something different to mitigate that risk." E2 further explained that working in teams is especially critical for

new hires, as they rely "on the expertise of the seasoned person that's got more experience, to ensure that whatever they're doing is safe." To be successful, E1 explained that "people have to come to the table with enough understanding of how a robot works" and "how it's programmed."

Expert E3 did not see cobot programming as particiularly difficult, saying that "if you can program industrial robots, programming is easy for cobots [...] except for a few additional safety features," and that it has "similar difficulties you might have just in general programming." However, E3 also noted that it may be more difficult for those who are not familiar with cobots, saying "if it's an operator that doesn't have any background or higher education, which is common in this industry, that might be tough just because of, you know, that background isn't there." E3 viewed this as a training problem, saying that "if we were to bring new people on, we just send them over to [person] and have them take a programming class."

In addition to programming, experts viewed the current state of programming tools as insufficient for supporting cobot program creation and facilitating understanding. E1 viewed teach pendants as insufficient by robot manufacturers, saying that while there are "clearly basics that people need to understand before they come to the table" and that "robot manufacturers could certainly do a better job of creating more usable interfaces for people to work with." E1 expanded on this difficulty with the teach pendant, saying that if you "give someone a Fanuc teach pendant and tell them to figure it out on their own" then "they're going to be lost even if they you know, understand the basics of industrial robotics." However, new systems show promise in addressing several of these issues. E3 viewed new programming systems as being able to support multiple users of different skill levels, with the goal being that you can "have operators that aren't necessarily super technically proficient, but they can still set up a cell, and then they can easily like, say, redeploy it for other operations."

E3 mentioned several recent systems they were aware of that were being advertised as being usable "without much programming skills" to "get a certain operation working." Similarly, E2 mentioned that they have seen recent tools with "more of the AI type stuff, where it will help you write stuff on its own and risk assess as part of the software." E2 felt that these would better support user programming, saying that "while I don't necessarily know they're going to catch everything. The more questions that can be asked, the better."

Overall, cobots were generally seen as inaccessible by experts, making it difficult to communicate how to use them or demonstrate capability. E1 viewed this inaccessibility as being due in part to the fact that "they're kind of hard to use, they're pretty darn expensive," and "there's not a lot of community support or tutorial material available." When elaborating, E1 reaffirmed that part of the issues is the "availability of hardware" and the other part is "the usability of the hardware." E3 had similar concerns about cobot accessibility, but mainly for communicating their capability and potential. When working with customers E3 said it is important to be "transparent with both of the robot and of [humans] in terms of integrating [the robot]," but that "getting [customers] open to the idea of having robots doing certain tasks is the hardest part" due to not being able to effectively demonstrate or communicate those capabilities and limitations.

Perceived usage of CoFrame

CoFrame was perceived positively by both expert and novice participants. While experts were given 15 minutes to explore CoFrame, leveraging it as a design probe for discussion, their impressions were overall positive. All experts particularly liked the visualization and highlighting the CoFrame uses to call attention to issues. E1 expressed this by saying "I thought this was quite cool, especially the pinch points and collisions." Similarly, E2

liked the expert frames and the focus on safety, saying that the "ability to look at this and kind of run through the process" while having the system highlight that "this is something you need to pay attention to" is "big." E3 liked that the use of these issues guided program development, saying "I like the fact that it shows the safety concerns for you" especially for "things we don't always consider" because for "collaborative robots, that's very important." E3 connected these issues to their process of building cobot applications, saying that "one big thing when we're working on field service or something, is kind of identifying exactly what the objectives are of that service" and "this kind of does that explicitly." E2 similarly expressed a connection between the frames and their process for building cobot applications, saying that the spreadsheets they use to analyze interactions for safety concerns are "the exact same thing" as what CoFrame does and that "it's really just a structured way to try to look at it to come up with those ideas on how you make that risk less"

Novices also generally liked CoFrame, with N13 explaining that "the UI was very [...] welcoming in terms of adding blocks, and [...] clearly defining what exactly each thing does." N10 liked that everything was within a single application, saying that they "haven't seen software where you can, actually, you know, program your stuff, run your stuff, see what like, how your robot is moving, and then your robot moves." N5 found that the task breakdown in the Task Goals section was "easy to understand" and they liked how "I knew what I was supposed to do" but "not necessarily right away like how to do it" as it provided an opportunity to explore and learn. N4 liked CoFrame's delineation between programming the robot and the machines used in the task, saying that "I liked the general idea of like splitting out the the robots, actions, and the different processes happening and looking for things like that." Similarly, N2 found that CoFrame forced them to think about the machines used in the process, as they "never thought about how those are programmed in, because that

might be really annoying."

Several novice participants expressed that they liked the structure and use of the expert feedback issues. N5 explained that they "like that it gives solutions," referring to the suggested ways to address the detected issue, and said that it "gives me the options" for how to address the problem while simultaneously highlighting the corresponding program block so that they "knew what section it was." N11 similarly expressed that the use of the feedback allowed them to "make adjustments" resulting in it being "easy to learn."

When comparing experts and novices, experts viewed the system through a different lens than novices, relying on their experience with other software and familiar workflows. This resulted experts perceiving CoFrame with having relatively high perception of novelty, but lower – although positive – views on its ability to facilitate programming. E1 explained that CoFrame could lean on systems such as "Autodesk fusion or Solidworks" to have similar, familiar functionality for users for the 3D simulator. Additionally, E1 mentioned how within RVIS "when the robot actually does collide with itself, or is going to, [...] it'll turn one of the robot's two offending links that collide with each other, it turns red." E2 expressed that CoFrame needed to be able to connect to the physical robot, as "it's a little bit trickier always when you're looking at a screen in 3D." For E3 this would also allow for more detailed information about the robot's state when using the gripper, as they frequently need to know "what that means for the I/O, like how we're actually closing our gripper."

However, both experts and novices saw value in CoFrame's use for multiple use cases in education and industry. N2 expressed that cobots are inaccessible for schools, saying that "you can't ship an a robot arm to like, you know your middle school," but highlighted that systems like CoFrame allow you to "just send them to a link and like teaches kids how to step by step think of like a simple movement of your arm." N2 further explained

that it is easy for them to take "those things for granted of like how that stuff moves" due to their experience and familiarity with robots, but that systems like CoFrame can allow users to "get into this like how processes work" and how to "do the hierarchical thinking." Similarly, N3 expressed that CoFrame could be used for education, saying "I think this is [...] a good program for beginners to get into how thinking for robotics actually begins, you know. It's not just about code, but also about the position, start location and location and all that." However, N3 also saw CoFrame as beneficial for industry usage, saying it would "make a lot of workflows smooth or gives control to some person, [...] I guess even debugging would be provided and all that." N3 further explained CoFrame's applicability in both areas, saying that "it's more dependent on what the creative goal is" but that they "don't see how [CoFrame] adds more strength towards one aspect or the other. I don't see that. I mean to me it looks like it's equally usable in both." One expert also discussed CoFrame's multi-use potential. E2 thought that tools like CoFrame could be "extremely useful at every stage. At the sales stage, because it's relatively simple and easy to put together. I could see putting together a simulation like this so that way, at least on a baseline, you know you've sold the right safety components, you've sold the right pieces, and I could see it being helpful that way, all the way through to maybe the final, final programming phase, depending on on what actually comes out of this programmer." Overall, E2 said that they were "very excited to see tools like this coming out. I think they're going to help out greatly in the future as as we get more and more people involved with this, and not necessarily relying as heavily on the experience and expertise that you have to have in order to successfully do this today."

Difficulty Interacting with System Supports

Several users expressed a desire to make actions more visible to the user within CoFrame. N10 expressed the task goals completion was "not no-

ticeable enough," and that they would prefer "if there was a pop up" to tell them when each step was completed. Another user, N12, desired for clearer indication within the simulator of occurring actions, having noted that when they used the processes to have the machine produce parts the simulator "doesn't show you doing that. So you're not actually sure if it's working." N12 expanded on this idea, opening it to additional visualizations to indicate future actions and goals, saying that it would be "useful [...] if, while you're making a trajectory, it like shows an outline of what's gonna happen."

In part, this desire for increased visibility stemmed from user difficulty with accessing pieces of information within CoFrame, from selecting blocks to simulate to interacting with the expert feedback. N13 felt it was difficult to select which block to simulate. However, while N13 understood that "you can have multiple processes" within a program, they expressed that having a "dropdown" which lets you select the process "you wanna run" would be preferred instead of "scrolling all the way up and then selecting that process." A few novice users had issues with accessing the expert feedback information. This was partially caused by CoFrame requiring users to request feedback, which users were split on whether they liked it or preferred the feedback continuously generating while they built their program. N6 said they preferred to request feedback, as "If I at least get all built, then all the issues that will ever exist will be in that tab when I get to it." N13 expressed a similar sentiment, wanted to "first set everything up and then understand our issues" because "if it's constantly giving me a feedback, then it it kinda interrupts with my thought process." Other users expressed wanting the feedback panel to update continuously as they built their program, providing continual feedback that they can choose to engage with or ignore. N2 thought the use of a button to request feedback was bad, saying "the button [...] is a very computer programmer mindset." N7 elaborated on this, saying that it could "refresh on its own

after some time" so that users does not "forget refreshing and noticing" problems in their problem. However, continually refreshing the expert feedback panel would make the feedback obvious to users, as one participant, N1, expressed that because the feedback panel did not show errors during their interaction they didn't think there were issues to fix in their program, saying "I just kind of saw that everything was already checked off. And so I figured once I get through tasks, once I have things set up, I can look at the feedback and improve it if I want to."

A few users expressed confusion over the expert feedback panel and the issues, finding that clicking the eyeball to select an issue was not intuitive. N4 expressed this best saying that "I didn't realize that I had to click on the eyeball. And then I didn't realize what that was doing when I clicked on the eyeball at first until you told me it was highlighting something in pink, and then I went to go look." One user, N8, suggested moving the feedback from the panel to the blocks directly, to tighten the coupling between them, saying "I guess why have it as a separate panel on the left versus like, what if you had a little badge, icon, or something on each of the steps on the right that would provide feedback, like directly on the step itself. directly on the block itself."

While participants liked the drag-and-drop programming, the implementation within CoFrame making the environment draggable and zoomable resulted in confusion and frustration by users and impacted user ability to engage with the documentation window. Users expected to be able to scroll the program, like a typical written program, but instead were met with zoom-based draggable features, with N13 saying that the draggable environment was bad because "I'm required to like, you know, zoom out and then move everything up. [...] It's not that intuitive." As users zoomed out to see their program, the individual blocks became small and not as noticeable. N6 explained that they "didn't like how small everything got" because they felt like they needed to zoom out to effec-

tively interact with the program. N13 elaborated saying that they found the zoom feature to be "a little painful" because they were "zooming in and out navigating these coding blocks and moving them up and down like from the UI perspective, I just didn't like it." This resulted in issues when viewing expert feedback from the feedback panel, as attempting to highlight specific blocks and provide the corresponding documentation resulted in it going unnoticed. N4 said that they were confused because "you're telling me all these things" but because their program was so large and they were zoomed out "I don't know where specifically these things are happening. And I. So I don't know what to do about them." A few users expressed that because of the zooming, they struggled to notice the highlighted program blocks, and occasionally weren't able to find the information panel due to it being off-screen and requiring the user to drag the environment to view it. N4 explained that "the highlighting wasn't very obvious" because it was difficult to see "a really thin pink line" when they were "often like zoomed all the way out to view like the whole thing."

Supporting Advanced Usage

Several participants talked about the eventuality of requiring CoFrame to allow for more in-depth tools or capabilities, extending its capability to support more advanced usage.

While participants liked the block-based programming system, one novice expressed a desire to understand what lower level code the blocks translated into. N3 explained that "it would be also nice if we could actually look at the code this generates. You know, so that educational way, they'll get to know what's going on under the hood." N3 further explained that this also benefits users by allowing them to extend functionality, saying that if users wanted "to do something else, you know, that's not available in this program. They could take it up independently. by adding their own blocks to it."

Several participants also desired for a stronger connection between CoFrame and the physical robot. N4 mentioned they were "curious about like the angle of joints and stuff like that," and that while they liked the simulation they would want to occasionally see "some sort of numerical output of it or something." While CoFrame does provide users with joint angles for specific user-created locations and waypoints, these appear in a menu when selecting the locations and waypoints, which no user utilized within the study. In addition to joint states, one novice user expressed a desire to interact with the robot physically to move and orient it for use in storing locations and configurations they can then use in their program. N12 explained that "in my engineering program in high school, [...] we worked with like an actual arm," which allowed them to use the teach pendant to "manually move the robot, create a point and then come back to that point later."

One expert also discussed the need for more advanced capabilities than what CoFrame allows. After interacting with CoFrame, E3 said that "with the gripper closing" they would need additional sensing and information to understand "what that means for the I/O, like how we're actually closing our grip." E3 elaborated on this saying I/O sensing and information was very important because "a big part of things is, especially with our machine tending, we're going to a PLC and it's a big part of the program that we're, you know, when we get to the machines, we're opening the door properly."

4.6 Real-World Deployment

Following the lab studies, we attempted to deploy and evaluate the CoFrame system in a real-world context with a small-to-medium enterprise (SME) partner.

Our team worked with a local SME manufacturer that expressed inter-

est in cobot integration. In working with our collaborator, we took several actions to ensure the privacy and confidentiality of the business as well as individuals with whom we interacted. First, we have omitted the identity of our collaborator in this paper and supplemental documentation. Second, our institution signed a non-disclosure agreement (NDA) with our collaborator, and the research team sought permission from our collaborator to publish the material presented in this paper. Third, we collected information (*i.e.*, video recordings, process information, and feedback) only after verbal consent was obtained (in process analyses) or consent forms were signed (in feedback sessions).

While our collaboration lasted approximately eight months, this time period included iterative modifications to the software tools we used for analysis and simulation. We expect our proposed approach to take less time with a strict minimum of four meetings (*i.e.*, initial discussion, collection of data, overview of implementation, and presentation) and additional meetings as needed depending on the nature of the collaboration. Therefore, we expect our approach to roughly require a time commitment between a few days and a few weeks. More streamlined software tools and organizational commitment can shorten this timeframe to a few hours.

During our initial visit to our SME partner, we presented on and discussed the use of the CoFrame system. Our partner was interested in the software, but expressed not knowing how to approach the use of the system, as they were not entirely sure where or how to incorporate cobots in their manufacturing process. Through this discussion, we learned that our collaborator was hesitant to test the software without first better understanding cobot capability and potential applications, desiring to be reasonably sure that a given application was well suited for integration prior to exploring software capability. The following sections illustrate our approach to collaborating with our SME partner, as well as discuss how we executed that approach.

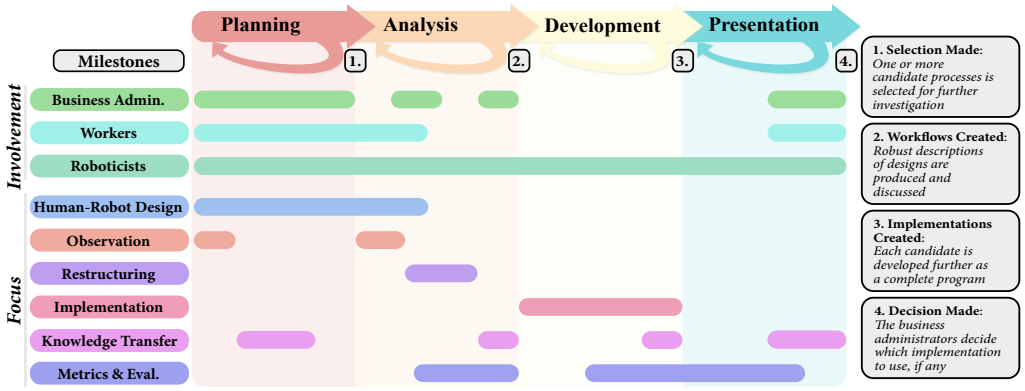

Figure 4.3: An overview of the four-phase approach including the individuals, foci, and milestones involved at each phase.

4.7 Approach

We propose a four-phase approach to help businesses understand the costs, outcomes, and implications of cobot integration in order to make better-informed decisions (see Figure 4.3). These four phases are *planning* (*i.e.*, understanding the context of cobot intervention), *analysis* (*i.e.*, defining the roles of the human and robot), *development* (*i.e.*, creating a new workflow involving the human and robot), and *presentation* (*i.e.*, gathering and presenting relevant information to the collaborator). These four phases provide a pathway to the integration process that business administrators can utilize to understand the effects of integrating a cobot into their workflows before making major commitments.

Planning Phase

The initial *planning phase* of our approach attempts to develop an understanding of existing workflows and allows the roboticist to ground their expertise within the context of these processes. This approach builds on the ideas of contextual inquiry (Beyer and Holtzblatt, 1999), where

observations and interviews are combined in order to develop a thorough understanding of a conceptual space. Thus, this phase requires the involvement of all three parties (*i.e.*, business administrators, workers, and roboticists), and places an initial focus on applying the roboticist's understanding of HRI through observations and discussions. This initial understanding may be achieved through on-site visits and tours, or a series of discussions regarding existing processes and workflows currently completed by human workers alone. Within these discussions, it is important to identify contextual factors including the workspace layout, spatial constraints, and resources required for a specific process including labor and parts. Additionally, it is important to identify opportunities for collaborative assistance by the cobot. While there may be several tasks or subtasks that can benefit from cobot involvement, a discussion of worker and business preferences (*e.g.*, reduction of undesirable work or optimization of critical tasks) will guide which tasks are most appropriate. For manufacturers with limited prior knowledge of cobots, this phase helps to ground any potential ideas for integration based on the realistic capabilities of cobots and helps to establish expectations of the impact a cobot could have. Follow-up discussions should occur as often as necessary for the manufacturer to gain a sufficient understanding of cobot integration and to agree upon the best possible task candidates for cobot intervention. By the end of the planning phase, manufacturers should have a cursory understanding of what cobot integration requires and yields, and roboticists should have one or more candidate tasks they can begin to analyze in the next phase.

Analysis Phase

Once the planning phase is complete, the roboticist can begin gathering data on the existing human-only work process for review in the *analysis phase*. This phase will initially require the involvement of all three parties

as the phase focuses on data collection. The gathered data may include blueprints of the work environment and its configuration, videos of process execution, timetables of task steps, subtask dependencies, component schematics, or other forms of data that describe the work process in fine detail. After data collection, the involvement of the worker and business administrators is lessened. From here the roboticist uses the collected data to concretely understand the environment and existing workflows so they may be restructured. The restructuring process initially involves analyzing the workflow, which can be done through methods such as hierarchical task analysis (Stanton, 2006), a method used to break a task down into goals and subgoals to understand its operation. Once a given task is understood at a granular level, the roboticist can complete the restructuring process by dividing the overarching goal into subtasks for the worker and the cobot based on worker preferences, robot capabilities, and overall optimization of the task. This process may place particular emphasis on limiting human or cobot idle time to allow for optimal efficacy of the human-cobot team. However, it is important that this process of assigning subtasks leverages the roboticist's knowledge of cobots and their capabilities, and incorporates principles of human-robot interaction and ergonomics. This practice allows the roboticist to ensure that the cobot acts as a effective collaborator, assists the operator in a safe manner, and improves overall task performance (Paliga, 2022; Simões et al., 2022; Cardoso et al., 2021). Given the unique stakeholder preferences that need to be considered in creating a new human-cobot workflow, continued discussion with the collaborator may be necessary to ensure that desired outcomes are achieved. During this phase, it is important that the roboticist develop an understanding of where and how the cobot can be optimally inserted within the existing workflow. Additionally, the roboticist must be aware of potential failure points caused by limitations in the cobot's capabilities. For example, if a particular component does not have convenient grasp

points, manipulation of such a component may be a task better suited for the human worker. At the end of the analysis phase, a new human-robot workflow should be produced and communicated to stakeholders for high-level feedback. Based on this feedback, the roboticist may need to iterate on prior planning and analysis steps.

Development Phase

During the *development phase*, the goal of the roboticist is to operationalize the newly created human-cobot workflow and produce outcome metrics that communicate the workflow's performance. This phase will primarily involve the roboticist, as they initially focus on the implementation by creating a simulation of the new workflow process in software systems such as Unity (Bartneck et al., 2015), RViz (Kam et al., 2015), Webots (Michel, 2004), or CoFrame (Schoen et al., 2022). The purpose of the simulation is to act as a general proof of concept that showcases where and how the cobot operates within the environment and demonstrates the feasibility of the new workflow as the collaborative process is executed. Once the implementation is complete, the roboticist will begin to produce metrics derived from the simulation in tandem with updating the implementation as needed. These metrics should account for the process's cycle time and the robot's idle time and include information about potential safety concerns and their mitigation. Based on the workcell setup and cobot that are utilized within the simulation, a roboticist can begin to approximate the price of components needed to recreate the simulation within the manufacturer's facility. As this integration plan is developed, potential flaws may become evident, thereby necessitating additional stakeholder discussion and iteration on prior completed steps. By the end of the development phase, the roboticist should have a concrete integration plan including the simulated workflow, process outcome metrics, and approximate component costs.

Presentation Phase

In the *presentation phase*, the roboticist synthesizes all information from the prior three phases and discusses it with the business administrators and workers. These data may include the procedure of the new human-cobot workflow, subtask timetables, process performance metrics, equipment or labor costs of integration, and overall profit per produced item. Once gathered, these data can be compared to the existing human-only work process and analyzed to determine the relative costs and benefits of the human-cobot task procedure and hardware installation or any variants that may have been developed. This information can be complex and cumbersome, so the roboticist may develop recommendations based on specific overarching needs and preferences conveyed by collaborators. All information should be formatted for submission to the manufacturing collaborator (*e.g.*, within a presentation or written report) and then discussed to ensure they fully understand the results and have any questions or concerns addressed. Once the presentation phase is complete, the manufacturer should have a thorough understanding of the potential integration plan, its outcomes, and any other information required to make an informed decision about cobot integration within their facility. From this point, additional discussion and iteration on proposed ideas can occur depending on the needs of the business and nature of the collaboration.

4.8 Case Study Application

Using our proposed approach, we present a case study of its application with our collaborator. In this section, we present the application of each of the four phases, actions that were taken in each, and feedback provided following our initial collaboration.

Planning Phase: Stakeholder Discussions

Our initial meeting with our collaborator involved touring their facility, discussing business needs and worker preferences, and seeking preliminary opportunities for cobot integration. From this initial meeting, we learned that the business administrators wanted to increase efficiency in their process to fulfill more product orders and saw cobots as a means of meeting this need. Additionally, the administrators expressed interest in having a cobot take over undesirable and messy tasks from the workers, a sentiment that was echoed by the workers themselves. The result of this initial visit helped set expectations with the manufacturer and allowed us to identify several potential areas for a cobot to assist in their process. Following this visit, our research team convened to discuss potential options and scheduled a follow-up meeting with the manufacturer to further discuss the potential of each option.

