The Pennsylvania State University

The Graduate School

# UNDERSTANDING PAIN POINTS OF ACT-R MODELERS: A HUMAN-CENTERED APPROACH

A Dissertation in

Informatics

by

Shan Wang



Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2024

The dissertation of Shan Wang was reviewed and approved by the following:

Frank Ritter
Professor of Information Sciences and Technology
Dissertation Advisor
Chair of Committee

Xiaolong (Luke) Zhang
Associate Professor of Information Sciences and Technology

Aiping Xiong
Associate Professor of Information Sciences and Technology

Farnaz Tehranchi
Assistant Professor of Engineering Design and Innovation

Carleen Maitland
Professor of Information Sciences and Technology
Program Head

## ABSTRACT

Among the approaches in artificial intelligence, a classical one is to understand the human mind and, therefore, simulate human cognition and behavior. ACT-R, an example of a cognitive architecture, is a unified theory of cognition realized as a computer program. However, it is relatively difficult to learn, and its associated tools are not only challenging to use but also have limited applicability in various contexts. Additionally, concerns beyond the usability of ACT-R tools appear to exist within the broader research community.

To gain a comprehensive understanding of these challenges, semi-structured interviews were conducted with ACT-R modelers (N=15) to explore their experiences with ACT-R. Specifically, the focus was on examining their pain points, the concerns of the broader research community, and their suggested solutions to the issues they encountered. As an example of an approach to simplify model-building, TAKLML (Task Analysis with Keystroke-Level Modeling and Learning) was introduced and applied. Two example applications of this approach to complex tasks are presented, demonstrating how it not only facilitates cognitive model building but also addresses challenging tasks within the field.

In general, this dissertation contributes to the ACT-R community, cognitive architecture users, the HCI community, and the education community. It provides a relatively comprehensive framework with details for understanding the pain points in building ACT-R models, which can be used to enhance the usability of such models. Additionally, a potential partial solution to these challenges is proposed. This dissertation encourages further discussion regarding the gap between academic research and practical applications within the fields of Human-Computer Interaction and Human Factors. It also contributes to the field of education by emphasizing the importance of respecting diversity and individual differences with an open and growth mindset, so that educators do not “correct” students when they are applying different approaches, whether in problem-solving or learning.

# TABLE OF CONTENTS

**LIST OF FIGURES** ..... **VI**

**LIST OF TABLES** ..... **IX**

**