During the follow-up visit, we developed a deeper understanding of the various tasks that could potentially benefit from cobot intervention. By observing worker processes, we were able to identify tasks that were repetitive or undesirable to workers. At the end of the visit, we discussed with the administrators which tasks would be best suited for cobot integration given our knowledge of cobots, our understanding of each process's potential for collaboration, the administrator's desired business outcomes, and the preferences of workers. From this discussion, an assembly task was chosen which required the collection of parts, silicone application, rivet fastening, and other subtasks. This particular task involved the assembly of high-volume units, which was a high priority for reduced cycle time and could lead to an increase in their throughput and overall ability to fulfill orders. The application of silicone in the assembly task was seen as extremely messy and therefore undesirable to workers, while also being ideal for the cobot given its ability to perform precise and repetitive motions. As a second candidate, we identified that the cobot could assist

further by supplying components and then applying silicone to them. Overall, the selected task was a clear fit for cobot integration and was the focus of further analysis.

Analysis Phase: Existing Workflow

Once a candidate task was identified, we visited the manufacturing facility again to gather detailed information about the specific task. We filmed workers performing the task, to be analyzed later, and asked them questions to clarify the general assembly procedure and variants that are utilized by different workers. Video was collected only after receiving verbal consent from workers and was stored and shared with our collaborator through university-approved digital storage. No employee identifying information was collected through these videos (*i.e.*, verbal information or view of worker faces). In consultation with our Institutional Review Board (IRB), these interactions were not considered research with human participants, as they focused on the manufacturing processes rather than the individuals, although feedback sessions were, as described in §4.8. Additionally, administrators provided schematics for parts and workcell layouts utilized within the assembly process.

We used the collected data to reconstruct the assembly process to understand the steps involved in the procedure, and how it varied between workers. This process is shown in Figure 4.4. We first analyzed the video to create a timeline of the assembly process. Some segments included discussions with workers and additional pauses during which workers would provide the camera with specific views of the process or parts. These sections were removed from our timeline reconstruction for accuracy. Additionally, when appropriate, durations of specific steps were averaged between workers. Our reconstructed timeline can be seen in Table 4.1.

Once we reconstructed a timeline of the human-only process, we identified potential subtasks where the cobot could assist the operator in the pro-

Figure 4.4: A subset of the steps involved in the manufacturer's assembly process. A. applying silicone to gaskets. B. applying silicone to pans and beginning the assembly process. C. continuing the assembly process by attaching gaskets.

Table 4.1: A timeline of the SME's existing procedure to complete one cycle of their assembly process.

Time	Production Step
1:01	Place gaskets and apply silicone (61s)
2:28	Place pans and apply silicone (87s)
2:34	Apply core to first pan (6s)
2:43	Apply second pan to core (9s)
3:06	Clean excess silicone (23s)
3:57	Apply gasket 1 to core and secure with rivets (51s)
4:43	Apply gasket 2 to core and secure with rivets (46s)
5:21	Apply gasket 3 to core and secure with rivets (38s)
5:57	Clean excess silicone (36s)
6:34	Apply gasket 4 to core and secure with rivets (37s)
9:14	Clean pans, apply final bracket and labels, and move finished core (160s)

cess. These subtasks were: (1) the cobot applying silicone to components placed by the worker and (2) the cobot picking and placing components, and applying silicone to them. While we had noted a possible delineation of work at the end of the planning phase, this step helped us to formally identify and justify the distribution of subtasks. While both assignments

appeared feasible, it was unclear what the optimal selection would be given the differences in benefits to the worker and process, as well as the required component costs. Exploring these different allocations of work allowed us to provide our collaborator with multiple options to consider depending on their budget and business needs. Both options were presented and confirmed to be practical. Using these potential processes, we next created two new timelines to represent the potential human-cobot process accounting for the possibility of one or two workers being assisted by a single cobot simultaneously. These approximate time-tables are used as the basis for the development phase, as they provide a rough outline of what needs to be achieved and in what time frame.

Development Phase: Simulation

Based on the approximate timetables that were created in the analysis phase, we created simulations of the new human-cobot workflows using the CoFrame (Schoen et al., 2022) system. Within the simulation, we initially modeled the workcell of the manufacturer based on the information they provided and adjusted it to reflect the changes brought by integrating a cobot into the space. The created models included walls to visualize spatial constraints, tables where the component preparation and assembly process occurs, a table on which a tool switcher would be placed, and a conveyor to symbolize a location for component pickup (see Figure 4.5). After modeling the environment, we added models for the cobot, a UR5e robot arm, models for the components that are required within the assembly process of a single product unit (*i.e.*, a core, pans, and gaskets), and defined regions of the workspace that the operator would work in. With the work environment, cobot, and components modeled, we next began to create simulations of the cobot and worker tasks.

Process 1: Silicone Application Only

The first process we created in simulation involved the worker placing components, the cobot applying silicone to those components, and the worker completing the remaining steps in the assembly process. A simplified version of the final timeline for this workflow can be seen in Table 4.2, and reflects the cobot's ability to assist two workers in parallel. Within our simulation, we defined locations for components to be placed and waypoints for the cobot's end effector to follow as it applied silicone. The cobot's end-effector speed was optimized for efficiency and safety based on feedback from CoFrame's review panel. The simulation also included processes to simulate the worker assembling components to recreate an entire production cycle and produce performance metrics that accurately captured the entire human-cobot workflow. These performance metrics

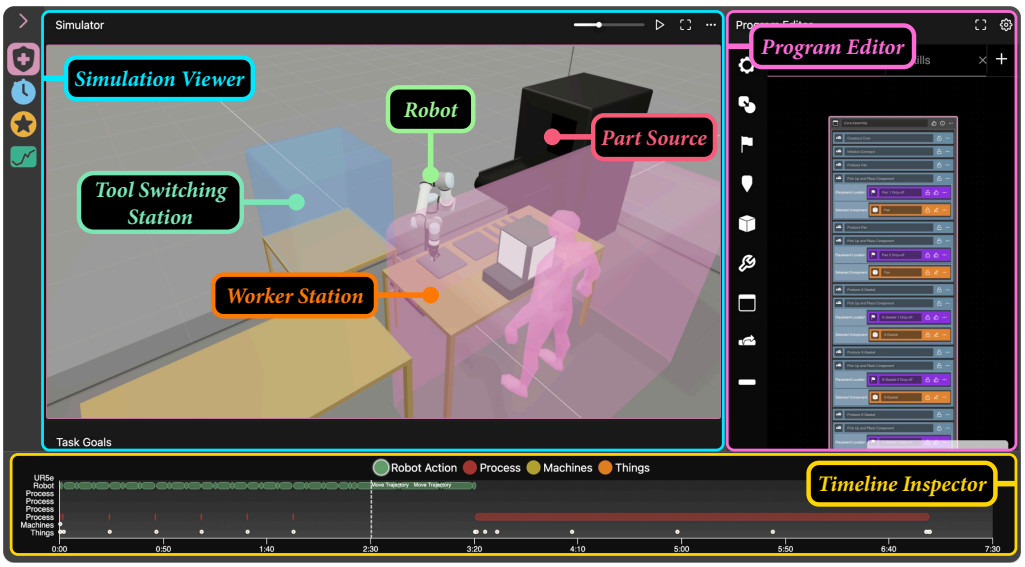

Figure 4.5: The program and simulated environment created for process 2. The environment captures the robot, a workspace for the operator, a workstation, a location to switch end effectors, and a source for component parts.

included cycle time, cobot idle time, and wear-and-tear cost.

Process 2: Silicone and Pick-and-Place Workflow

Our second simulated workflow involved the cobot picking up, placing, and applying silicone to components, and the worker completing the remainder of the assembly process. Similar to our first simulation, locations and waypoints were defined for components and the cobot's end-effector path. However, in this simulation, after placing components, the cobot navigates to a designated tool switcher zone to exchange its gripper for a silicone dispenser. The cobot's end-effector speed was calibrated based on feedback from CoFrame and the performance metrics that it generated. While this workflow offloads the additional task of placing components from the worker to the cobot, and resultingly reduces cobot idle time, this process still allows the cobot to assist two workers in parallel. Table 4.3 shows the timeline for this workflow, and Figure 4.5 illustrates the simulation setup.

Table 4.2: A timeline of process 1 including the steps and timing for a cobot to assist one or two workers in applying silicone.

<table>
<tr><th>Time</th><th>Cobot</th><th>Worker 1</th><th>Worker 2</th></tr>
<tr><td>0:17</td><td>Idle (17s)</td><td>(Cycle Start) Place pans and gaskets (17s)</td><td rowspan="2">Assemble pans, gaskets, and core (58s)</td></tr>
<tr><td>0:58</td><td rowspan="2">Apply silicone to pans and gaskets for worker 1 (58s)</td><td rowspan="3">(Previous Cycle End) Finalize core construction (196s)</td></tr>
<tr><td>1:15</td><td>(Cycle Start) Place pans and gaskets (17s)</td></tr>
<tr><td>2:13</td><td>Apply silicone to pans and gaskets for worker 2 (58s)</td><td rowspan="3">(Previous Cycle End) Finalize core construction (196s)</td></tr>
<tr><td>3:33</td><td rowspan="3">Idle (298s)</td><td rowspan="2">Assemble pans, gaskets, and core (218s)</td></tr>
<tr><td>4:31</td></tr>
<tr><td>7:11</td><td>Assemble pans, gaskets, and core (160s)</td></tr>
</table>

Table 4.3: A timeline of process 2 including the steps and timing for a cobot to assist one or two workers in gathering components and applying silicone.

<table>
<tr><th>Time</th><th>Cobot</th><th>Worker 1</th><th>Worker 2</th></tr>
<tr><td>2:10</td><td>Place pans and gaskets for worker 1 (130s)</td><td rowspan="4">Finalize core construction (196s)</td><td rowspan="4">Continue assembly of pans, gaskets, and core (203s)</td></tr>
<tr><td>2:19</td><td>Switch from gripper to dispenser for worker 1 (9s)</td></tr>
<tr><td>3:15</td><td>Apply silicone to pans and gaskets for worker 1 (56s)</td></tr>
<tr><td>3:16</td><td rowspan="3">Switch from dispenser to gripper for worker 2 (9s)</td></tr>
<tr><td>3:23</td><td rowspan="8">(Cycle Start) Assemble pans, gaskets, and core (218s)</td><td rowspan="4">Finalize core construction (196s)</td></tr>
<tr><td>3:24</td></tr>
<tr><td>5:34</td><td>Place pans and gaskets for worker 2 (130s)</td></tr>
<tr><td>5:43</td><td>Switch from gripper to dispenser for worker 2 (9s)</td></tr>
<tr><td>6:39</td><td>Apply silicone to pans and gaskets for worker 1 (56s)</td><td rowspan="3">(Cycle Start) Assemble pans, gaskets, and core (15s)</td></tr>
<tr><td>6:48</td><td>Switch from dispenser to gripper for worker 1 (9s)</td></tr>
<tr><td>6:54</td><td>Idle (6s)</td></tr>
</table>

Equipment Costs

As the final step of this phase, we obtained cost estimates for the cobot and other materials required for integration based on our simulations. These estimates helped to develop requirements to reconstruct the simulations within the manufacturer's facility. Cost estimates were obtained through discussions with a robotics vendor. While not included within our simulations, we also obtained cost information for a range extender that would allow the cobot to move between two workstations. If the range extender were to be utilized, the manufacturer could use a single cobot to

assist two workers in parallel with minimal changes to their existing setup. Alternatively, the same benefits may be achieved by reconfiguring the workspace such that a single cobot could directly access two workspaces. These hardware options were included to allow for greater flexibility in achieving preferred outcomes and allowed us to present multiple options to the manufacturer so they could select one based on their needs and constraints.

Presentation Phase: Reporting Results

Using the performance metrics that were generated from our simulations and the cost information that we obtained for each workflow, we created a complete write-up to describe both processes, their associated costs, and the trade-offs of each. We presented these findings to the manufacturer, describing in detail each plan, showing them the simulations and output metrics of each, and answering any remaining questions they had.

Synthesizing Results

Based on our analysis of the original human-only workflow, the total cycle time was 9:14 minutes. By comparison, our simulation of processes 1 and 2 yielded cycle times of 7:11 minutes and 6:54 minutes, respectively. As a result, processes 1 and 2 would reduce total cycle time by approximately 22% and 25% respectively. However, the manufacturer would need to weigh the improvement these reduced cycle times provide against the total costs of integration. For process 1, our estimated integration cost was $38,470.00 USD, but resulted in more idle time by the cobot. Process 2's increased task assignment reduced this idle time, but required additional hardware and would cost an estimated $47,350.00 USD to be integrated into the facility. Additionally, both workflows would incur some level of wear-and-tear cost from regular operation. According to our simulation,

Table 4.4: This table shows several metrics comparing both proposed processes and their direct differences.

	Cycle Time in Seconds	Time Reduction	Robot Idle Time in Seconds	Wear and Tear Cost	Component Cost
Process 1	431.2	22.17%	372.8	Negligible	$38,470.00
Process 2	414	25.27%	219.6	$0.02	$47,350.00
Difference	17.2	3.10%	153.2	$0.02	$8,880.00

this cost would be negligible for process 1 but would be $0.02 USD per cycle for process 2. Both workflows produced clear benefits from a business perspective, but process 2 was able to fulfill worker preferences by offloading an undesirable task (*i.e.*, applying silicone to components) to the cobot, although at a much higher cost (shown in Table 4.4 and Figure 4.6).

Presenting Results

During our final meeting with our collaborator, we presented all the results uncovered throughout our collaboration. We reviewed the overall steps we had taken over the span of our several-month partnership, focusing on the factors that motivated the need for cobot intervention within our collaborator's facility and highlighting the decisions made along the way to progress toward the final proposals. We discussed the process required to create our simulations, including the simulation environment (*i.e.*, CoFrame), our model of the manufacturer's workspace, and the cobot we utilized within the simulations. This overview helped to familiarize our collaborator with the simulations before providing a full demonstration. Next, we provided a detailed demonstration of each workflow simulation along with simplified timelines to illustrate each process. When showing process 2, we provided an overlay video of the human work process to clearly visualize the worker's and cobot's coordinated effort. These

demonstrations provided an intuitive and compelling view of the cobot's effectiveness. For each of the simulations, we discussed the required integration costs and the performance metrics they each achieved. While the ultimate decision of whether and how to integrate a cobot belonged to our collaborator, we attempted to provide a clear understanding of the benefits of each workflow and how they compared to one another. Following our presentation, our collaborator had several questions regarding potential next steps and practical considerations if they were to pursue either option further. We addressed these questions and offered further support, should the need arise.

Feedback Session

Following our final presentation with our SME collaborator, we held one additional discussion session to receive feedback on our presented work and four-phase approach. Within this session, we sought feedback from two manufacturing engineers and a manufacturing engineer manager. While we expressed interest in involving production workers in this session, we were unable to due to their work schedules. At the beginning of the session, consent forms were read and signed by all participants. During the session, we briefly presented our four-phase approach and conducted a semi-structured interview with the participants as a group. The session was conducted over Zoom and recorded. We transcribed the interview and conducted a thematic analysis of the interview data, which revealed three primary findings discussed below.

First, we learned that our approach was able to capture worker needs at all levels. The manager emphasized that production workers would be very pleased if an undesirable task (*i.e.*, messy silicone caulk application) were offloaded to a cobot partner. He further emphasized that the use of cobots in this process would not eliminate the worker, but rather make their work more efficient. Additionally, the engineers reported that their

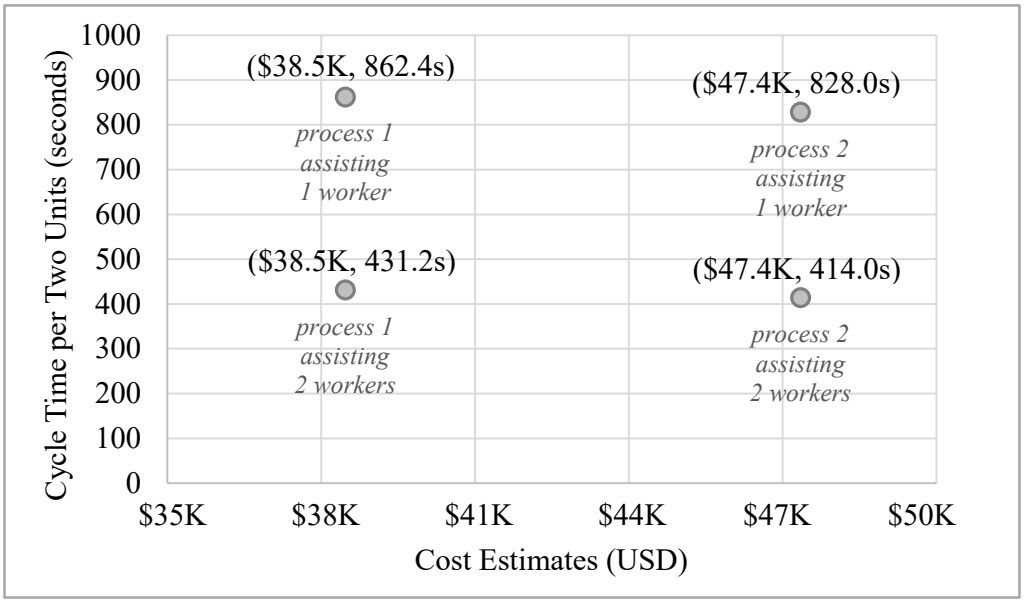

Figure 4.6: This plot showcases the costs of our processes and their cycle times to assemble two units when one or two workers are assisted by the cobot.

needs were met by providing relevant information including robot motion sequences, spatial constraints, and comparisons to the currently manual process. Finally, the manager reported that our collaboration answered many of their questions about cobot application.

Second, our work acted as an effective proof of concept for our collaborator. The manager expected that work toward cobot integration would occur in phases and that the first phase would be determining feasibility. The manager stated, *"I think this project really helped us answer the question, 'can we do this?', right? 'Is this a good application?' I think that really helped us."* Specifically, our collaborators expressed that the simulations illustrated the "big picture" of what such a workflow could resemble and allowed for a clear comparison to the existing processes. These sentiments convey how our proposed process can answer initial critical questions that may be stumbling block for many businesses in evaluating whether cobot

integration is appropriate in their facilities.

Finally, we learned that, although our case study provided our collaborator with an understanding of feasibility toward making informed decisions, implementing our presented solutions would require additional action. Our collaborators explained that management approval for such a project would require information including comparisons across available cobot systems, observation of these systems in actual production, and generation of supplier lists. Then, resulting elements such as exact workcell layouts, equipment specifications, and cost estimates can be proposed for budget approval. Although some of these implementation steps extend beyond what our approach encapsulates, they can clearly benefit from and build upon the results generated utilizing this approach.

4.9 Discussion

Cobots have a high potential for application within manufacturing settings (Liu et al., 2022a), but designing appropriate and effective applications requires considerable knowledge held by only a few individuals. This makes utilizing cobots for collaborative tasks difficult or unattainable for most. This presents numerous opportunities for researchers and system developers to explore ways of supporting user understanding and increasing cobot accessibility.

The results of our lab study highlight the challenges in industry, as discussed by experts, illustrating an overarching need to make cobot technology more accessible to individuals as a means to help bridge the skills gap. While prior work has begun to explore ways of making physical robots accessible to the community, such as through incorporating cobots in makerspaces (Ionescu and Schlund, 2019), our work demonstrates the potential for increasing accessibility using simulated robots in a structured programming environment. This reduces the barrier to accessibility, only

requiring an internet connection, as users can work with simulated robots while leveraging CoFrame's abstractions of cobot concepts to build an understanding of collaborative principles.

Additionally, both expert and novice users saw the benefit of these abstractions, developed through our translation of the *Safety First Expert Model* (Siebert-Evenstone et al., 2021) into interactive textual and visual program feedback, as a means of supporting user knowledge and producing effective collaborations. While users had varying expectations and desires for the behavior of CoFrame's features, all liked the conceptualization of the explicit program feedback and issue detectors, along with the extensive coupling of the visual and textual information to support the creation of cobot programs.

Through our real-world deployment of CoFrame in our collaboration with an SME manufacturer, we learned that users are not all at the "programming" stage for cobot usage. Users need additional support to feel confident in understanding cobot tasks and capabilities prior to exploring the task through programming, necessitating additional support in processes leading up to programming. We presented our approach to supporting users and demonstrated the steps required to apply our proposed approach, develop new human-cobot workflows, and assist a business in making an informed decision regarding cobot integration. This experience offered critical insights into the information that a business needs to make informed decisions about integration and what roboticists require to assist that business. These insights show that our approach can begin to answer our initial questions regarding complementary cobot intervention, worker preferences, and successful cobot integration. First, our approach can offer a deep understanding of the manufacturer's workflow and create solutions that utilize cobots' collaborative capabilities. Further, our approach can enable roboticists to apply principles of collaborative robotics, ergonomics, and HRI to develop effective workflows that meet worker

preferences. Finally, our approach can help satisfy the needs of the larger business without sacrificing the prior two goals. This intersection of proper cobot utilization, consideration of worker preferences, and optimization of business goals is critical to cobot integration.

While our case study illustrates the utility of our proposed approach, certain elements warrant further exploration. First, our proposed human-cobot processes were developed only in simulation. Given that simulations cannot accurately reflect real-world performance, further *in situ* evaluations will likely produce additional insights into the effectiveness of our approach. Additionally, while our case study proceeded in a relatively linear fashion, further evaluation of our approach's iterative capacity (*i.e.*, the ability to iterate between phases rather than only within individual phases) may help us to understand its ability to handle complex manufacturing processes. Furthermore, to make the integration process more accessible for SMEs, future work can seek to develop tools and frameworks that support SMEs in their endeavor to make informed decisions about cobot integration.

Design Implications

Based on our results, we present several design implications for the development of future expert feedback programming systems.

Design Implication 1 *Supports should be designed for collaborative use.* Both expert and novice users benefited from and valued the supports and visualizations incorporated within *CoFrame*. Our results illustrate the current difficulties and challenges in cobot usage faced by experts, with teaming being one method used for producing and verifying the safety of cobot programs. *CoFrame*'s use of visualization, structured feedback, and highlighting of problematic areas of code can act as a collaborator for users, performing the role of a teammate to provide verification. This was evi-

dent in the way novices engaged with the support, but also in the expert's discussion of the way the support directed them to issues they would already be looking for. Future systems should support users by designing supports that can be used collaboratively, allowing users to request feedback that draws their attention to issues and support problem-solving, such as with visual cues, contextual prompts, step-by-step guidance, or interactive explanations.

Design Implication 2 *Support diverse users and workflows.* Users bring different backgrounds to the cobot programming process, spanning a range of goals, experiences, and expectations. This results in a variance of preferences for how scaffolding and abstraction supports should behave. Our results show how users were split over when to incorporate feedback, or how to best inform them of issues and changes to their program. Additionally, users envisioned multiple use cases of *CoFrame* outside of just industrial programming, from being used to increase accessibility in education through web-based programming and simulation, to supporting sales and customer understanding by demonstrating prototype programs. These differences in users and use cases reflect the need to support diverse workflows for user engagement in creating cobot programs and addressing program feedback. By offering multiple workflows for addressing the programming process, future systems can allow users to tailor the experience to their workflows, supporting users in the development of programs, troubleshooting problems, and progressively deepening their understanding of cobots over time.

Design Implication 3 *Support knowledge progression and exploration.* While high-level abstractions can get users initially engaged with and remain oriented within a task, users who are more experienced or curious seek to understand the underlying mechanisms for cobot operation. Our study revealed that participants began their interactions with the high-level

abstractions provided in the system, but grew curious about the lower-level representations for interacting with cobots; often desiring advanced features to understand the state of the cobot. To allow for the progression of exploration and understanding, future systems should support user exploration of lower-level abstractions that map the high-level interface to low-level concepts. This can occur by allowing users access to lower-level programming implementations, as discussed by users in this study, but can also occur by removing the simplifications and abstractions that the system provides, such as grasp simplification. While real-world programs will need to account for grasp detection, friction, gravity, etc., these factors are not present in *CoFrame* to allow users to focus on conceptual problem-solving of the task. Future systems should support multiple levels of fidelity, allowing users to toggle the level of abstraction for these supports and progressively engage with more complex representations and real-world concerns. By supporting knowledge progression and exploration in this way, future programming systems can promote deeper learning and more sophisticated collaborations with cobots.

Limitations & Future Work

While our lab study serves as an initial exploration of novice and expert user perceptions, one limitation of this study is the relatively small number of expert participants. Recruiting domain experts is a well-known challenge, but was further restricted due to our target demographic of cobot integrators. Future research should aim to broaden the participant pool by incorporating a more diverse range of expert stakeholders, including instructors and industry professionals from multiple domains.

Additionally, our real-world deployment highlights the difficulties for industry in utilizing cobot tools. Future research should more closely work with industry to understand users and explore the development of tools in real-world contexts. This will allow for more rapid identification of

issues and concerns, allowing for new tools to better support users in the planning and programming of cobot interactions.

4.10 Chapter Summary

As industries confront a growing skills gap, there is a clear need for new systems that simplify the use of cobots. While cobots hold great potential for use within SME manufacturing facilities, SMEs may not have the required knowledge to fully utilize the collaborative capabilities of cobots or understand the implications of their integration. Effective integration and usage of cobots requires consideration of human worker needs and preferences, proper utilization of cobot strengths, and improved performance of manufacturing processes. Through the results of the two lab studies, our evaluation of CoFrame demonstrates the potential for these techniques within cobot programming, highlighting where participants enjoyed support from several aspects of the software. However, it also illustrates a variance in expectations and needs across users, highlighting the differences in their backgrounds, both in terms of knowledge and capability.

This is further exemplified through our real-world deployment, where CoFrame was used to help develop the resulting program but required additional external support structures to effectively plan for cobot usage and integration. Integrating cobots into existing manufacturing workflows requires extensive knowledge of cobots, their capabilities, human-robot interaction principles, and a deep understanding of the manufacturer's needs. While the design of CoFrame, presented in Chapter 3, demonstrates how the design of programming systems can support users through abstraction and scaffolding of robotics knowledge, this chapter illustrates the need for additional user support earlier in the process. Specifically, users require additional support in planning effective cobot interactions before

reaching the programming stage. Our approach facilitates this deep understanding, assists in the development of effective integration proposals, and supports SME manufacturers in making informed decisions about cobot integration within their facilities.

This chapter presents an empirical evaluation of the CoFrame system through lab studies and real-world deployment. It highlights user perceptions and use of the system's supports for building cobot programs, building on the understanding of the design impacts from Chapter 3. Additionally, it presents a theoretical framework for working with SME users to facilitate additional cobot understanding that is lacking within the design of the system. This highlights additional opportunities for developing support for user creation of cobot interactions.

5 ALLOCOBOT: DESIGNING PLANNING SYSTEMS FOR COLLABORATIVE INTERACTIONS

This chapter presents the design and implementation of the *Allocobot* human-robot collaboration task planning system. It addresses the challenge of supporting user planning of human-robot collaboration, leveraging user expertise while abstracting critical robotics, human factors, and economics knowledge. In this chapter, we discuss the motivation of the work, review relevant work to understand the design and application of the system, detail our representation of collaborative tasks and the implementation for learning effective collaborations, and reflect on the design of the system through an illustrative workflow. This chapter includes work from a manuscript in progress (White et al., 2025a), building on work presented by Schoen (2023).

5.1 Motivation

Recent trends in research illustrate an increasing focus on human-robot collaboration within the concepts of Industry 5.0 (Nahavandi, 2019). Industry 5.0 focuses on collaboration as a means to increase productivity, personalize products, and create flexible work cells, while enabling and empowering human workers through the use of robots. Collaborative robots, or "cobots," are one of the technologies well suited to address this need, due to their safety focus and capability to assist human workers in shared collaborative workspaces as full coworkers (Fanuc; Kuka; Universal Robots). Cobot market share continues to grow (Grand View Research, 2023), as manufacturers begin to explore how to effectively utilize cobots within their processes.

However, deploying cobots in real-world scenarios has proven to be a difficult task for manufacturers. This has resulted in a growing skills gap in

industry, as many manufacturers struggle to realize the benefits of human-robot collaboration, often focusing on using cobots as cheap forms of automation or for simple start-and-stop based interactions (Michaelis et al., 2020). Several attempts have been made to bridge the gap, often focusing on programming systems (Schoen et al., 2022; Emeric et al., 2020) and increasing cobot accessibility within makerspaces (Ionescu and Schlund, 2019; Ionescu, 2020).

However, recent work has demonstrated the need for additional support for manufacturers to understand and make informed decisions when creating collaborative interactions (Sullivan et al., 2024). This suggests that support is needed not only for programming cobots but also in the earlier stages when planning tasks and workflow allocation. Planning for human-robot collaboration is particularly challenging for users due to the extensive multidisciplinary knowledge required across economics, ergonomics, human factors, robotics, and human-robot interaction to successfully create viable collaborations. For instance, recent work has shown that when creating collaborative tasks, the characteristics of the job affect the degree to which the cobot will impact the ergonomics of the human collaborator (Liu et al., 2022b). However, successful collaborations require an in-depth understanding of all these concepts, balancing the economic concerns, such as budget and implementation cost, with ergonomic factors, such as human worker impact, to determine what collaborations should look like.

One method to address this gap is the creation of new systems that abstract this multidisciplinary knowledge while supporting user creation of human-robot collaborations. In this chapter, we present an update to the design and implementation of the *Allocobot* system initially presented by Schoen (2023), illustrating how it creates viable interaction plans through our design choices in data representation and interaction simulation. The result of the *Allocobot* system is a policy that accounts for all collaborator

decisions, allowing for flexibility in task executions needed in real-world deployments.

The contributions described in this chapter include[1]:

1. An update to the design and implementation of the *Allocobot* human-robot collaboration planning system (see Appendix A for a link to the system).

2. The presentation and discussion of a workflow that demonstrates the behavior of the system and its support for end-users building human-robot collaborative plans.

5.2 Related Work

As more manufacturers begin to integrate collaborative robots into their workflows, there is a growing interest in supporting the task allocation process. However, creating collaborative interactions remains challenging as users must find appropriate tasks that align with cobot capabilities (Kadir et al., 2018), and assess how best to allocate work between robot and human agents. A significant body of prior work has explored the use of optimization-based methods that seek to allocate work between human and robot agents based on a variety of metrics (Huang et al., 2023; Calzavara et al., 2023; Monguzzi et al., 2022; Battini et al., 2016). One example is by Pearce et al. (2018), who combine metrics on ergonomics and productivity to create human-robot interactions that minimize the total task time and impact on the human worker.

When optimizing task allocations, ergonomics is a common objective for analyzing tasks and mitigating risk factors for human workers (Potvin,

[1]The research in this chapter is derived from a manuscript in progress by myself, Dr. Andrew Schoen, Dr. Anna Konstant, Dr. Josiah Hanna, Dr. Robert Radwin, and Dr. Bilge Multu, and presents an update to the design and implementation initially presented by Dr. Andrew Schoen.

2012). Common metrics for assessing task impact on human workers include the Strain Index (Garg et al., 2007, 2017), the Metabolic Prediction Model (Garg, 1976), and the Revised NIOSH Lifting Equation (Waters et al., 1994). These metrics are often applied holistically, evaluating components in the context of the entire task, and are used to allocate strenuous work away from humans and onto robot agents, reducing overall risk (Liao et al., 2023; Dalle Mura and Dini, 2022). Additionally, many approaches also consider task productivity, as literature has identified trade-offs between ergonomics improvements and task productivity (Pearce et al., 2018; Wang and Li, 2019; Huang et al., 2021; Raatz et al., 2020).

However, several challenges still limit the broader usage of cobots, such as how to have the cobot operate safely within the environment (Kildal et al., 2018; Malm et al., 2019), and how robots should interact with individuals (Galin and Meshcheryakov, 2020; Khalid et al., 2017). While some researchers have explored parallel task optimization (Pearce et al., 2018) where humans and robots have minimal task overlap, others have investigated collaborative tasks where the human and robot agents need to interact on a task together (Wang and Li, 2019).

Despite this, many existing methods produce statically scheduled allocations of work, assigning a fixed set of tasks to agents (Kheirabadi et al., 2023). In contrast, real-world interactions are often more complex, requiring an awareness of both collaborators and the task state (Bi et al., 2021) to inform decision-making, resulting in dynamic workflows. Effective collaborative interactions require that agents adapt to changing environments and conditions (El Zaatari et al., 2019). Failure to consider these factors can create uncollaborative environments, as collaborative environments are dictated in part by the nature of the task and are not inherent to the application of cobots (Guertler et al., 2023; Liu et al., 2022b).

One way of representing all these factors is through Petri Nets, a directed bipartite graph consisting of two node types (*Places* and *Transitions*)

(Peterson, 1977). Petri Nets have been shown to be an effective tool for the modeling and analysis of workflows (Van der Aalst, 1998; Sun and Jiang, 2009; Adam et al., 1998), with variants of the Petri Net implementation being used to represent time (Zuberek, 1991) and resources (Souravlas et al., 2020). This makes Petri Nets well suited for collaborative tasks, due to their capability to model both parallel and concurrent actions (Casalino et al., 2019; Hu and Chen, 2017; Ziparo et al., 2011) as well as being able to represent the state of processes (Zurawski and Zhou, 1994).

The work presented in this chapter builds on this understanding, combining factors from multiple disciplines with Petri Nets to learn effective human-robot task allocations.

5.3 Approach

The original design of *Allocobot*, as described in Schoen (2023), is centered around enabling flexible and effective collaboration between human and robot workers. It accomplishes this through a hierarchical task representation, decomposing jobs into several components, consisting of agents, tasks, targets, and environmental information. A complete breakdown from the initial implementation can be seen in Figure 5.1.

For each task, *Allocobot*uses an action primitive representation to define the actions necessary for agents to perform. The actions describe how agents interact with objects, move about the environment, and define the impact of the action on the agent. These primitives are designed to be additive to build more complex actions and are compatible with both independent and joint activities to allow for shared task representations.

Allocobot supports both robot and human agents, each defined by a unique set of capabilities and constraints. When analyzing tasks, *Allocobot* examines the set of primitives and determines whether each agent is capable of accomplishing the task. Similarly, it looks at the pairing of

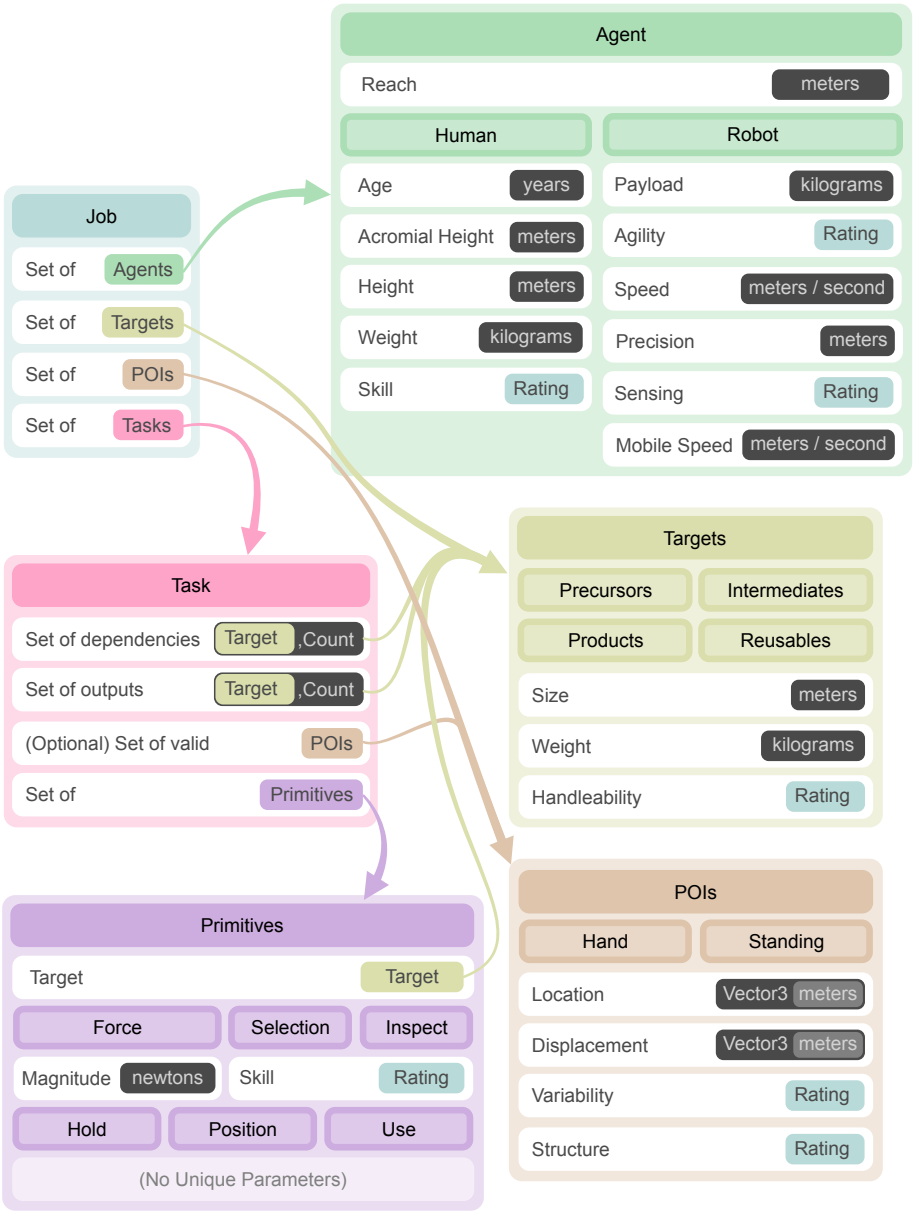

Figure 5.1: A breakdown of *Allocobot*'s job representation. Each card represents core components defining a job, and includes Agents to be utilized in the job, a set of Tasks or steps to complete the job, Targets to be manipulated throughout the job, Points of Interest (POIs) that represent the job environment, and Primitives that define the actions for each Task. Figure from Schoen (2023).

agents to determine if tasks can be split, allowing for subsets of primitives to be completed by each agent.

Tasks in *Allocobot* operate on a set of targets, which describe the physically interactable objects within the environment. These include reusable components, such as tools, precursors, intermediates, and products, which describe input, intermediate, and output parts.

Allocobot also encodes environmental information through points of interest (POIs). POIs include both standing and hand locations, representing locations where agents will stand to perform work activities and where the activity occurs. These locations are utilized to determine whether agents need to move between locations to reach various objects.

Overall, this hierarchical breakdown enables *Allocobot* to track the position and state of the agents and targets within the environment over time. This allows *Allocobot* to understand task progression and collaboration characteristics by analyzing the state of these components. This is enhanced by the primitive and point of interest (POI) representation, allowing *Allocobot* to understand how the collaboration impacts agents and understand agent limitations such as reachability or movement.

To support flexible collaboration, *Allocobot* encodes this information using a Petri Net representation. This allows the system to represent the partial orderings of tasks and dynamic execution paths. This approach is useful for the modeling of concurrent and interdependent tasks, allowing for the simulation of collaborative tasks by adjusting the state of the Petri Net. This simulation is used by the reinforcement learning approach to find a policy that represents a human-robot collaboration. The resulting policy represents the optimal interaction as well as non-optimal variations, where agents may deviate from the optimal policy, encoding multiple options for continuing collaborations in non-optimal states. This flexibility is crucial for real-world human-robot teaming.

Update to Original Design

In this section, we describe the number of ways we build on the initial design of *Allocobot*.

As *Allocobot* pulls from an understanding of human factors and ergonomics, economics, robotics, and human-robot interaction, several factors need to be represented within jobs to allow users to add and adjust parameters that affect the factor prioritization and output collaboration of the job. Primarily, jobs in *Allocobot* have been extended to include several user-specified parameters that can affect the produced collaboration, including monetary information such as worker hiring and onboarding costs, robot purchasing and setup costs, and the cost of electricity in kilowatt hours. Additionally, users can specify an α value that represents the relative weighting of the monetary and ergonomic factors *Allocobot* considers in producing collaborations. The addition of this information represents *Allocobot*'s ability to account for multidisciplinary factors while allowing users to specify prioritization, which impacts output collaborations.

In the original specification of primitives, the *Position* primitive was restricted to just the turning or rotating of a target. This restriction resulted in *Allocobot* assuming that agents do no *Move* objects within the environment, meaning that certain real-world actions that require users to move their hands about the environment, such as wiping a surface or guiding a target into a keyed or fitted position, could not easily be modeled. In the updated design, a displacement factor has been added to the *Position* primitive. This differs from the *Move* primitive in the restriction that the displacement factor assumes the agent's hand ends up in the same location it began.

In the original design, *Allocobot* did not specify the starting or goal location of targets. This resulted in the system allowing for precursor targets to be created and used at any location, without any restrictions on carrying them around the environment. This resulted in *Allocobot* having

agents move to locations to create and use the part there, without needing to carry the parts around the environment to be used in other locations. To better represent real-world interactions and constraints, *Allocobot* has been updated to include additional representation for all targets and tasks that restrict their locations. Now, when precursor targets are placed within the environment, *Allocobot* has a representation of the starting location of that precursor object, requiring agents to move to the location to carry the target to new locations. Tasks can similarly be restricted for specific locations within the environment, better representing real-world workflows where machines or processes may be tied to specific locations. With this representation, the placement of targets within the environment will either disable or allow specific tasks at each POI. For example, placing targets away from a POI rather than near it will mean that *Agents* must first carry targets to the POI to use them at the POI to complete a task. Additionally, this representation allows for the specification of product targets, allowing users to distinguish between locations where actions occur and where products are moved to be shipped or used in order jobs.

Previously, *Allocobot* required users to specify parameters dictating human capability, such as various heights and weights. This required users to have in-depth knowledge of the relationships of multiple factors for human capability, which could result in changes to the collaboration. In the updated design, *Allocobot* asks users to specify a population percentile and gender they are targeting for the human agent. This means that a user is able to easily specify a task such that 99% of the population can complete the task, which is easier for them to understand rather than how the specific heights and weights map onto human capability.

Finally, the original design of *Allocobot* did not model robot capability. In our attempt to map robot capability to the primitives defined, we were unable to find suitable prior work that quantifies robot capability or error rates. As part of the updated design, an initial implementation for this

capability has been modeled based on our expertise and understanding of cobots. This implementation includes factors based on the robot's sensing and movement capabilities as defined by the user.

5.4 Implementation

As described in Schoen et al. (2022), *Allocobot* utilizes a Timed-Transition Petri Net to represent the execution and ordering of tasks as well as the location of agents and targets within the environment. The Petri Net model utilizes places, transitions, and arcs to dictate the movement of tokens through the system. These tokens represent the targets and agents, with places representing the state of the task and the location of the environment. Transitions connect to places through arcs, determining how long actions take before moving tokens from the set of input places to the set of output places within the Petri Net. Using this Petri Net representation, we leverage techniques in reinforcement learning to simulate and learn the sequence of transitions necessary to facilitate effective task completion.

In the remainder of this section, we describe the updates to the original *Allocobot* implementation, with particular attention to the implementation of agents, cost estimation, and our approach to reinforcement learning.

Cost Modeling

Allocobot explicitly models both ergonomic and economic costs for each transition in the Petri Net. Costs are split into two categories: onetime and extrapolated. onetime costs represent the costs that occur exactly once per job and are usually related to the setup of the job and environment, such as the purchasing cost of the robot or the hiring and onboarding cost of the human worker. extrapolated costs are used to represent the costs that recur each time the job is run, such as worker compensation, cost of robot task failure, and ergonomic impact to human workers. These costs are

used by the reinforcement learning approach, described in Section 5.4, to determine optimal interaction policy.

Economic Costs

Allocobot requires the user to specify the costs associated with several portions of the job setup process. These costs include onetime estimated costs for hiring and training human agents or the purchasing price of the robot agent. Additionally, this includes extrapolated costs such as salary costs for the human agent, the cost per kilowatt hour of electricity, the cost of *Precursor* parts for the process, and finally the value of the *Product* parts produced by the job. While not directly a cost variable, *Allocobot* requires the user to specify the level of sensing capability for the robot agent, which is used to estimate the cost of configuring the work cell. We use Zane Michael's estimate of between four and six times the cost of the robot (Maw, 2018) for the cost of a custom work cell, with four times the cost representing a low level of sensing capability, and six times the cost for a high level.

Ergonomic Cost Modeling

The ergonomic cost is a measure of the ergonomic risk to the human worker when performing different actions. Two common factors affecting ergonomic risk are force and duration of exertion. Many ergonomic tools utilize force and time, such as the Strain Index (SI), Revised Strain Index (RSI), Garg Metabolic Prediction Model, NIOSH Lifting Index, Rohmert's Law, and the Hand, Shoulder, and Lifting Threshold Limit Values (TLVs) Garg et al., 2017, 1978; Potvin, 2012; THOMAS R. WATERS and FINE, 1993; of Governmental Industrial Hygienist), 2002. Therefore, we modeled ergonomic risk by multiplying the normalized force in the action and the duration of the exertion (see 5.1). In our equation, we calculate the

normalized force to be the required force for the action relative to the person's strength, commonly known as a percentage of the maximum voluntary contraction (%MVC) (Basmajian and Luca, 1985).

$$\text{Extrapolated Cost} = \%MVC \times \text{Duration of Exertion} \tag{5.1}$$

Additionally, the user must input information about the worker, such as biological sex and percentile of the population, so that *Allocobot* can compute the maximum strength and capability of the worker in a specific task. These maximum capabilities are then used for each action to calculate the maximum voluntary contraction (see 5.2) to determine whether the human worker is capable of performing a given task and what the impact of that task is on them. Further details of the ergonomic modeling can be found in Konstant (2024).

$$\%MVC = \frac{\text{Required Force}}{\text{Strength of Agent}} \tag{5.2}$$

Productivity Modeling

Methods-Time Measurement (MTM) by Barnes (1980) was used to model the time in Allocobot. MTM is a predetermined time system tool used to analyze the basic motions in a job, task, or subtask. MTM-1 consists of 9 basic elements: reach, move, turn, apply pressure, grasp, position, release, disengage, eye times (focus and travel), and body, leg, and foot motions. The primitives are defined in Schoen (2023) and are mapped to the MTM-1 motions. *Allocobot* requires that for each task of a job, the user must specify the set of primitives associated with completing the task. This allows *Allocobot* to compute the total time needed for each task to be completed. Further details of the productivity modeling can be found in Konstant (2024).

Table 5.1: Robot error rate for physical interaction tasks, defined as a function of sensing capability, and both location structure and variability.

		Robot Sensing		
Structure	Variability	Low	Medium	High
Low	Low	3.00E-01	1.50E-02	4.00E-03
	Medium	7.00E-01	6.00E-02	7.50E-03
	High	9.50E-01	1.00E-01	1.00E-02
Medium	Low	1.50E-01	1.00E-02	2.50E-03
	Medium	4.00E-01	2.00E-02	5.00E-03
	High	8.00E-01	6.00E-02	7.50E-03
High	Low	1.00E-02	4.00E-03	1.00E-03
	Medium	2.00E-01	1.50E-02	3.00E-03
	High	5.00E-01	4.00E-02	6.00E-03

Robot Cost Modeling

In our review of prior work, we were unable to find work that quantifies robot capability or error rates. *Allocobot* relies on these factors to determine the capability and costs of assigning work to each agent. We created an initial set of error rates based on our experience and expertise with cobots. For our modeling of the robot's cost, we consider two scenarios for error. The first is in the physical interaction with objects, such as by applying force or the manipulation of objects, and is modeling as a function of the robot's sensing capability and the variability and structure of the location (Table 5.1). The second form of error comes from visually demanding tasks, such as part inspection or selection. This form of error is modeled as a function of the robot's sensing capability and the skill rating for the associated action (Table 5.2).

Table 5.2: Robot error rate for visual inspection tasks, defined as a function of sensing capability and skill rating.

		Robot Sensing		
		Low	Medium	High
Skill Required	Low	5.00E-03	2.00E-03	5.00E-04
	Medium	1.50E-02	6.00E-03	7.50E-04
	High	1.00E-01	2.00E-02	1.00E-03

Reinforcement Learning

As discussed in Schoen (2023), we use reinforcement learning (RL) to simulate traces through the Petri Net model that represent human-robot collaboration, exploring multiple orderings of actions and optimizing based on the costs associated with transitions. Specifically, we use Proximal Policy Optimization (PPO) (Schulman et al., 2017) combined with action masking (Huang and Ontañón, 2020) to restrict the available action space to only viable actions at each timestep. PPO acts as a middle ground between policy gradient methods (Sutton et al., 1999; Grondman et al., 2012) and trust region policy optimization (Schulman et al., 2015), using a clipped objective function that penalizes large deviations in the model between iterations, ensuring small, iterative steps toward the optimal policy.

Reward Function

Conceptually, when training, the RL agent must select transitions for all the human and robot agents specified in the job to progress the time of the simulation. This allows us to explicitly account for actions that progress the state of the collaboration, while also allowing agents to rest. Each selected transition results in a negative reward (cost) to the RL agent based on

the transition's specified onetime and extrapolated costs for ergonomic and economic metrics and the user-specified α value. Rest actions for agents are rewarded based on the ratio of total task time and exertion time, with higher ratios being positively rewarded, incentivizing rest, and lower ratios incurring costs, disincentivizing rest. Additionally, the agent trains against deadlocks, similar to Hu et al. (2020), by imposing high costs to transitions that result in no viable transitions in the subsequent state, such as by removing all agents from the job. If the RL agent encounters one of these transitions, the action is penalized, and the training episode is ended. Overall, the cost associated with each transition is the combination of the ergonomic cost associated with the transition itself and the economic cost of labor and parts.

These costs result in the RL agent accumulating an increasingly negative reward, which it attempts to minimize. PPO works best when providing a combination of negative and positive rewards to help it find the optimal policy. We reward the RL agent positively for transitions that progress towards the goal, as defined by the user's set of tasks, as well as when it reaches the goal state, which is marked by having produced all *Products* associated with the *Job*, resulting in a large positive reward.

Specifically, the reward function for the RL agent is broken down into several factors. It first checks whether the selected action results in the agent reaching the goal state (Equation 5.3) or creates an invalid state (Equation 5.4), such as by using targets that do not exist.

$$goal_i \begin{cases} 10000 & \text{if at goal state} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

$$inv_i \begin{cases} -100000 & \text{if state is invalid} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

The reward function also checks for deadlocks, checking whether the

new state results in the discarding of all agents. If all agents are discarded, the agent is given a negative reward and the episode is concluded (Equation 5.5).

$$dis_i \begin{cases} -100000 & \text{if all agents discarded} \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

In order to incentivize rest, we explore a simple exponential function based on the relation of the total time the agent spends exerting energy to the total time spent on the task (Equation 5.6). This incentivizes the RL agent to allocate rest actions when the ratio approaches 1, and disincentivizes rest as it approaches 0.

$$rest_i \begin{cases} e^{\frac{\text{exertion time}}{\text{total task time}}} - 1.5 & \text{if rest action used} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

Finally, the RL agent checks to see if the action progresses the *Job*, using the *Tasks* defined by the user to determine the reward of increasing linear reward (Equation 5.7. For example, if the action is the first *Task* of n needed to complete the *Job*, then the reward is 1, whereas if it is the n^{th} *Task*, the reward is n.

$$prog_i \begin{cases} i & \text{if action is the } i^{th} \text{ Task} \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

Altogether, the RL agent's reward function for taking some $action_i$ is defined as the sum of these components, minus the ergonomic and economic costs balanced according to the α parameter as described earlier, and minus 1 as a cost for exploration (Equation 5.8).

$$\begin{aligned} r_i = {} & dis_i + inv_i + rest_i + goal_i + prog_i - \alpha \times econ_i \\ & - (1 - \alpha) \times ergo_i - 1 \end{aligned} \tag{5.8}$$

This creates a multifaceted reward function, allowing the RL agent to optimize for many factors of the collaboration.

PPO Implementation

To implement this reward function, we utilized Stable-Baselines3's (Raffin et al., 2021) contrib repository for their implementation of the PPO algorithm with action masking. Action masking allows for the constraining of the action space at each timestep of the simulation, restricting the agent to only valid actions. If the prerequisites of a transition are not met, then it is not considered a valid transition for the given state. For example, producing a *Product* part without using the necessary inputs would be invalid. This reduces model training time by removing the need to train against invalid transitions, focusing only on the viable possibilities for a given state.

While the reward function determines the value of specific actions on the Petri Net, the action mask allows us to restrict the available actions at each time step to only potentially viable actions. The action mask uses the outgoing arcs for each place and transition in the Petri Net and the available agents to determine viable actions. The simulation tracks whether agents are free to engage in an action or if they are already involved in an assigned task. The action mask marks all the transitions associated with busy agents as invalid. Next, the action mask uses the outgoing arcs of each place that connects a transition and checks to ensure the place has enough tokens to satisfy the transition, marking it invalid if not. The remaining set of transitions is marked valid and used to select an action for the current step of training.

The action mask initially restricts the action space further. The RL agent distinguishes transitions in the Petri Net between *Simulation* and *Setup* transitions. *Setup* transitions reflect the initial setup phase of the simulation, where decisions about which agents to include, work allocation, and

placement of *Targets* in the environment are made. *Simulation* transitions reflect the transitions associated with the collaboration itself, capturing the movement of agents through the environment and the progress towards the completion of all *Products*. As a result of this, when beginning model training, the action mask initially restricts the RL agent to consider only *Setup* transitions, forcing the agent to fully set up the environment prior to simulation.

As described by Schoen (2023), we represent the Petri Net network as a set of matrices and vectors. Within the simulation, each observation of the current state of the Petri Net is represented as a vector, where each index represents the number of tokens in that place. We represent the change to the Petri Net caused by transitions as a matrix, with rows indexing the places in the Petri Net and columns indexing the transitions. Each index_{ij} in the matrix represents the net change in tokens at place_i for transition_j. For example, if transition_j consumes 1 token from place_i and then produces 1 token for place_i, index_{ij} is 0, whereas if it produced 0 tokens then index_{ij} would be -1. Additionally, as each transition has a time associated with its execution, we split this matrix into two, one for token consumption $\mathtt{input}$ and one for token production $\mathtt{output}$. This allows us to consume the input tokens when the RL agent selects an action, and then add the output tokens after the transition has finished firing.

As *Allocobot* allows for multiple agents to be utilized collaboratively or independently, this consumption and production of tokens simplifies the changes to the network. *Allocobot* assumes a max firing of the Petri Net at each timestep to progress the simulation, meaning that all possible agents must perform an action. For each agent, the RL agent selects an action, and the $\mathtt{input}$ matrix is used to consume the corresponding tokens from the Petri Net. As each agent completes their action, the Petri Net is updated using the $\mathtt{output}$ matrix, and the agent is assigned a new action. In this way, each observation of the Petri Net correctly reflects the current tokens

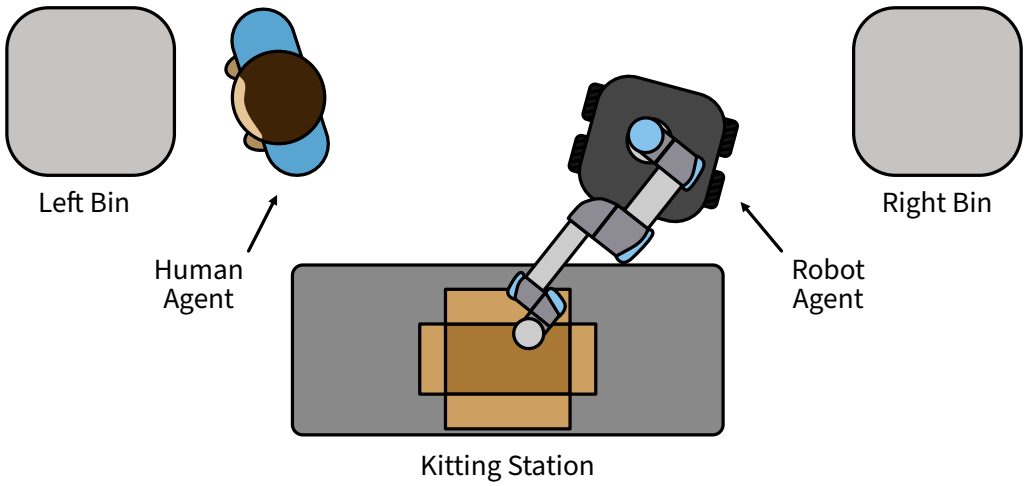

Figure 5.2: Example environment for the kitting task case study. It includes two bins to pick parts from to assemble at the kitting station.

available within the network, but does not represent the held incoming tokens.

After training, we are left with a policy network that we apply to the problem space and determine the optimal sequence of tasks to complete the collaborative job. This network allows us to iteratively step through the collaboration and identify the costs associated with the collaboration, the allocation of tasks between the various agents, and the impact of the tasks on each agent. This information is persistent and recoverable after training, allowing for in-depth analysis on the resulting collaborative task.

5.5 Case Study

To demonstrate how *Allocobot* supports user creation of human-robot collaborations, we developed an illustrative workflow that discusses system behavior and output. This workflow is inspired by prior work in robot kitting (Mancini et al., 2018; Balakirsky et al., 2013), where a robot and human worker must grab objects from locations in the environment to

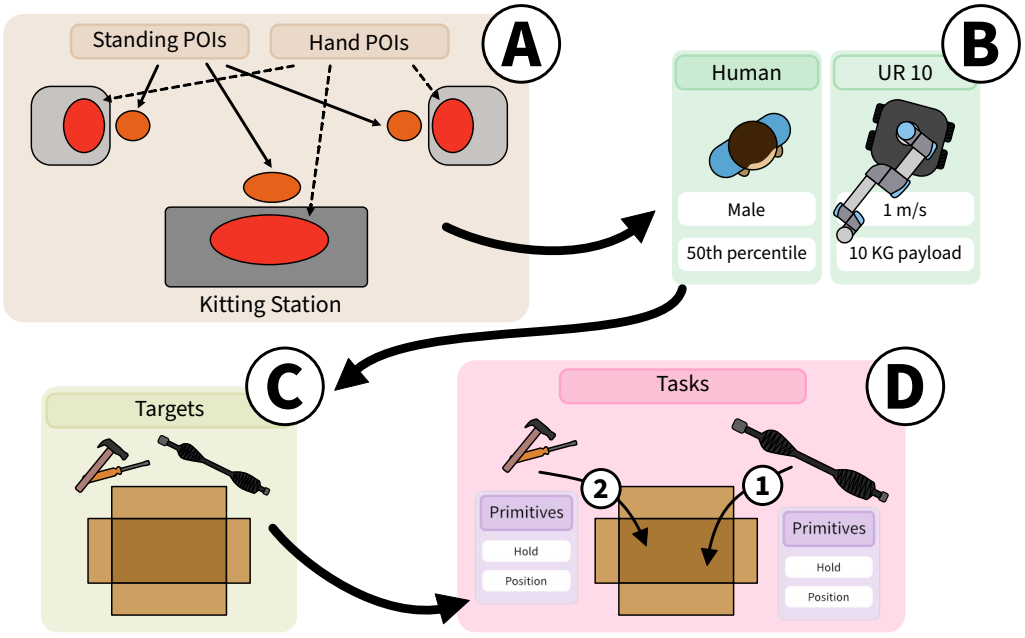

Figure 5.3: The user begins by defining all POIs in the environment (A), then specifies the desired *Agents* for the collaboration (B). Then the user defines all *Targets* (C), and the tasks that use them to complete the job (D). Finally, the user simulates the interaction to produce the best collaboration.

pack a "kit" used in another job. To illustrate the workflow, we created a mock environment (Figure 5.2), and present the workflow at a high level in Figure 5.3.

The user wants to explore an existing kitting task in their facility in the framing of human-robot collaboration. The user begins by analyzing their task environment, defining it within the *Allocobot* system. They note that the environment contains two bins, which contain the parts for the kit, and a table where the kit is assembled. The user marks the locations next to the bins and table as standing POIs within the system, and the areas where agents will interact with the target parts as hand POIs, using a relative positioning of locations based on the table where the kit is assembled.

The user then adds the potential agents for the collaboration. They first

specify a human agent as a potential agent to be used in the job, defining the job as fit for the 50th percentile male, and adding wage and hiring cost information. Then the user specifies the information for the robot agent, opting for the UR10 robot on a mobile base with a max speed of 1 m/s, and defines the energy consumption and initial investment cost of the robot setup.

Next, the user begins to define the targets involved in the task. They know the task involves the kit itself and two objects, one from each bin, that must be placed within the kit. They specify precursor targets for the kit and both objects to be placed in it, followed by intermediate targets representing the box with one precursor or the other, and a product target representing the kit with both objects in it. For each target, they restrict the potential locations to the corresponding areas in the environment, better reflecting how they want the task to be conducted.

Then the user begins to think about the order of steps to complete the task, namely, the addition of each part to the kit. They want one part to be added before the other due to its weight, so they define the transitions to add a single object at a time to the kit. Following this, the user adds the primitives associated with each transition. As each transition is the addition of the part to the kit, the user knows that they only need to add a *Position* and *Hold* primitive to represent the placing of the object within the kit.

Having defined the environment, agents, and task, the user turns their attention to the final parameter for the job. The user needs to determine the α weighting factor for the job, setting it to 0.5 to represent weighing both the ergonomic and economic factors equally when evaluating the collaboration.

Finally, the user begins the reinforcement learning process using the job they specified and analyzes the output. Once training has been completed, the user finds that the produced collaboration focuses on the human

worker only, discarding the robot. To understand the impact of the α weighing, the user adjusts the weighting to 0.3, indicating a weighting of $70\% - 30\%$ split for ergonomic and economic factors, and reruns the simulation. This time, the simulation produces an interaction with the robot adding the heavy object to the kit and the human adding the lighter one.

5.6 Discussion

Designing effective human-robot collaborations is a complex challenge, requiring specialized knowledge across robotics, human factors and ergonomics, economics, and human-robot interaction. This represents an open challenge for industry, where users with knowledge across these disciplines are rare. *Allocobot* represents our approach to addressing this issue. Through a combination of Petri Nets, unique action primitives, and reinforcement learning, *Allocobot* enables users to create and explore human-robot collaboration plans.

Our case study highlights how *Allocobot* supports users through the planning process. The user focuses on describing the environment, component parts, and steps necessary for the collaboration, without needing to explicitly model capabilities, constraints, or strategies for collaboration. Instead, they rely on *Allocobot*'s abstraction of agent capability and task planning to generate complete human-robot collaborations, optimizing for ergonomic and economic factors. In this way *Allocobot* allows users to focus on specific aspects of the jobs and environments they should be familiar with while enabling them to create human-robot collaborations.

This approach has several implications for the design of systems for human-robot collaboration. First, by enabling users to create effective collaborations without requiring multidisciplinary expertise, *Allocobot* lowers the barrier of entry for human-robot collaboration. Users can instead

leverage their existing knowledge and environment to explore potential collaborations. Second, *Allocobot* supports user understanding by allowing for the exploration of trade-offs in the design of collaboration. Users can adjust costs, primitives, or factor weighting through the α parameter to understand how these affect the resulting collaboration. Additionally, users are not restricted to real-world robot designs and limitations, allowing them to freely explore the potential minimum specifications needed to facilitate a successful collaboration. Overall, this ability to rapidly model, simulate, and compare scenarios is important in facilitating user understanding of collaborations.

Limitations & Future Work

One limitation of our work is the lack of a planned empirical evaluation to assess *Allocobot*'s capability to support users in producing viable real-world collaborations. Although our case study illustrates how the system is designed to enable user creation of human-robot collaborations, future work will seek to explore the efficacy of *Allocobot*'s output collaborations.

Our approach has several additional design limitations. First, we assume a deterministic representation for the timing of actions within our Petri-Net representation. While in the real world, the time it takes to complete a given action is variable, we used a simplified approach, assuming the best-case scenario for any action times. Second, we assume human agents are incapable of failing any given task if they are ergonomically capable of completing the task. For the robot agents, a percentage of monetary cost was calculated to account for the possibility of such an agent failing the task, but no such percentage is calculated for human agents. Accounting for the variable timing of actions and the potential failure of actions introduces additional model complexity and should be explored in the future to better reflect real-world expectations of collaborations.

Another limitation is in our modeling of the environment for the Petri-

Net. To serve as an initial step to address the collaborative planning problem and reduce complexity, we assume that objects in the environment are static unless acted upon by agents. This results in difficulty modeling for factories with automated assembly lines and conveyor belts, as *Allocobot* requires the conveyor systems to be agents or for the static locations of objects to be defined.

Finally, we could not find suitable models for robot capability in the literature. *Allocobot* encodes a model of robot capability based on our understanding, but future work should seek to empirically understand real-world robot capability, performance, and failure rates.

5.7 Chapter Summary

Planning human-robot collaboration is a difficult task that requires extensive knowledge of human factors, ergonomics, economics, robotics, and human-robot interaction. This presents several challenges for manufacturers seeking to use cobots. *Allocobot* attempts to simplify this process for users, abstracting the process of allocating work between human and robot agents based on their capability and the ordering of tasks to facilitate a collaborative task. This abstraction enables users to focus on the environment and task they are already familiar with, leveraging their domain and task knowledge as inputs that *Allocobot* can decompose and assign to agents based on its representation of multiple domain factors. This provides additional scaffolding to the user, enabling them to create and iterate through multiple collaborations, each potentially viable.

This chapter presents an update to the design of the *Allocobot* system, a system for supporting user creation and exploration of human-robot task allocations. Our method utilizes models across multiple disciplines to analyze and understand what viable human-robot collaboration looks like. Our reinforcement learning approach, in theory, allows for dynamic

environments, accounting for both optimal and suboptimal options for facilitating the interaction, due to the nature of its policy-based output. Finally, we demonstrated through the use of a case study how *Allocobot* allows users to create human-robot collaborations, leveraging the abstract representations of multiple domains within the system while focusing on user domain knowledge and expertise through the incorporation of simple system inputs.

6 EVALUATING ALLOCOBOT AS A WORK ALLOCATION AND TASK PLANNING OPTIMIZATION TOOL

This chapter presents our evaluation of the Allocobot task allocation and scheduling system. It explores multiple real-world scenarios and highlights how human-robot collaboration can be achieved in each. In this chapter, we discuss the motivation for the work, highlight relevant related work, describe our approach to evaluating the system, present our findings, and discuss the use of the Allocobot system in creating human-robot collaborations. This chapter includes work from a manuscript in progress (White et al., 2025a).

6.1 Motivation

Planning for effective human-robot collaboration is a complex task, involving multidisciplinary knowledge across ergonomics, human factors, robotics, and human-robot interaction. Successful collaboration depends on balancing these factors with task efficiency, productivity, and economic constraints to ensure that human well-being is prioritized along with the overall performance of the system. As industry increases its adoption of collaborative robot (cobot) technologies (Grand View Research, 2023), there is a growing need for methods that can support the planning of human-robot collaborations, as the diversity of factors involved makes this process challenging for real-world scenarios.

Prior work has explored several ways to increase cobot accessibility in the hopes of supporting user understanding, such as by making cobots accessible to the community (Ionescu and Schlund, 2019; Ionescu, 2020), or through the creation of novel programming systems (Schoen et al., 2022; Emeric et al., 2020). Additional work has been conducted to generate collaborative plans using techniques in optimization (Pearce et al.,

2018), genetic algorithms (Bobka et al., 2016), and reinforcement learning (Zhang et al., 2022). However, while some work optimizes for factors such as ergonomics (Potvin, 2012) or productivity (Wang and Li, 2019; Huang et al., 2021), optimizing for factors across multiple disciplines while simultaneously allowing for dynamic workflows remains challenging.

Our work seeks to evaluate the Allocobot system, a task allocation and scheduling optimizer for human-robot collaboration. Allocobot integrates metrics from ergonomics, human factors, economics, robotics, and human-robot interaction to produce task allocation policies that reduce strain on human workers while enabling flexible workflows. In this chapter, we present several real-world scenarios and use Allocobot to explore what human-robot collaboration can look like within them. Through these real-world scenarios, we investigate how Allocobot balances trade-offs in human-robot collaboration and adapts to diverse human-centered scenarios.

The contributions described in this chapter include[1]:

1. An empirical evaluation of the Allocobot system using real-world case studies.

2. A discussion of the impact of Allocobot's parameters on human-robot collaboration.

6.2 Related Work

Cobots are designed with a focus on safety and ease of use, which has contributed to the growing adoption of the technology within the manufacturing domain (Grand View Research, 2023; Simões et al., 2020). Their ability to work collaboratively with people as teammates (Christiernin,

[1]The research in this chapter is derived from a manuscript in progress by myself, Dr. Andrew Schoen, Dr. Anna Konstant, Dr. Josiah Hanna, Dr. Robert Radwin, and Dr. Bilge Mutlu.

2017; Michalos et al., 2015), increasing productivity through task parallelization (Pearce et al., 2018) or through interacting together collaboratively to facilitate task completion (Wang and Li, 2019). However, prior work has highlighted the difficulty manufacturers face when attempting to integrate collaborative robots into their facilities (Sullivan et al., 2024). Small-to-medium enterprises often lack workers with the skills necessary to effectively utilize cobots, due in part to the growing worker shortage and skills gap (Giffi et al., 2018; Michaelis et al., 2020; Wingard and Farrugia, 2021).

Determining how best to use cobots within an application is difficult. While it is known that cobots are capable of reducing ergonomic impact on workers (Liao et al., 2023; Dalle Mura and Dini, 2022), this often has tradeoffs with task productivity (Pearce et al., 2018; Wang and Li, 2019; Huang et al., 2021; Raatz et al., 2020). This presents challenges for determining what an acceptable level of impact is on the human worker, but is further complicated when considering how to operate cobots safely within the environment (Kildal et al., 2018; Malm et al., 2019), and how they should interact with individuals (Galin and Meshcheryakov, 2020; Khalid et al., 2017).

To address this difficulty, prior work has explored several approaches for generating, deploying, and analyzing policies for facilitating human-robot collaboration. These methods include techniques in optimization (Pearce et al., 2018), simulation and genetic algorithms (Bobka et al., 2016; Brosque et al., 2020), and reinforcement learning(Zhang et al., 2022; El-Shamouty et al., 2020; Modares et al., 2015). These approaches attempt to simplify the process of building human-robot collaborations, providing concrete metrics by which to evaluate the collaboration, such as the ergonomic impact on the human collaborator (Potvin, 2012) and task productivity (Pearce et al., 2018).

Allocobot is a system that seeks to address the challenges presented

here by encoding multiple models of domain knowledge and using them to optimize human-robot interactions. Our work seeks to evaluate the Allocobot system's capability of creating viable task allocations and interaction plans.

6.3 Method

Here we describe our experimental setup to evaluate the Allocobot system.

Scenarios

We demonstrate Allocobot's capabilities through two real-world scenarios: automotive manufacturing and small parts assembly. These scenarios were selected due to their potential for human-robot collaboration and to demonstrate Allocobot's versatility across varying environments and collaborative scenarios.

Each scenario was modeled based on video analysis of real-world workers performing the job's associated tasks. From this, we produced a detailed breakdown of the job, along with the associated primitives for each task, as well as modeled the environment the agents must interact in (Figure 6.1).

Scenario 1: Automotive Manufacturing. The automotive manufacturing scenario is inspired by the half-shaft installation job at a partner automotive manufacturer. This scenario involves workers moving around a workcell to gather parts and tools from various locations and bring them to an engine on the other side of the workcell for installation (Figure 6.1 A). To reduce the complexity and variability of the task, we use a simplified setup compared to the real-world equivalent. For this scenario, we assume the engine is stationary as opposed to being on an unguided autonomous vehicle, which is more typical of real environments. Additionally, we simplified the job's steps to combine actions that would not make sense

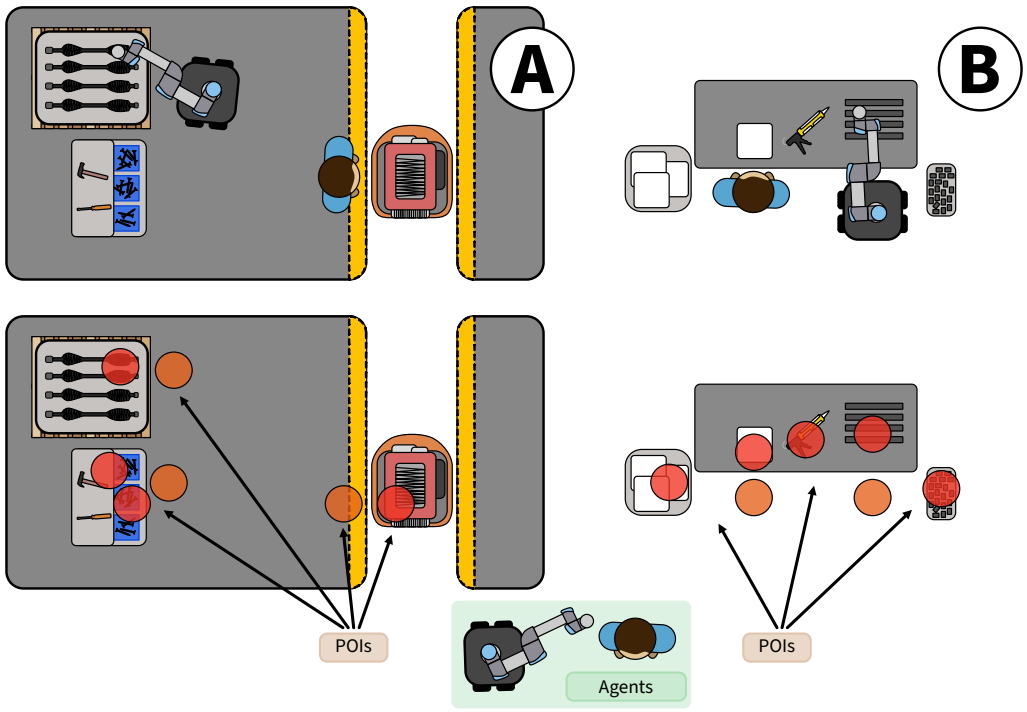

Figure 6.1: Scenario environments for the evaluation of Allocbot. (A) depicts an automotive assembly job where agents will need to move around the environment to grab parts and tools to work on the engine. (B) depicts a small parts assembly job where agents will prepare and assemble parts. For both (A) and (B) areas marked in orange represent the POIs where agents will stand, while the areas marked in red represent hand locations where they will perform work.

to split among multiple agents. For example, when checking whether the half-shaft is properly attached, the agent must apply a force, tugging the object several times to verify its connectedness. Splitting this into multiple steps would allow different agents to each perform a check on the component, rather than guaranteeing a single agent performs all checks.

For the automotive manufacturing scenario, we only consider the initial portion of the job, due to its high ergonomic impact on the human worker. Concretely, we model this high-impact portion of the job within a four-step

process. The first step of this process is the insertion of an engine protection template, followed by the second step where the agent attaches the half-shaft. The third step removes the protector, and the agent completes the job by doing a manual inspection of the work. All pieces and tools used in the scenario are located on the left side of the environment, requiring agents to take multiple trips to bring components from the left side to the right, where the engine is located.

Scenario 2: Small Parts Assembly. The small parts assembly scenario is inspired by an assembly task from a small to medium enterprise (SME) as presented by Sullivan et al. (2024). This scenario involves workers doing prep work on multiple parts before assembling them together. This scenario does not require agents to move about the environment much, as all necessary parts are located near the workstation (Figure 6.1 B).

For the small parts assembly scenario, the job consists of prep work and assembly. Agents must first prepare multiple sets of gaskets and metal pans by applying silicone to them. Following this, agents can attach the gaskets and pans to a "core" through a multi-step process. These are outlined as four distinct actions: prepare gaskets, prepare pans, add pans to the core, and add gaskets to the core. To complete the job, agents must perform each action multiple times, as a fully assembled unit consumes one core, two pans, and four gaskets. Within this scenario, parts are located in bins around the table, with tools being accessible to both agents.

Experimental Setup

For each of the scenarios, we utilized the same set of input parameters for the Allocobot system, only changing the *Targets*, *POIs*, and *Primitives* to match the needs of the scenario. Each scenario was run with two agents, one human and one robot. The human agent was set to the 50th percentile male, meaning the human worker assigned collaboration tasks should be doable by a 50th percentile male. The robot agent was modeled after

the Universal Robot UR10 (Robots). Specifically, we defined our robot agent to have a reach of $1.3\mathrm{m}$, a payload capacity of $10\mathrm{kg}$, an end-effector speed of $1\mathrm{m/s}$, a precision of $0.0001\mathrm{m}$, and an energy consumption of $300W$, as defined in the spec sheet. We set the purchase price for the arm to $\$35,000$, with an estimated annual maintenance cost of $\$5,000$. We ranked the agility factor of the UR10 as medium, and set the desired level of sensing to be low to indicate we want to use the system with minimal add-on features. For each scenario, we assume the UR10 robot to be on a mobile base, with a height of one meter off the ground, and with a mobile speed of $4\mathrm{m/s}$.

For each scenario, we varied the α weighting of economic and ergonomic costs from 0 to 1 in increments of 0.1. This varies the weighting of the ergonomic and economic costs from 100% ergonomic costs to 100% economic costs, with an α of 0.5 meaning an equal weighting between ergonomic and economic factors. For each job and α value, we train the RL agent for one million steps.

Measures & Analysis

We conducted a quantitative analysis of Allocobot's task allocations across varying α values to understand the impact on agent behavior, task allocation, task efficiency, and overall cost. Allocobot produces logs with agent assignments, primitive-level actions, and task metadata. The assignment of tasks to either the human agent, robot agent, or a combination of them, as well as the primitives associated with them, was extracted from these logs and visualized to examine patterns in collaboration and workload distribution. Additionally, logs were analyzed for non-allocated work, such as the fetching of parts.

Both ergonomic and economic costs were extracted from these logs to assess the α parameter's influence on task characteristics. To assess the physical impact on the human worker, we used Allocobot's internal

ergonomic cost model. This model tracks strain across specific body areas, including the hand, arm, shoulder, and whole body, based on maximum voluntary contraction (MVC). We use this value to illustrate how the allocation of work impacts the human agent. Monetary cost was also recorded for each allocation, including both setup costs and operational costs. In addition to the total cost, we computed an alternative version that excludes the initial robot setup cost to isolate the cost difference driven by agent time and material use.s

Finally, the time to complete the task was computed based on the duration of all actions executed in a given allocation up until the final product was created. This allowed us to measure the efficiency of each allocation strategy and understand the role of parallelization and agent substitution in reducing overall task time.

6.4 Results

Here we present the results of Allocobot for each job. For each job, we vary α from 0 to 1 in increments of 0.1 and illustrate how it affects the resulting collaboration.

Scenario 1: Automotive Manufacturing

Figure 6.2 demonstrates the result of Allocobot's allocations for the Automotive Manufacturing scenario, highlighting both the agent assigned to each task as well as the primitives defining their role in that task. Overall, Allocobot produced varied task allocations and primitive assignments, resulting in varied impacts on the human agents and overall task time.

When ergonomic factors are the only factors being prioritized ($\alpha = 0.0$), Allocobot allocated the robot to the entire task. This is expected, as ergonomic strain does not impact the robot, but it also indicates the error rate of the robot is acceptable compared to the cost that would be incurred

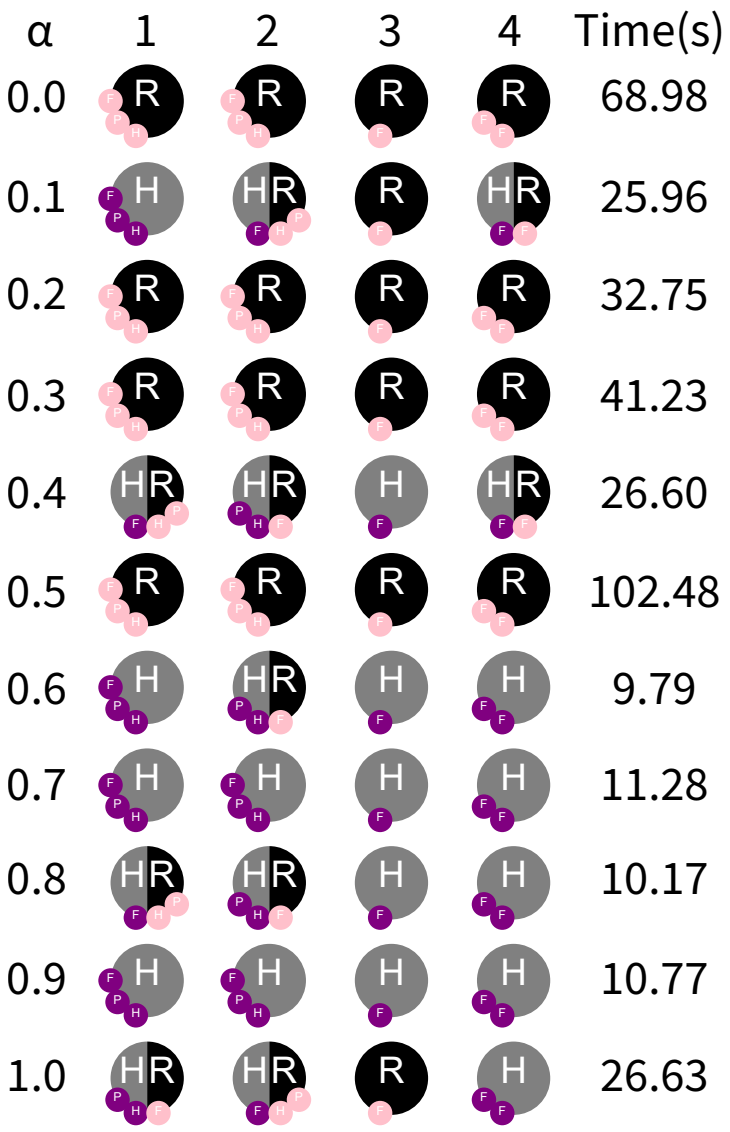

Figure 6.2: The change in work allocation and primitive assignment due to the change in alpha weighting for the Automotive Manufacturing Scenario's four-step process. For each α value, we see the resulting allocation of robot (R), human (H), or human-robot (HR) to the task. Additionally, each allocation specifies which primitives are assigned to each agent, including force (F), position (P), or hold (H). Finally, on the right is the total cycle time for the collaboration.

through collaboration. However, we see a similar allocation for α values 0.2, 0.3, and 0.5. While this could indicate a similar prioritization of factors, the total time for the task varies for each. When analyzing the outputs of these collaborations, we find that for $\alpha = 0.2$ the human agent is added to the task to carry the protector for the robot, shortening the overall task time when compared to $\alpha = 0.3$. For $\alpha = 0.5$, the human agent was not added to the task, but the policy determined that the robot agent should rest for 60 seconds following adding the half-shaft to the engine. If this action were removed, the time would more closely align with $\alpha = 0.3$.

However, in most scenarios, the human agent was still added to the job, with the exception of α values 0.3 and 0.5, where the human agent was not added. As the robot was fully allocated work for α values 0.0, 0.2, 0.3, and 0.5, we analyze tasks 0.0 and 0.2 further to understand the role of the human agent. For $\alpha = 0.2$, the human agent was added to the task and carried a protector before resting for the remainder of the task. In this way, the human agent was used to parallelize part fetching for the robot agent to use. Similarly, for $\alpha = 0.0$, the human agent was used to fetch a half-shaft. However, the result of the human carrying the half-shaft was ignored as the policy had the robot agent grab a second half-shaft that was utilized within the remainder of the process.

For the Automotive Manufacturing scenario, task two of the process requires agents to add the half-shaft to the engine, and is the most strenuous job for the human agent to perform. This task involves agents holding the half-shaft, positioning it in the correct place, and applying force to attach it. However, in two scenarios, α values 0.7 and 0.9, Allocobot allocated the entire job to the human agent.

Figure 6.3 shows how each of the allocations produced by Allocobot impacted the human agent. Namely, it illustrates that the Automotive Assembly job primarily focuses on arm strain as measured by Allocobot. Additionally, it shows that the allocations for α values 0.0, 0.1, 0.7, and

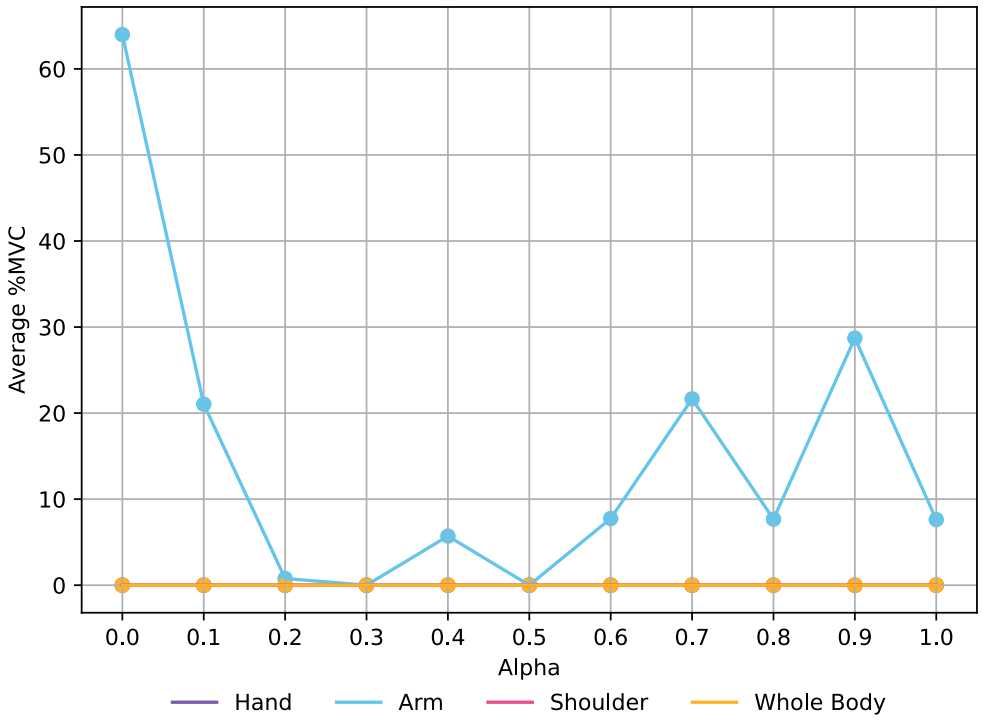

Figure 6.3: The ergonomic impact on the human agent measured by Allocobot for each produced allocation in the Automotive Manufacturing scenario.

0.9 result in the highest impact on the human worker. For 0.7 and 0.9, this is expected due to the allocation of work only using the human agent. However, similar to α values 0.0 and 0.2 with the roles reversed, Allocobot adds the robot agent for $\alpha = 0.7$, having the robot carry the half-shaft for the human agent to use in the process, whereas for $\alpha = 0.9$ the robot agent is not included in the task. Additionally, we see high values for α values 0.0 and 0.1 caused by carrying the half-shaft (for both) and then inserting and checking its installation (for $\alpha = 0.1$). The value for 0.0 is especially high, given that the human agent's only task is to carry the half-shaft.

Figure 6.4 shows how each allocation performs in terms of overall cost

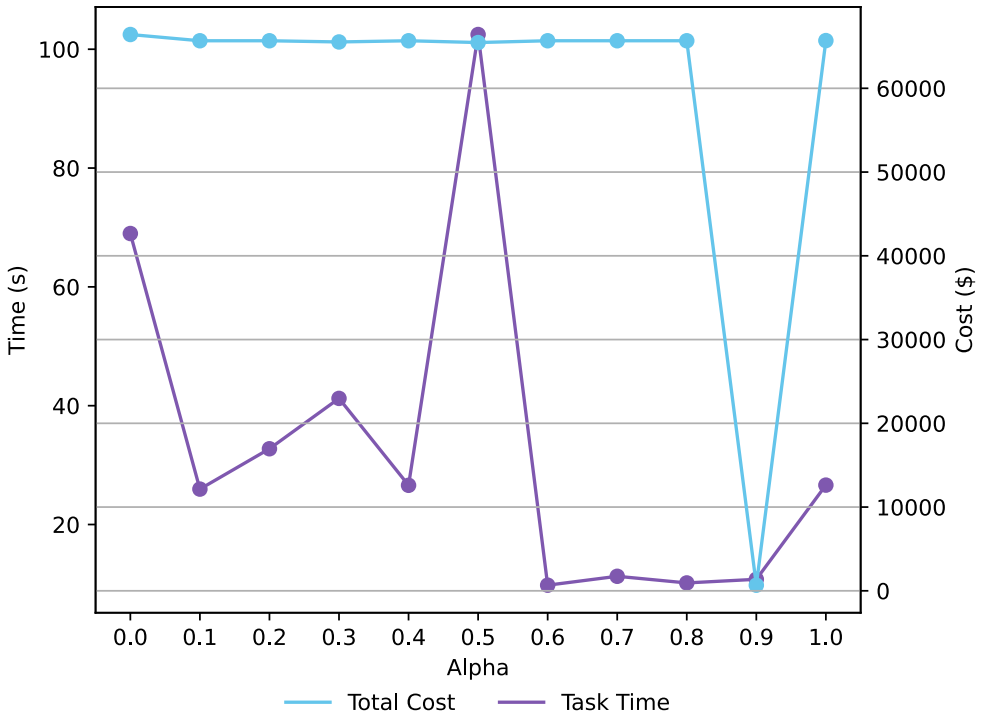

Figure 6.4: The total cost and time for task completion for each produced allocation in the Automotive Manufacturing scenario.

and time needed to complete the task. In terms of cost, notably $\alpha = 0.9$ was the only instance where Allocobot did not include the robot agent in some form. When removing the setup costs, such as the hiring cost of the human agent and the purchasing cost of the robot setup (Figure 6.5), we see little overall impact due to the change in the α parameter. We would expect this, as the parts needed to complete the task should be consistent. We see that for most values of α, the cost is well optimized. In these instances, the agents perform the bare minimum actions required to complete the task and use only the parts necessary. However, α values 0.0, 0.3, and 0.5 result in higher costs stemming from the potential for error when the robot agent performs certain actions, such as when it carries the

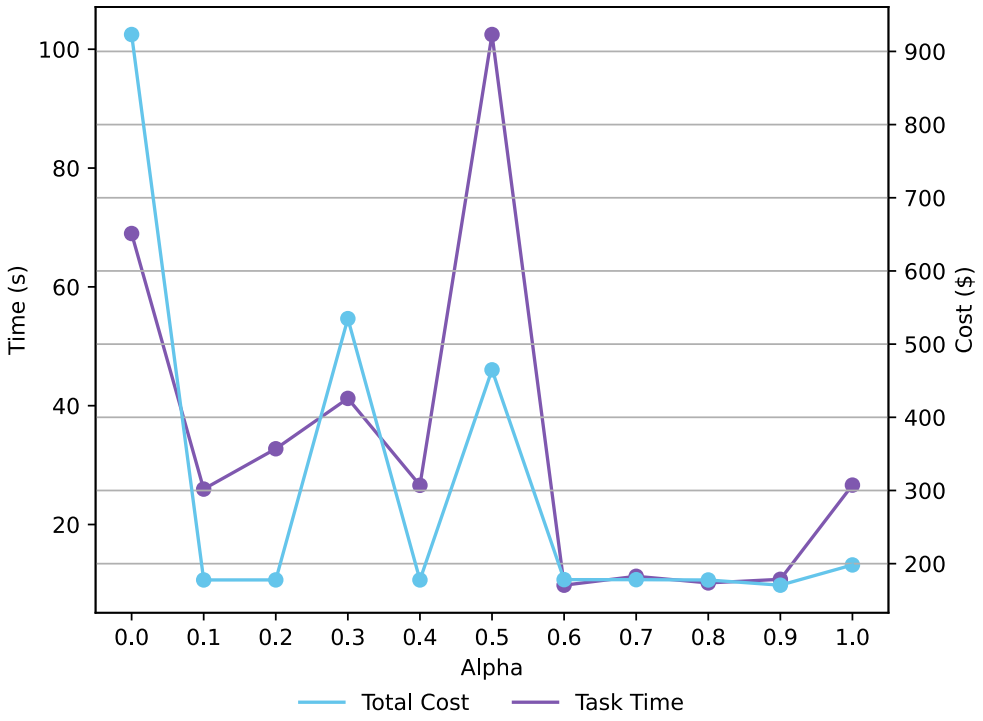

Figure 6.5: The total cost of the task, minus the cost of the task setup, and time for task completion for each produced allocation in the Automotive Manufacturing scenario.

half-shaft in $\alpha = 0.5$.

However, we do see a noticeable decrease in task time when looking at α values greater than 0.5 compared to those less than 0.5. Analyzing these allocations, we find that this difference is primarily due to the reliance on the robot to fetch parts. The robot performs actions slower than the human agent, severely impacting overall task time. When looking at the best case for the robot agent when it performs the task alone, we see that $\alpha = 0.3$ has a time of 41.23 seconds. By adding the human agent to fetch one part, the protector in $\alpha = 0.2$, we decrease this time to 32.75 seconds. For the allocations where $\alpha > 0.5$, these actions are more common, with the

human agent frequently moving around the environment to fetch parts, or both agents fetching parts simultaneously.

Scenario 2: Small Parts Assembly

Figure 6.6 shows the result of Allocobot's allocations for the Small Parts Assembly scenario. For most values of α, the system was not able to find an allocation that completes the task. Of the α values that resulted in completed tasks, all except for 0.5 result in a sharing of work for at least one task within the job. For $\alpha = 0.5$, we see that the human is allocated all tasks, while also having the shortest total time for completing the job. Analyzing the allocation output further shows that Allocobot still adds the robot agent to the job, but does not utilize it at all. Instead, the robot agent rests throughout the entire job. In contrast, $\alpha = 0.1$ has the highest total job time. This is due to Allocobot not effectively utilizing the agents, adding multiple rest actions for the agents that produce a negative reward, as well as using additional parts within the process. Similarly, both α values 0.0 and 0.5 also add additional steps to complete the task, having the agents prepare multiple additional parts before completing the task.

In terms of ergonomic impact, as shown in Figure 6.7, the Small Parts Assembly job primarily utilizes arm-based movements and actions, similar to the Automotive Manufacturing scenario. However, this scenario is not strenuous on the human worker, with an average %MVC of about 12% in the worst scenario. While the job was not completed for $\alpha = 0.9$, it was the only scenario where Allocobot did not include the human agent in the job, resulting in an average %MVC of 0%. From the allocations where the task was completed, $\alpha = 0.8$ resulted in the human agent having the smallest average %MVC, but also took the longest for the task to be completed. Analyzing this output further, we find that the forces exerted by the human agent are minimal throughout the task, sharing the workload with the robot for most actions.

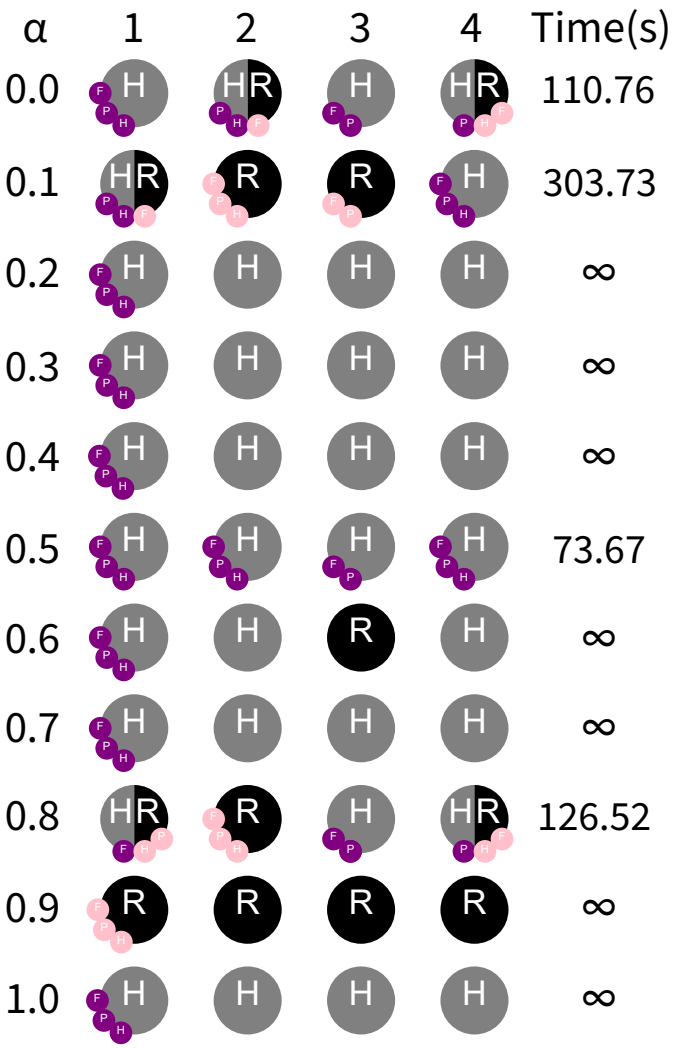

Figure 6.6: The change in work allocation and primitive assignment due to the change in alpha weighting for the Small Parts Assembly's four-step process. For each α value, we see the resulting allocation of robot (R), human (H), or human-robot (HR) to the task. Additionally, each allocation specifies which primitives are assigned to each agent, including force (F), position (P), or hold (H). The lack of assigned primitives indicates that the task has not been performed after being allocated to the agent. Finally, on the right is the total cycle time for the collaboration, where ∞ indicates the task is not completed.

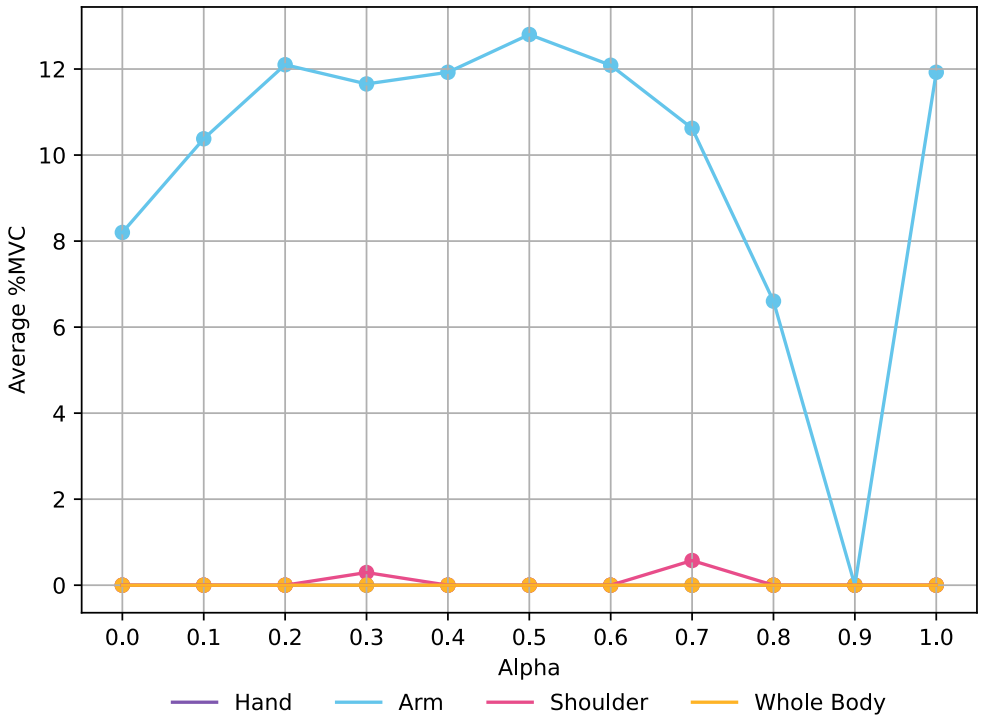

Figure 6.7: The ergonomic impact on the human agent measured by Allocobot for each produced allocation in the Small Parts Assembly scenario.

Figure 6.8 shows the trade-off in total task time and monetary cost, ignoring the setup cost for the job. When analyzing the tasks, we find that for α values 0.2, 0.3, 0.4, 0.6, 0.7, 0.9, and 1.0 Allocobot has agents continually grabbing parts to use in the process, but not using them to progress the state of the job. While this process of continually grabbing parts but not using them results in some incurred cost as measured by the Allocobot system, these costs are minimal when compared to the costs associated with the agents completing the task.

Additionally, there is a notable discrepancy between α values 0.0 and 0.1 with values 0.5 and 0.8. When analyzing the outputs, it becomes clear that α values 0.0 and 0.1 similarly create extra parts that are not utilized

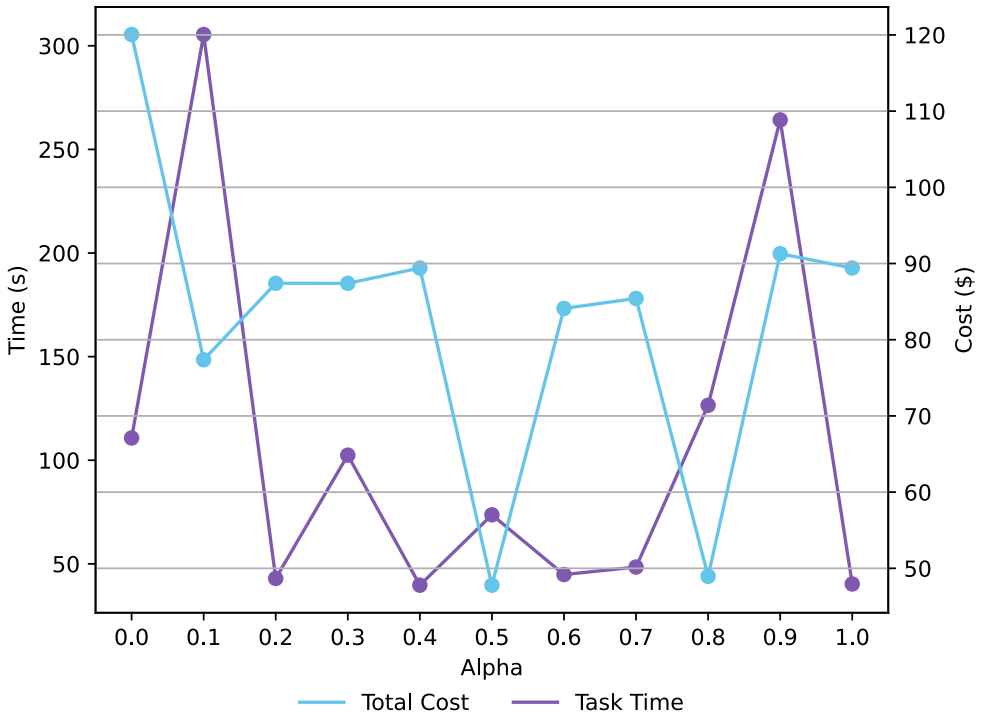

Figure 6.8: The total cost of the task, minus setup costs, and time for task completion for each produced allocation in the Small Parts Assembly scenario.

within the process. In this regard, α values 0.5 and 0.8 represent near-optimal use of parts, with discrepancies between them occurring due to labor and electricity costs.

6.5 Discussion

We presented several real-world scenarios that illustrate the capability of *Allocobot* to produce viable plans for human-robot collaborations. These scenarios demonstrate how changes in factor prioritization influence the characteristics of the job, such as task allocation and ergonomic strain

on the human agent. Overall, this highlights the versatility of the Allocobot system, as it produces multiple different forms of human-robot collaboration, dependent on the α parameter. These collaborations go beyond simple task allocation, revealing patterns of task parallelization and concurrency that are only evident when analyzing the job outputs.

In the Automotive Manufacturing scenario, Allocobot produced successful allocations that completed the scenario for all α values, demonstrating its adaptability and versatility in allocations. The outputs were feasible, given the input constraints, and reflected multiple collaboration strategies. However, as the Small Parts Assembly scenario demonstrated, the system is not always able to find a solution. While the system will always determine how to allocate work among the potential agents, it may not always result in a completed task. This can be due to the reward function of the reinforcement learning algorithm not sufficiently incentivizing task progression or punishing poor interaction states, resulting in the policy getting stuck in local minima.

While not explored in this work, since Allocobot is an abstract representation of work and capability for agents, this, in theory, allows for the exploration of minimum robot capability to facilitate effective collaboration. In a similar manner of exploration as illustrated using the α parameter, users of Allocobot can adjust robot parameters, such as payload, speed, reach, and cost, to analyze the impact on collaboration and task allocation. This allows users to understand the minimum robot requirements needed to perform the task as desired. This can make cobots more accessible due to the system supporting user understanding of interactions and reducing the barrier of entry that previously required specific knowledge and training.

Overall, our results highlight the potential of the Allocobot system and its ability to produce viable collaborative interactions while removing the need for users to understand multiple disciplines to create human-robot

collaborations.

Limitations & Future Work

Our work presents a set of real-world scenarios that illustrate the creation of collaborative interaction plans, but lacks an understanding of how end-users would interact with such a system to create collaborative interactions. Future work should explore how to build interfaces that leverage user knowledge and support their decision-making in creating effective collaborative interactions.

Additionally, our results demonstrate the limitations of the current Allocobot system. Within the Small Parts Assembly scenario, Allocobot was not always able to produce allocations and outputs that resulted in completed jobs. This indicates that additional work is needed to adjust the internal reward function, as well as increase transparency into the reasons it fails for particular allocations. This transparency can assist user understanding, without requiring them to dig through the output files.

Finally, while one of the benefits of Allocobot's approach is the result of a policy, allowing for adaptation to changing environments even as they deviate from the optimal interaction, our analysis did not explore how robust the system is to these deviations. Future work should explore how such an approach can recover unexpected inputs that cause deviations from the optimal allocation, as real-world deployments that rely on these policies need to reliably be capable of completing the task.

6.6 Chapter Summary

Planning for and creating collaborative interactions between humans and robots is a challenging process requiring in-depth multidisciplinary knowledge. Due to this difficulty, industrial use of cobots is challenging and has resulted in a growing need for systems that support user knowledge

and allow for the exploration and creation of human-robot collaborations. Allocobot represents one method of addressing this issue, optimizing for factors from ergonomics, human factors, economics, robotics, and human-robot interaction.

This chapter presents an empirical evaluation of the Allocobot system using real-world scenarios to illustrate the approach's versatility in creating human-robot collaborative task allocations. The system allows users to focus on concrete input values and output metrics, abstracting the multidisciplinary knowledge that would normally be required to create the collaboration, all while scaffolding the process for users to allow them to build viable human-robot collaborations. While the system does not solve all scenarios, this work demonstrates the potential of the approach while highlighting the need for future work.

7 GENERAL DISCUSSION

7.1 Summary & Significance of Work

This dissertation argues that cobots are difficult to use, both for planning human-robot collaborations and programming them. We argued that this difficulty requires new systems that provide abstraction and scaffolding supports to assist users in creating cobot interactions. The work presented in this dissertation highlights how we engage both end-users and the literature to understand how to provide these supports through system design. In Chapter 3, we presented an update to the CoFrame programming system, a system built on a model of cobot expertise from the literature, and illustrated through case studies how the system's design can support user program creation and understanding. In Chapter 4, we presented the evaluation of the CoFrame system, with domain experts, novices, and a real-world deployment. Our results revealed expert perceptions of the difficulties of cobot use within industry and how the concepts CoFrame encodes are beneficial in addressing the gap. Our results also revealed novice user perceptions of the CoFrame system, highlighting their use of each support and their desire to progressively engage with more advanced concepts and information. The final part of our CoFrame evaluation demonstrated its capability for real-world programming of collaborative interactions, but also revealed the need for additional support to assist users in understanding how to plan for human-robot collaboration. Through our evaluation and engagement with end-users, both in lab studies and in a real-world deployment, we identified how the system supports cobot programming and where additional work is required. Overall, CoFrame represented our first step in addressing the skills gap, demonstrating the potential new systems have in supporting user creation of cobot programs.

However, through the real-world deployment of CoFrame, we identified additional support structures required for users to begin building cobot programs, which led to the creation of the Allocobot system presented in Chapter 5. In discussing the updated design of the system, we highlighted how the system encodes models from multiple domains to enable users to create collaborative allocations and plans based on their domain knowledge. This was demonstrated through a case study that walked through how users would interact with the Allocobot system to create viable collaborations by using its Petri Net representation and reinforcement learning approach. We presented the evaluation of Allocobot in Chapter 6, where we presented two real-world scenarios and analyzed the result of using Allocobot to create collaborations and allocations for each scenario. Our evaluation revealed Allocobot's capability and flexibility in producing multiple types of collaboration across varying environments and interaction needs. This work demonstrated how users can interact with inputs and outputs they understand while leveraging a system that encodes several domains of knowledge to produce viable allocations of work. Allocobot represents our second step to addressing the skills gap by addressing the needs identified within the real-world deployment of CoFrame.

Overall, this dissertation presents both CoFrame, a programming system, and Allocobot, an allocation and planning system, as tools to begin addressing the skills gap identified in industry by supporting user creation and understanding of cobot programs. We demonstrated the way each system accomplishes this through technical and empirical evaluations. We designed and built these systems as end-user tools and hope that new systems and future work continue to build on and expand them to address the challenges of using cobots.

7.2 Evaluation of Thesis

Here, I revisit the thesis statement discussed at the beginning of the dissertation: **Cobot interfaces that integrate abstraction and scaffolding can facilitate planning and programming human-robot collaborations.** In this dissertation, I have adopted the definitions of abstraction and scaffolding as "deciding what details we need to highlight and what details we can ignore" (Wing, 2008) and "support which enables a student to achieve a goal or action that would not be possible without that support" (Guzdial, 1994) respectively. In the remainder of this section, I discuss how interfaces that integrate the concepts of abstraction and scaffolding can facilitate the planning and programming of human-robot collaborations, and how this is supported by the work in this dissertation.

In Chapters 3 and 5, I demonstrated how the design of both the CoFrame and Allocobot systems can support user planning and creation of human-robot collaborations through the abstraction of expert knowledge across multiple domains. Additionally, I demonstrated how the design of the systems can scaffold and support user knowledge and understanding, allowing users to incrementally engage with the system through the adjustment of parameters (e.g., Chapter 5) and gradual introduction of collaborative interaction concepts while providing initial guidance for how to start building programs (e.g., Chapter 3). In both Chapters 3 and 5, I illustrated how a fictitious user can approach and use each system while focusing on their domain knowledge to plan for and program human-robot collaborations. These chapters illustrate the use of abstraction and scaffolding in facilitating user creation of human-robot collaborations by allowing users to focus on collaborations at a representative or goal-driven level (abstraction) while providing strategies and a means for interacting with the higher-level representation (scaffolding).

However, the use of abstraction and scaffolding for facilitating user creation of collaborations was demonstrated in Chapters 4 and 6, where

I presented the evaluations of the Coframe and Allocobot systems. In Chapter 4, I presented the evaluation of the CoFrame system with end-users, demonstrating its capability for real-world applications as well as highlighting how users perceived and leveraged CoFrame's support to build cobot programs. In Chapter 6, I presented the evaluation of the Allocobot system using real-world scenarios to demonstrate how, given a few input parameters, the system can create viable collaborative allocations and plans. These chapters demonstrate how the use of the abstraction and scaffolding supports guide user interaction of the systems and enable them to create effective cobot programs and plans.

7.3 Challenges & Limitations

While each chapter presented in this dissertation outlines various challenges and limitations, they are consolidated and synthesized here to discuss the broader challenges and limitations in building and evaluating systems for creating human-robot collaborations.

Challenges in System and Support Design

Designing systems and support for users to create cobot plans and programs presents a number of challenges. Cobots represent a combination of knowledge across multiple domains to facilitate effective usage. Any system will need to balance between supporting users with the knowledge and capability of utilizing cobots with those who need to learn about them. This can be a difficult tradeoff as the supports needed for one may not overlap with the other. Allowing for users of different backgrounds and expertise presents additional challenges, as highlighted in our evaluation of CoFrame (Chapter 4), where users desired low-level controls or additional functionality for real-world use.

Additionally, it is difficult to combine models and perspectives across multiple disciplines. Systems must consider these perspectives, but when issues arise, it can be difficult to determine how to balance each perspective in addressing the issue. As presented in Chapter 5, Allocobot encodes multiple models for use in its reinforcement learning approach, and is no exception to the challenge of balancing each aspect of these models to produce viable collaborative plans and allocations. While the black box nature of the reinforcement learning approach is desirable as it allows for the combination of these multiple models as well as allows users to focus on inputs and outputs, it obscures insight into decision making and impact of models. This makes the process of understanding model impact on the resulting collaboration reconstructive, where users receive the output collaboration and must contextualize the results according to the input parameters to understand how Allocobot selected each action. This presents challenges when designing systems, as trading off this black box nature may introduce additional complexity for solving the collaboration, but allow for additional understanding regarding decision making.

Finally, designing supports for systems presents several challenges in determining the correct level of abstraction to support user interactions with the system. Low-level abstractions may benefit expert users who already have the depth of knowledge needed for cobot usage, but higher levels are required for scaffolding novice user experiences and learning. This can present challenges for systems that target a wider demographic of users, as this abstraction impacts considerations such as how closely to model and align expectations with the real world. For example, CoFrame (Chapter 3) employed a simplistic simulation model, ignoring aspects of gravity and friction that may cause issues with cobot object grasping. While grasp detection is something expert users consider and deal with, presenting this to novice users results in less time spent learning general cobot interaction concepts. The same principle applies for the modeling

of real-world human behavior and variability, such as in Allocobot (Chapter 5). By using a simplified model, users can more easily understand the outputs and general workflow of the system, but it may not fully translate to real-world implementations.

Challenges in Evaluation

In evaluating the systems created for enabling user creation and understanding of cobot systems, a number of challenges arise. It is known in the literature that it is difficult to recruit expert participants at any point of the designing and evaluating process. As highlighted in our background research, the growing worker shortage and skills gap (Michaelis et al., 2020; Wallace, 2021) put these individuals' time in high demand, resulting in difficulty engaging them outside of the scope of their existing work. While partnering and collaborating with manufacturers is a viable alternative, this produces its own issues. It can limit the scope and generalizability of the evaluation, consisting of processes specific to that partner, but also presents a greater need for system capability to allow for real-world use and contexts. These real-world deployments are highly valuable for understanding user needs and where the system either does or does not meet those needs.

Additionally, when evaluating systems that support user program creation and understanding, evaluation plans and metrics must balance an understanding of system usability with learning outcomes. Measuring learning and understanding is difficult, even at a surface level, as they may require longer studies or comparisons to understand the impact of each support. The work in this dissertation sought to strike a balance, as illustrated in Chapter 4, through the use of multiple approaches and studies with different sets of users.

7.4 Future Work

This dissertation advocates for the use of abstraction and scaffolding to support users in the creation of cobot programs. In the course of this work, we have identified a number of areas for future work to explore.

System Extension and Iteration

As a direct extension of the work presented in this dissertation, there are many opportunities for the iteration and extension of the systems presented. In Chapter 4, we highlight how users desire more advanced features and capabilities, bridging the simplified representations for programming to more traditional methods. This graduation removal of abstraction is known to benefit learning (Waite et al., 2018; Devathasan et al., 2022), and presents several open questions for how to facilitate this within cobot system designs. Then in Chapter 6, we present a system for work allocation and planning. There are several open questions regarding how to design a user interface for such a system that highlights and enables users to focus on the environment and task components of the job to create a viable collaboration.

However, the work in this dissertation presents two separate systems for supporting users. As demonstrated in our real-world deployment (Chapter 4), additional support for programming needed to be explored, which led to our exploration of task allocation and planning. While we explored the use of reinforcement learning to incorporate models across multiple disciplines to produce collaborative plans, this presents trade-offs for understanding outputs. Future work should explore the needs of real-world users to understand whether the additional insight alternative approaches may provide is necessary or if providing tools for analyzing the output of these black box models is sufficient.

While each system addresses the needs of its respective area, new

systems will need to explore the combination of both, supporting the planning process and transitioning that into realizable programs. By creating more coupled and integrated systems, users can move between the planning and programming processes as needed, deriving the support and exploration supported by each process.

System Evaluations

Future work should strive to evaluate systems in real-world contexts with end-users. Chapter 4 demonstrates the benefits of this, gaining expert insight into industry, perceptions, and interactions with the system by novices, and an understanding of how the system does and does not address industry needs in a real-world deployment. This approach allows for greater buy-in from industry through partnerships and collaborations that allow for mutual benefit between parties, and can also demonstrate where system designs are well implemented and where additional supports are required for real-world use.

However, it is well known that the recruitment of experts can be difficult, as discussed in Section 7.3. Future evaluations should consider broader groups of users and experts, encompassing system integrators, technicians, operators, instructors, trade schools, etc. By broadening the scope of potential users, this builds a wider understanding of the needs of potential users across multiple areas of industry, allowing for more generalized support and system implementation as well as enabling more users to create cobot programs. It is important to complement these expert insights with novices, as while experts have the perception and bias of what is currently done within the industry, novices may represent new ideas for approaching solutions, given their lack of exposure to current practices.

Additional Representations and Supports

While the work in this dissertation incorporates several abstraction and scaffolding supports within cobot interfaces to assist user creation of human-robot collaborations, new technology and approaches that can be used for abstraction and scaffolding might increase accessibility and system ease of use. Namely, the introduction of large language models (LLMs) has been increasingly explored in many domains. They can provide familiar interactions through their understanding of language, which might enable users to more easily program or plan collaborative interactions. However, given the prioritization of safety, exploration of this technology within cobots has been minimal, given the lack of guardrails on output generation. However, the pairing of this technology with several designs presented in this dissertation, such as the Expert Frames presented in Chapter 3, can be one method of increasing accessibility and ease of use while maintaining efficiency and safety.

Additionally, while the work presented in this dissertation highlights the planning and programming process, future work will need to explore the connection of programming and planning systems to the physical world. As highlighted in our evaluation of the CoFrame system (Chapter 4), users will eventually want and need to understand the connection between what they are doing within programming and simulation to how it reacts to and impacts real-world environments and collaborations.

7.5 Conclusion

This dissertation presented work for understanding how cobot interfaces that integrate abstraction and scaffolding facilitate user creation of cobot programs and plans. In each chapter, we attempted to highlight the use of abstraction and scaffolding, both for design and evaluation. In particular, we presented the design of a programming system, CoFrame, and an

allocation and planning system, Allocobot, highlighting how the design of the systems encapsulates expert domain knowledge that enables users to build and understand cobot programs. Additionally, we evaluated each of these systems to demonstrate their capability in real-world contexts and to understand how they support users.

While I hope this dissertation has demonstrated the potential for abstraction and scaffolding supports in cobot systems, additional work is needed to both address the growing skills gap and deploy these systems long-term in the real world. While the ideas presented in this dissertation illustrate how to support cobot understanding and program creation, there is a need to connect this work with physical cobot systems and continue deploying systems in real-world environments to engage users. Ultimately, I hope that new systems will iterate on the designs and ideas presented here to build better systems and support for end-users, increasing the accessibility of cobots in the real world.

A.1 Links to Systems

Both systems presented in this dissertation are hosted on GitHub and are publicly accessible.

CoFrame, as described in Chapter 3, is hosted on the Wisc-HCI GitHub page (`https://github.com/Wisc-HCI/CoFrame`), with a live version of the website also deployed there using GitHub's systems (`https://wisc-hci.github.io/CoFrame/`). Users can engage with the live website or download and run the code locally.

Allocobot's code repository is split into two separate repositories. The first repository (`https://github.com/Wisc-HCI/allocobot`) serves to build the Petri Nets, as described in Chapter 5, and the corresponding JSON files that represent them. Users can interact with this code repository to build and configure their desired jobs. The second repository (`https://github.com/Wisc-HCI/PetriNetRL`) hosts the reinforcement learning code that utilizes the JSON files produced by the first repository to learn collaborations.

A.2 Study Data and Materials

All study data and materials that have been presented within this dissertation have been uploaded to an OSF repository (`https://osf.io/hkdse/?view_only=5ab447e7a6a1472dafb34dffcb510973`). The repository contains questionnaires, procedures, anonymized transcripts, and survey metrics presented in the evaluation of the CoFrame system in Chapter 4, as well as the configuration and output files from Allocobot for both scenarios presented in Chapter 6.

Franka Research 3. https://franka.de/products/franka-research-3.

Van der Aalst, Wil MP. 1998. The application of petri nets to workflow management. *Journal of circuits, systems, and computers* 8(01):21–66.

Aaltonen, Iina, and Timo Salmi. 2019. Experiences and expectations of collaborative robots in industry and academia: Barriers and development needs. *Procedia Manufacturing* 38:1151–1158.

Adam, Nabil R, Vijayalakshmi Atluri, and Wei-Kuang Huang. 1998. Modeling and analysis of workflows using petri nets. *Journal of Intelligent Information Systems* 10:131–158.

Adriaensen, A, F Costantino, G Di Gravio, and R Patriarca. 2022. Teaming with industrial cobots: A socio-technical perspective on safety analysis. *Human Factors and Ergonomics in Manufacturing & Service Industries* 32(2): 173–198.

Akundi, Aditya, Daniel Euresti, Sergio Luna, Wilma Ankobiah, Amit Lopes, and Immanuel Edinbarough. 2022. State of industry 5.0—analysis and identification of current research trends. *Applied System Innovation* 5(1):27.

Alexandrova, Sonya, Zachary Tatlock, and Maya Cakmak. 2015. Roboflow: A flow-based visual programming language for mobile manipulation tasks. In *2015 ieee international conference on robotics and automation (icra)*, 5537–5544. IEEE.

Andrew, Megan, Timothy Marler, Jesse Lastunen, Hannah Acheson-Field, and Steven W Popper. 2020. *An analysis of education and training programs in advanced manufacturing using robotics*. RAND.

Andronas, Dionisis, Angelos Argyrou, Konstantinos Fourtakas, Panagiotis Paraskevopoulos, and Sotiris Makris. 2020. Design of human robot collaboration workstations–two automotive case studies. *Procedia Manufacturing* 52:283–288.

ANSI/RIA R15.06-2012. 2012. Industrial Robots and Robot Systems - Safety Requirements. Standard: ANSI/RIA R15.06-2012.

ANSI/RIA R15.08-1-2020. 2020. Industrial Mobile Robots - Safety Requirements - Part 1: Requirements for the Industrial Mobile Robot. Standard: ANSI/RIA R15.08-1-2020.

Autor, David. 2021. Good news: There's a labor shortage. *The New York Times*.

Bachiller-Burgos, Pilar, Ivan Barbecho, Luis V Calderita, Pablo Bustos, and Luis J Manso. 2020. Learnblock: A robot-agnostic educational programming tool. *IEEE Access* 8:30012–30026.

Bag, Surajit, Gunjan Yadav, Pavitra Dhamija, and Krishan Kumar Kataria. 2021. Key resources for industry 4.0 adoption and its effect on sustainable production and circular economy: An empirical study. *Journal of Cleaner Production* 281:125233.

Balakirsky, Stephen, Zeid Kootbally, Thomas Kramer, Anthony Pietromartire, Craig Schlenoff, and Satyandra Gupta. 2013. Knowledge driven robotics for kitting applications. *Robotics and Autonomous Systems* 61(11): 1205–1214.

Barnes, R.M. 1980. *Motion and time study: Design and measurement of work*. 7th ed. John Wiley & Sons.

Bartneck, Christoph, Marius Soucy, Kevin Fleuret, and Eduardo B Sandoval. 2015. The robot engine—making the unity 3d game engine work for

hri. In *2015 24th ieee international symposium on robot and human interactive communication (ro-man)*, 431–437. IEEE.

Basmajian, J., and De Luca. 1985. *Muscles alive: Their functions revealed by electromyography (5th ed.)*. Williams & Wilkins.

Battini, Daria, Xavier Delorme, Alexandre Dolgui, Alessandro Persona, and Fabio Sgarbossa. 2016. Ergonomics in assembly line balancing based on energy expenditure: a multi-objective model. *International Journal of Production Research* 54(3):824–845.

Berx, Nicole, Wilm Decré, and Liliane Pintelon. 2022. Examining the role of safety in the low adoption rate of collaborative robots. *Procedia CIRP* 106:51–57.

Beyer, Hugh, and Karen Holtzblatt. 1999. Contextual design. *interactions* 6(1):32–42.

Bi, Zhu Ming, Chaomin Luo, Zhonghua Miao, Bing Zhang, WJ Zhang, and Lihui Wang. 2021. Safety assurance mechanisms of collaborative robotic systems in manufacturing. *Robotics and Computer-Integrated Manufacturing* 67:102022.

Bobka, Paul, Tomas Germann, Jakob K Heyn, Roman Gerbers, Franz Dietrich, and Klaus Dröder. 2016. Simulation platform to investigate safe operation of human-robot collaboration systems. *Procedia Cirp* 44: 187–192.

Boucher, Sasha, Margaret Cullen, and Andre Calitz. 2022. Smes' perceptions of the use of ai and cobots. In *2022 international business conference*, 1449. TSHWANE UNIVERSITY OF TECHNOLOGY.

Bounouar, Mouad, Richard Bearee, Ali Siadat, and Tahar-Hakim Benchekroun. 2022. On the role of human operators in the design process of cobotic systems. *Cognition, Technology & Work* 1–17.

Brosque, Cynthia, Elena Galbally, Oussama Khatib, and Martin Fischer. 2020. Human-robot collaboration in construction: Opportunities and challenges. In *2020 international congress on human-computer interaction, optimization and robotic applications (hora)*, 1–8. IEEE.

Buchanan, Bruce G, and Reid G Smith. 1988. Fundamentals of expert systems. *Annual review of computer science* 3(1):23–58.

Byun, Myunghwan, and Jeyoun Dong. 2023. Approaches to usability improvement of teaching for the collaborative robot. In *2023 ieee international conference on software services engineering (sse)*, 1–5. IEEE.

Calzavara, Martina, Maurizio Faccio, and Irene Granata. 2023. Multi-objective task allocation for collaborative robot systems with an industry 5.0 human-centered perspective. *The International Journal of Advanced Manufacturing Technology* 128(1-2):297–314.

Cardoso, André, Ana Colim, Estela Bicho, Ana Cristina Braga, Marino Menozzi, and Pedro Arezes. 2021. Ergonomics and human factors as a requirement to implement safer collaborative robotic workstations: A literature review. *Safety* 7(4):71.

Casalino, Andrea, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco. 2019. Optimal scheduling of human–robot collaborative assembly operations with time petri nets. *IEEE Transactions on Automation Science and Engineering* 18(1):70–84.

Chowdhury, Aparajita, Aino Ahtinen, Roel Pieters, and Kaisa Vaananen. 2020. User experience goals for designing industrial human-cobot collaboration: A case study of franka panda robot. In *Proceedings of the 11th nordic conference on human-computer interaction: Shaping experiences, shaping society*, 1–13.

Chrisinger, David. 2019. The solution lies in education: artificial intelligence & the skills gap. *On the Horizon*.

Christiernin, Linn Gustavsson. 2017. How to describe interaction with a collaborative robot. In *Proceedings of the companion of the 2017 acm/ieee international conference on human-robot interaction*, 93–94.

Chryssolouris, G, D Mavrikios, and L Rentzos. 2016. The teaching factory: A manufacturing education paradigm. *Procedia Cirp* 57:44–48.

Clarke, Victoria, and Virginia Braun. 2017. Thematic analysis. *The journal of positive psychology* 12(3):297–298.

Cohen, Yuval, Shraga Shoval, and Maurizio Faccio. 2019. Strategic view on cobot deployment in assembly 4.0 systems. *IFAC-PapersOnLine* 52(13): 1519–1524.

Colgate, J Edward, Michael Peshkin, and Stephen H Klostermeyer. 2003. Intelligent assist devices in industrial applications: a review. In *Proceedings 2003 ieee/rsj international conference on intelligent robots and systems (iros 2003)(cat. no. 03ch37453)*, vol. 3, 2516–2521. IEEE.

Dagdilelis, Vassilios, Maya Sartatzemi, and Katerina Kagani. 2005. Teaching (with) robots in secondary schools: some new and not-so-new pedagogical problems. In *Fifth ieee international conference on advanced learning technologies (icalt'05)*, 757–761. IEEE.

Dalle Mura, Michela, and Gino Dini. 2022. Job rotation and human–robot collaboration for enhancing ergonomics in assembly lines by a genetic algorithm. *The International Journal of Advanced Manufacturing Technology* 1–14.

Devathasan, Kezia, Celina Berg, and Daniela Damian. 2022. The role of abstraction in introductory programming. In *Proceedings of the 2022 acm sigplan international symposium on splash-e*, 7–13.

Djuric, Ana M, RJ Urbanic, and JL Rickli. 2016. A framework for collaborative robot (cobot) integration in advanced manufacturing systems. *SAE International Journal of Materials and Manufacturing* 9(2):457–464.

Dmytriyev, Yevheniy, Marco Carnevale, Hermes Giberti, and Giovanni Todeschini. 2022. On cobot programming in industrial tasks: A test case. In *2022 international congress on human-computer interaction, optimization and robotic applications (hora)*, 1–9. IEEE.

Dong, Jeyoun, Wookyong Kwon, Dongyeop Kang, and Seung Woo Nam. 2021. A study on the usability evaluation of teaching pendant for manipulator of collaborative robot. In *International conference on human-computer interaction*, 234–238. Springer.

El-Shamouty, Mohamed, Xinyang Wu, Shanqi Yang, Marcel Albus, and Marco F Huber. 2020. Towards safe human-robot collaboration using deep reinforcement learning. In *2020 ieee international conference on robotics and automation (icra)*, 4899–4905. IEEE.

El Zaatari, Shirine, Mohamed Marei, Weidong Li, and Zahid Usman. 2019. Cobot programming for collaborative industrial tasks: An overview. *Robotics and Autonomous Systems* 116:162–180.

Emeric, Colombet, Debled Geoffroy, and Dossou Paul-Eric. 2020. Development of a new robotic programming support system for operators. *Procedia Manufacturing* 51:73–80.

Enrique, Daisy Valle, João Carlos M Druczkoski, Tânia Miranda Lima, and Fernando Charrua-Santos. 2021. Advantages and difficulties of implementing industry 4.0 technologies for labor flexibility. *Procedia Computer Science* 181:347–352.

Ernst, Jette, and Charlotte Jonasson. 2020. Serving robots?–exploring human and robot social dynamics in everyday hospital work. In *36th egos*

colloquium 2020: Organizing for a sustainable future: Responsibility, renewal & resistance.

Faccio, Maurizio, Irene Granata, Alberto Menini, Mattia Milanese, Chiara Rossato, Matteo Bottin, Riccardo Minto, Patrik Pluchino, Luciano Gamberini, Giovanni Boschetti, et al. 2023. Human factors in cobot era: A review of modern production systems features. *Journal of Intelligent Manufacturing* 34(1):85–106.

Fanuc. Fanuc.

Fraser, Neil. 2015. Ten things we've learned from blockly. In *2015 ieee blocks and beyond workshop (blocks and beyond)*, 49–50. IEEE.

Frijns, Helena Anna, and Christina Schmidbauer. 2021. Design guidelines for collaborative industrial robot user interfaces. In *Human-computer interaction–interact 2021: 18th ifip tc 13 international conference, bari, italy, august 30–september 3, 2021, proceedings, part iii 18*, 407–427. Springer.

Galin, Rinat R, and Roman V Meshcheryakov. 2020. Human-robot interaction efficiency and human-robot collaboration. In *Robotics: Industry 4.0 issues & new intelligent control paradigms*, 55–63. Springer.

Gao, Yuxiang, and Chien-Ming Huang. 2019. Pati: a projection-based augmented table-top interface for robot programming. In *Proceedings of the 24th international conference on intelligent user interfaces*, 345–355.

Garg, Arun. 1976. A metabolic rate prediction model for manual materials handling jobs. Ph.D. thesis. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-07-26.

Garg, Arun, Don B. Chaffin, and Gary D. Herrin. 1978. Prediction of metabolic rates for manual materials handling jobs. *American Industrial Hygiene Association Journal* 39(8):661–674.

Garg, Arun, J Steven Moore, and Jay M Kapellusch. 2007. The strain index to analyze jobs for risk of distal upper extremity disorders: Model validation. In *2007 ieee international conference on industrial engineering and engineering management*, 497–499. IEEE.

———. 2017. The revised strain index: an improved upper extremity exposure assessment model. *Ergonomics* 60(7):912–922.

George, Peter, Chi-Tsun Cheng, Toh Yen Pang, and Katrina Neville. 2023. Task complexity and the skills dilemma in the programming and control of collaborative robots for manufacturing. *Applied Sciences* 13(7):4635.

Ghorabaee, Mehdi Keshavarz. 2016. Developing an mcdm method for robot selection with interval type-2 fuzzy sets. *Robotics and Computer-Integrated Manufacturing* 37:221–232.

Giannopoulou, Georgia, Elsi-Mari Borrelli, and Fiona McMaster. 2021. " programming-it's not for normal people": A qualitative study on user-empowering interfaces for programming collaborative robots. In *2021 30th ieee international conference on robot & human interactive communication (ro-man)*, 37–44. IEEE.

Giffi, Craig, Paul Wellener, Ben Dollar, Heather Ashton Manolian, Luke Monck, and Chad Moutray. 2018. Deloitte and the manufacturing institute skills gap and future of work study. *Deloitte Insights*.

Gil-Vilda, Francisco, A Sune, José Antonio Yagüe-Fabra, Carlos Crespo, and Herve Serrano. 2017. Integration of a collaborative robot in a u-shaped production line: a real case study. *Procedia Manufacturing* 13: 109–115.

Goetz, Jennifer, Sara Kiesler, and Aaron Powers. 2003. Matching robot appearance and behavior to tasks to improve human-robot cooperation.

In *The 12th ieee international workshop on robot and human interactive communication, 2003. proceedings. roman 2003.*, 55–60. Ieee.

Golin, Marta, and Christopher Rauh. 2022. The impact of fear of automation.

of Governmental Industrial Hygienist), ACGIH (American Conference. 2002. Threshold limit values for chemical substances and physical agents in the work environment.

Grahn, Sten, and Björn Langbeck. 2014. benefits of collaborative robots in assembly–an evaluation scheme. In *The 6th swedish production symposium*.

Grand View Research. 2023. Collaborative robots market size, share & trends analysis report by payload capacity, by application (assembly, handling, packaging, quality testing), by vertical, by region, and segment forecasts, 2023–2030. Technical Report GVR-1-68038-371-3, Grand View Research.

Grondman, Ivo, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)* 42(6):1291–1307.

Guertler, Matthias, Laura Tomidei, Nathalie Sick, Marc Carmichael, Gavin Paul, Annika Wambsganss, Victor Hernandez Moreno, and Sazzad Hussain. 2023. When is a robot a cobot? moving beyond manufacturing and arm-based cobot manipulators. *Proceedings of the Design Society* 3: 3889–3898.

Guzdial, Mark. 1994. Software-realized scaffolding to facilitate programming for science learning. *Interactive learning environments* 4(1):001–044.

Hentout, Abdelfetah, Mustapha Aouache, Abderraouf Maoudj, and Isma Akli. 2019. Human–robot interaction in industrial collaborative robotics:

a literature review of the decade 2008–2017. *Advanced Robotics* 33(15-16): 764–799.

Holm, Jacob Rubæk, Edward Lorenz, and Jørgen Stamhus. 2021. The impact of robots and ai/ml on skills and work organisation.

Horst, John, Elena Messina, Jeremy Marvel, et al. 2021. Best practices for the integration of collaborative robots into workcells within small and medium-sized manufacturing operations. *National Institute of Standards and Technology Advanced Manufacturing Series* 100–41.

Hu, Bin, and Jing Chen. 2017. Optimal task allocation for human–machine collaborative manufacturing systems. *IEEE Robotics and Automation Letters* 2(4):1933–1940.

Hu, Liang, Zhenyu Liu, Weifei Hu, Yueyang Wang, Jianrong Tan, and Fei Wu. 2020. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. *Journal of Manufacturing Systems* 55:1–14.

Huang, Congfang, Shiyu Zhou, Jingshan Li, and Robert G Radwin. 2023. Allocating robots/cobots to production systems for productivity and ergonomics optimization. *IEEE Transactions on Automation Science and Engineering*.

Huang, Jun, Duc Truong Pham, Ruiya Li, Mo Qu, Yongjing Wang, Mairi Kerin, Shizhong Su, Chunqian Ji, Omar Mahomed, Riham Khalil, et al. 2021. An experimental human-robot collaborative disassembly cell. *Computers & Industrial Engineering* 155:107189.

Huang, Justin, and Maya Cakmak. 2017. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *2017 12th acm/ieee international conference on human-robot interaction* (*hri*, 453–462. IEEE.

Huang, Shengyi, and Santiago Ontañón. 2020. A closer look at invalid action masking in policy gradient algorithms. *arXiv preprint arXiv:2006.14171*.

Ionescu, Tudor B. 2020. Meet your personal cobot, but don't touch it just yet. In *2020 29th ieee international conference on robot and human interactive communication (ro-man)*, 1113–1118. IEEE.

Ionescu, Tudor B, and Sebastian Schlund. 2019. A participatory programming model for democratizing cobot technology in public and industrial fablabs. *Procedia CIRP* 81:93–98.

ISO/TS 15066:2016. 2016. Robots and Robotic Devices. Collaborative Robots. Standard: ISO/TS 15066:2016.

Jacobs, Lindsay. 2024. Collaborative robots in the workplace: Occupational, geographic, and demographic opportunities for technology adoption.

Jamwal, Anbesh, Rajeev Agrawal, Monica Sharma, and Antonio Giallanza. 2021. Industry 4.0 technologies for manufacturing sustainability: A systematic review and future research directions. *Applied Sciences* 11(12): 5725.

Janjanam, Durgaprasad, Bharathi Ganesh, and L Manjunatha. 2021. Design of an expert system architecture: An overview. In *Journal of physics: Conference series*, vol. 1767, 012036. IOP Publishing.

Javaid, Mohd, Abid Haleem, Ravi Pratap Singh, Shanay Rab, and Rajiv Suman. 2022. Significant applications of cobots in the field of manufacturing. *Cognitive Robotics* 2:222–233.

Kaasinen, Eija, Franziska Schmalfuß, Cemalettin Özturk, Susanna Aromaa, Menouer Boubekeur, Juhani Heilala, Päivi Heikkilä, Timo Kuula,

Marja Liinasuo, Sebastian Mach, et al. 2020. Empowering and engaging industrial workers with operator 4.0 solutions. *Computers & Industrial Engineering* 139:105678.

Kadir, Bzhwen A, Ole Broberg, Carolina Souza da Conceição, et al. 2018. Designing human-robot collaborations in industry 4.0: explorative case studies. In *Ds 92: Proceedings of the design 2018 15th international design conference*, 601–610.

Kam, Hyeong Ryeol, Sung-Ho Lee, Taejung Park, and Chang-Hun Kim. 2015. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems* 60:337–345.

Kapinus, Michal, Zdeněk Materna, Daniel Bambušek, Vítězslav Beran, and Pavel Smrž. 2023. Arcor2: Framework for collaborative end-user management of industrial robotic workplaces using augmented reality. *arXiv preprint arXiv:2306.08464*.

Khalid, A, P Kirisci, Z Ghrairi, KD Thoben, and J Pannek. 2017. Towards implementing safety and security concepts for human-robot collaboration in the context of industry 4.0. In *39th international matador conference on advanced manufacturing*, vol. 2, 55–63.

Kheirabadi, Mahboobe, Samira Keivanpour, Yuvin Adnarain Chinniah, and Jean-Marc Frayret. 2023. Human-robot collaboration in assembly line balancing problems: Review and research gaps. *Computers & Industrial Engineering* 186:109737.

Kildal, Johan, Alberto Tellaeche, Izaskun Fernández, and Iñaki Maurtua. 2018. Potential users' key concerns and expectations for the adoption of cobots. *Procedia CIRP* 72:21–26.

Knudsen, Mikkel, and Jari Kaivo-Oja. 2020. Collaborative robots: Frontiers of current literature. *Journal of Intelligent Systems: Theory and Applications* 3(2):13–20.

Konstant, Anna Elizabeth. 2024. Human-robot collaboration in industrial tasks. Ph.D. thesis, ProQuest LLC, Ann Arbor, MI. Available from ProQuest.

Kopp, Tobias, Marco Baumgartner, and Steffen Kinkel. 2021. Success factors for introducing industrial human-robot interaction in practice: an empirically driven framework. *The International Journal of Advanced Manufacturing Technology* 112:685–704.

Kuka. Kuka.

Lappalainen, Inka. 2019. Logistics robots as an enabler of hospital service system renewal? In *The 10 years naples forum on service. service dominant logic, network and systems theory and service science: Integrating three perspectives for a new service agenda. ischia, italy*.

Lasi, Heiner, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. 2014. Industry 4.0. *Business & information systems engineering* 6:239–242.

Leitão, Paulo, Carla AS Geraldes, Florbela P Fernandes, and Hasmik Badikyan. 2020. Analysis of the workforce skills for the factories of the future. In *2020 ieee conference on industrial cyberphysical systems (icps)*, vol. 1, 353–358. IEEE.

Leng, Jiewu, Weinan Sha, Baicun Wang, Pai Zheng, Cunbo Zhuang, Qiang Liu, Thorsten Wuest, Dimitris Mourtzis, and Lihui Wang. 2022. Industry 5.0: Prospect and retrospect. *Journal of Manufacturing Systems* 65:279–295.

Lewandowski, R, and JI Olszewska. 2020. Automated task scheduling for automotive industry. In *2020 ieee 24th international conference on intelligent engineering systems (ines)*, 159–164. IEEE.

Liao, Hao-yu, Yuhao Chen, Boyi Hu, and Sara Behdad. 2023. Optimization-based disassembly sequence planning under uncertainty for human–robot collaboration. *Journal of mechanical design* 145(2):022001.

Liu, Li, Fu Guo, Zishuai Zou, and Vincent G Duffy. 2022a. Application, development and future opportunities of collaborative robots (cobots) in manufacturing: A literature review. *International Journal of Human–Computer Interaction* 1–18.

Liu, Li, Andrew J Schoen, Curt Henrichs, Jingshan Li, Bilge Mutlu, Yajun Zhang, and Robert G Radwin. 2022b. Human robot collaboration for enhancing work activities. *Human Factors* 00187208221077722.

Malik, Ali Ahmad, and Alexander Brem. 2021. Digital twins for collaborative robots: A case study in human-robot interaction. *Robotics and Computer-Integrated Manufacturing* 68:102092.

Malik, Ali Ahmad, Tariq Masood, and Rehana Kousar. 2021. Reconfiguring and ramping-up ventilator production in the face of covid-19: Can robots help? *Journal of Manufacturing Systems* 60:864–875.

Malm, Timo, Timo Salmi, Ilari Marstio, and Iina Aaltonen. 2019. Are collaborative robots safe? In *Automaatiopäivät23*, 110–117. Suomen automaatioseura.

Mancini, Massimiliano, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, and Barbara Caputo. 2018. Kitting in the wild through online domain adaptation. In *2018 ieee/rsj international conference on intelligent robots and systems (iros)*, 1103–1109. IEEE.

Marvel, Jeremy A. 2014. Collaborative robots: A gateway into factory automation. *National Institute of Standards and Technology*.

Massa, Daniele, Massimo Callegari, and Cristina Cristalli. 2015. Manual guidance for industrial robot programming. *Industrial Robot: An International Journal* 42(5):457–465.

Matsas, Elias, and George-Christopher Vosniakos. 2017. Design of a virtual reality training system for human–robot collaboration in manufacturing tasks. *International Journal on Interactive Design and Manufacturing (IJIDeM)* 11(2):139–153.

Mavrikios, Dimitris, Nikolaos Papakostas, Dimitris Mourtzis, and George Chryssolouris. 2013. On industrial learning and training for the factories of the future: a conceptual, cognitive and technology framework. *Journal of Intelligent Manufacturing* 24(3):473–485.

Maw, Isaac. 2018. Time and money: How much do industrial robots cost?

Mayr-Dorn, Christoph, Mario Winterer, Christian Salomon, Doris Hohensinger, and Rudolf Ramler. 2021. Considerations for using block-based languages for industrial robot programming-a case study. In *2021 ieee/acm 3rd international workshop on robotics software engineering (rose)*, 5–12. IEEE.

Michaelis, Joseph E, Amanda Siebert-Evenstone, David Williamson Shaffer, and Bilge Mutlu. 2020. Collaborative or simply uncaged? understanding human-cobot interactions in automation. In *Proceedings of the 2020 chi conference on human factors in computing systems*, 1–12.

Michalos, George, Sotiris Makris, Panagiota Tsarouchi, Toni Guasch, Dimitris Kontovrakis, and George Chryssolouris. 2015. Design considerations for safe human-robot collaborative workplaces. *Procedia CIrP* 37:248–253.

Michel, Olivier. 2004. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems* 1(1):5.

Miller, David. 2021. Robotics Adoption Survey Finds Ups, Downs, and a Few Surprises. Online; accessed August 30, 2021.

Mladenović, Monika, Ivica Boljat, and Žana Žanko. 2018. Comparing loops misconceptions in block-based and text-based programming languages at the k-12 level. *Education and Information Technologies* 23:1483–1500.

Modares, Hamidreza, Isura Ranatunga, Frank L Lewis, and Dan O Popa. 2015. Optimized assistive human–robot interaction using reinforcement learning. *IEEE transactions on cybernetics* 46(3):655–667.

Monguzzi, Andrea, Mahmoud Badawi, Andrea Maria Zanchettin, and Paolo Rocco. 2022. A mixed capability-based and optimization methodology for human-robot task allocation and scheduling. In *2022 31st ieee international conference on robot and human interactive communication (ro-man)*, 1271–1276. IEEE.

Munzer, Thibaut, Marc Toussaint, and Manuel Lopes. 2018. Efficient behavior learning in human–robot collaboration. *Autonomous Robots* 42: 1103–1115.

Nahavandi, Saeid. 2019. Industry 5.0—a human-centric solution. *Sustainability* 11(16):4371.

Paliga, Mateusz. 2022. Human–cobot interaction fluency and cobot operators' job performance. the mediating role of work engagement: A survey. *Robotics and Autonomous Systems* 155:104191.

Pauliková, Alena, Zdenka Gyurák Babel'ová, and Monika Ubárová. 2021. Analysis of the impact of human–cobot collaborative manufacturing implementation on the occupational health and safety and the quality requirements. *International Journal of Environmental Research and Public Health* 18(4):1927.

Pearce, Margaret, Bilge Mutlu, Julie Shah, and Robert Radwin. 2018. Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes. *IEEE transactions on automation science and engineering* 15(4):1772–1784.

Perzylo, Alexander, Nikhil Somani, Stefan Profanter, Ingmar Kessler, Markus Rickert, and Alois Knoll. 2016. Intuitive instruction of industrial robots: Semantic process descriptions for small lot production. In *2016 ieee/rsj international conference on intelligent robots and systems (iros)*, 2293–2300. IEEE.

Peshkin, Michael, and J Edward Colgate. 1999. Cobots. *Industrial Robot: An International Journal* 26(5):335–341.

Peterson, James L. 1977. Petri nets. *ACM Computing Surveys (CSUR)* 9(3): 223–252.

Pieskä, Sakari, Jari Kaarela, and Jari Mäkelä. 2018. Simulation and programming experiences of collaborative robots for small-scale manufacturing. In *2018 2nd international symposium on small-scale intelligent manufacturing systems (sims)*, 1–4. IEEE.

Pollak, Anita, Mateusz Paliga, Matias M Pulopulos, Barbara Kozusznik, and Malgorzata W Kozusznik. 2020. Stress in manual and autonomous modes of collaboration with a cobot. *Computers in Human Behavior* 112: 106469.

Potvin, Jim R. 2012. Predicting maximum acceptable efforts for repetitive tasks: an equation based on duty cycle. *Human factors* 54(2):175–188.

Qin, Jian, Ying Liu, and Roger Grosvenor. 2016. A categorical framework of manufacturing for industry 4.0 and beyond. *Procedia cirp* 52:173–178.

Raatz, Annika, Sebastian Blankemeyer, Tobias Recker, Dennis Pischke, and Peter Nyhuis. 2020. Task scheduling method for hrc workplaces

based on capabilities and execution time assumptions for robots. *CIRP Annals* 69(1):13–16.

Raffin, Antonin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research* 22(268):1–8.

Ras, Eric, Fridolin Wild, Christoph Stahl, and Alexandre Baudet. 2017. Bridging the skills gap of workers in industry 4.0 by human performance augmentation tools: Challenges and roadmap. In *Proceedings of the 10th international conference on pervasive technologies related to assistive environments*, 428–432.

Resnick, Mitchel, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Communications of the ACM* 52(11):60–67.

Rivera-Pinto, Andoni, Johan Kildal, and Elena Lazkano. 2023. Toward programming a collaborative robot by interacting with its digital twin in a mixed reality environment. *International Journal of Human–Computer Interaction* 1–13.

Robots, Universal. Ur10 technical specifications. `https://www.universal-robots.com/media/50895/ur10_en.pdf`.

———. 2023. Media kit. `https://www.universal-robots.com/media/1828242/07_2023_ur_media_kit.pdf`.

Rozier, Kristin Y. 2011. Linear temporal logic symbolic model checking. *Computer Science Review* 5(2):163–203.

Sadik, Ahmed R, and Bodo Urban. 2017. Flow shop scheduling problem and solution in cooperative robotics—case-study: One cobot in cooperation with one worker. *Future Internet* 9(3):48.

Salvatore, Miranda, Riemma Stefano, et al. 2021. Smart operators: How industry 4.0 is affecting the worker's performance in manufacturing contexts. *Procedia Computer Science* 180:958–967.

Sanders, David Adrian, Alexander Gegov, and David Ndzi. 2018. Knowledge-based expert system using a set of rules to assist a teleoperated mobile robot. In *Intelligent systems and applications: Extended and selected results from the sai intelligent systems conference (intellisys) 2016*, 371–392. Springer.

Sanneman, Lindsay, Christopher Fourie, Julie A Shah, et al. 2021. The state of industrial robotics: Emerging technologies, challenges, and key research directions. *Foundations and Trends® in Robotics* 8(3):225–306.

Sauppé, Allison, and Bilge Mutlu. 2015. The social impact of a robot co-worker in industrial settings. In *Proceedings of the 33rd annual acm conference on human factors in computing systems*, 3613–3622.

Schoen, Andrew, Curt Henrichs, Mathias Strohkirch, and Bilge Mutlu. 2020. Authr: A task authoring environment for human-robot teams. In *Proceedings of the 33rd annual acm symposium on user interface software and technology*, 1194–1208.

Schoen, Andrew, and Bilge Mutlu. 2024. Openvp: A customizable visual programming environment for robotics applications. In *Proceedings of the 2024 acm/ieee international conference on human-robot interaction*, 944–948.

Schoen, Andrew, Nathan White, Curt Henrichs, Amanda Siebert-Evenstone, David Shaffer, and Bilge Mutlu. 2022. Coframe: A system for

training novice cabot programmers. In *2022 17th acm/ieee international conference on human-robot interaction (hri)*, 185–194. IEEE.

Schoen, Andrew J. 2023. *Representations, tools and interfaces for improving expert design of collaborative human-robot interactions*. The University of Wisconsin-Madison.

Schou, Casper, Jens S Damgaard, Simon Bøgh, and Ole Madsen. 2013. Human-robot interface for instructing industrial tasks using kinesthetic teaching. In *Ieee isr 2013*, 1–6. IEEE.

Schrepp, Martin, Andreas Hinderks, and Jörg Thomaschewski. 2014. Applying the user experience questionnaire (ueq) in different evaluation scenarios. In *Design, user experience, and usability. theories, methods, and tools for designing the user experience: Third international conference, duxu 2014, held as part of hci international 2014, heraklion, crete, greece, june 22-27, 2014, proceedings, part i 3*, 383–392. Springer.

Schrepp, Martin, Andreas Hinderks, and Jorg Thomaschewski. 2017. Construction of a benchmark for the user experience questionnaire (ueq). *International Journal of Interactive Multimedia and Artificial Intelligence* 4(4): 40–44.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Senft, Emmanuel, Michael Hagenow, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. 2021a. Situated live programming

for human-robot collaboration. In *The 34th annual acm symposium on user interface software and technology*, 613–625.

Senft, Emmanuel, Michael Hagenow, Kevin Welsh, Robert Radwin, Michael Zinn, Michael Gleicher, and Bilge Mutlu. 2021b. Task-level authoring for remote robot teleoperation. *Frontiers in Robotics & AI*.

Shi, Jane, Glenn Jimmerson, Tom Pearson, and Roland Menassa. 2012. Levels of human and robot collaboration for automotive manufacturing. In *Proceedings of the workshop on performance metrics for intelligent systems*, 95–100.

Shipps, Belinda P, and Robert L Howard. 2013. Unemployment and job creation programs: Is there a skills gap. *Review of Business* 33(2):103–118.

Shmatko, Natalia, and Galina Volkova. 2020. Bridging the skill gap in robotics: Global and national environment. *SAGE Open* 10(3): 2158244020958736.

Siebert-Evenstone, Amanda, Joseph E Michaelis, David Williamson Shaffer, and Bilge Mutlu. 2021. Safety first: Developing a model of expertise in collaborative robotics. In *Advances in quantitative ethnography: Second international conference, icqe 2020, malibu, ca, usa, february 1-3, 2021, proceedings 2*, 304–318. Springer.

Silva, Andreia, Ana Correia Simões, and Renata Blanc. 2022. Criteria to consider in a decision model for collaborative robot (cobot) adoption: A literature review. In *2022 ieee 20th international conference on industrial informatics (indin)*, 477–482. IEEE.

Simões, Ana C, António Lucas Soares, and Ana C Barros. 2019. Drivers impacting cobots adoption in manufacturing context: A qualitative study. In *Advances in manufacturing ii: Volume 1-solutions for industry 4.0*, 203–212. Springer.

Simões, Ana Correia, Ana Pinto, Joana Santos, Sofia Pinheiro, and David Romero. 2022. Designing human-robot collaboration (hrc) workspaces in industrial settings: A systematic literature review. *Journal of Manufacturing Systems* 62:28–43.

Simões, Ana Correia, António Lucas Soares, and Ana Cristina Barros. 2020. Factors influencing the intention of managers to adopt collaborative robots (cobots) in manufacturing organizations. *Journal of engineering and technology management* 57:101574.

Słowikowski, Marcin, Zbigniew Pilat, Michał Smater, and Jacek Zieliński. 2018. Collaborative learning environment in vocational education. In *Aip conference proceedings*, vol. 2029, 020070. AIP Publishing LLC.

Souravlas, Stavros, Stefanos Katsavounis, and Sofia Anastasiadou. 2020. On modeling and simulation of resource allocation policies in cloud computing using colored petri nets. *Applied Sciences* 10(16):5644.

Spencer, David A. 2018. Fear and hope in an age of mass automation: debating the future of work. *New Technology, Work and Employment* 33(1): 1–12.

Stanton, Neville A. 2006. Hierarchical task analysis: Developments, applications, and extensions. *Applied ergonomics* 37(1):55–79.

Steinmetz, Franz, Verena Nitsch, and Freek Stulp. 2019. Intuitive task-level programming by demonstration through semantic skill recognition. *IEEE Robotics and Automation Letters* 4(4):3742–3749.

Steinmetz, Franz, Annika Wollschläger, and Roman Weitschat. 2018. Razer—a hri for visual task-level programming and intuitive skill parameterization. *IEEE Robotics and Automation Letters* 3(3):1362–1369.

Sullivan, Dakota, Nathan Thomas White, Andrew Schoen, and Bilge Mutlu. 2024. Making informed decisions: Supporting cobot integration

considering business and worker preferences. In *Proceedings of the 2024 acm/ieee international conference on human-robot interaction*, 706–714.

Sun, Ping, and Changjun Jiang. 2009. Analysis of workflow dynamic changes based on petri net. *Information and Software Technology* 51(2): 284–292.

Sutton, Richard S, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12.

Tamas, Levente, and Mircea Murar. 2019. Smart cps: vertical integration overview and user story with a cobot. *International Journal of Computer Integrated Manufacturing* 32(4-5):504–521.

THOMAS R. WATERS, ARUN GARG, VERN PUTZ-ANDERSON, and LAWRENCE J. FINE. 1993. Revised niosh equation for the design and evaluation of manual lifting tasks. *Ergonomics* 36(7):749–776.

Tsarouchi, Panagiota, George Michalos, Sotiris Makris, Thanasis Athanasatos, Konstantinos Dimoulas, and George Chryssolouris. 2017. On a human–robot workplace design and task allocation system. *International Journal of Computer Integrated Manufacturing* 30(12):1272–1279.

Universal Robots. UniversalRobots.

Vysocky, Ales, and Petr Novak. 2016. Human-robot collaboration in industry. *MM Science Journal* 9(2):903–906.

Waite, Jane Lisa, Paul Curzon, William Marsh, Sue Sentance, and Alex Hadwen-Bennett. 2018. Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal of Computer Science Education in Schools* 2(1):14–40.

Wallace, Jamie. 2021. Getting collaborative robots to work: A study of ethics emerging during the implementation of cobots. *Paladyn, Journal of Behavioral Robotics* 12(1):299–309.

Wang, Juan, and Di Li. 2019. Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. *Sensors* 19(5): 1023.

Waters, TR, V Putz-Anderson, and A Garg. 1994. Applications manual for the revised niosh lifting equation. Tech. Rep., National Institue for Occupational Safety and Health, DHHS (NIOSH).

Weintrop, David, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David C Shepherd, and Diana Franklin. 2018. Evaluating coblox: A comparative study of robotics programming environments for adult novices. In *Proceedings of the 2018 chi conference on human factors in computing systems*, 1–12.

Weintrop, David, David C Shepherd, Patrick Francis, and Diana Franklin. 2017. Blockly goes to work: Block-based programming for industrial robots. In *2017 ieee blocks and beyond workshop (b&b)*, 29–36. IEEE.

Weintrop, David, and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children*, 199–208.

———. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18(1):1–25.

Weiss, Astrid, Ann-Kathrin Wortmeier, and Bettina Kubicek. 2021. Cobots in industry 4.0: A roadmap for future practice studies on human–robot collaboration. *IEEE Transactions on Human-Machine Systems* 51(4):335–345.

White, Nathan Thomas, Andy Schoen, Anna Konstant, Josiah Hanna, Robert Radwin, and Bilge Mutlu. 2025a. Allocobot: A framework and pipeline for flexible human-robot collaborative task allocation. Working title, authorship not finalized, manuscript in preparation.

White, Nathan Thomas, Andy Schoen, Dakota Sullivan, Curt Henrichs, Amanda Siebert-Evenstone, David Shaffer, and Bilge Mutlu. 2025b. Coframe: A system for training novice cobot programmers. Working title, authorship not finalized, manuscript in preparation.

Wing, Jeannette M. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366(1881):3717–3725.

Wingard, Jason, and Christine Farrugia. 2021. *The great skills gap: Optimizing talent for the future of work*. Stanford University Press.

Wojtynek, Michael, Jochen Jakob Steil, and Sebastian Wrede. 2019. Plug, plan and produce as enabler for easy workcell setup and collaborative robot programming in smart factories. *KI-Künstliche Intelligenz* 33(2): 151–161.

Wurhofer, Daniela, Thomas Meneweger, Verena Fuchsberger, and Manfred Tscheligi. 2015. Deploying robots in a production environment: A study on temporal transitions of workers' experiences. In *Human-computer interaction–interact 2015: 15th ifip tc 13 international conference, bamberg, germany, september 14-18, 2015, proceedings, part iii 15*, 203–220. Springer.

Xu, Xun, Yuqian Lu, Birgit Vogel-Heuser, and Lihui Wang. 2021. Industry 4.0 and industry 5.0—inception, conception and perception. *Journal of manufacturing systems* 61:530–535.

Yin, Luxiu, Juan Luo, and Haibo Luo. 2018. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Transactions on Industrial Informatics* 14(10):4712–4721.

Zhang, Rong, Qibing Lv, Jie Li, Jinsong Bao, Tianyuan Liu, and Shimin Liu. 2022. A reinforcement learning method for human-robot collaboration in assembly tasks. *Robotics and Computer-Integrated Manufacturing* 73:102227.

Ziaeefard, Saeedeh, Michele H Miller, Mo Rastgaar, and Nina Mahmoudian. 2017. Co-robotics hands-on activities: A gateway to engineering design and stem learning. *Robotics and Autonomous Systems* 97:40–50.

Zieliński, Krzysztof, Krzysztof Walas, Juan Heredia, and Mikkel Baun Kjærgaard. 2021. A study of cobot practitioners needs for augmented reality interfaces in the context of current technologies. In *2021 30th ieee international conference on robot & human interactive communication (ro-man)*, 292–298. IEEE.

Ziparo, Vittorio A, Luca Iocchi, Pedro U Lima, Daniele Nardi, and Pier Francesco Palamara. 2011. Petri net plans: A framework for collaboration and coordination in multi-robot systems. *Autonomous Agents and Multi-Agent Systems* 23:344–383.

Zuberek, Wlodek M. 1991. Timed petri nets definitions, properties, and applications. *Microelectronics Reliability* 31(4):627–644.

Zurawski, Richard, and MengChu Zhou. 1994. Petri nets and industrial applications: A tutorial. *IEEE Transactions on industrial electronics* 41(6): 567–583.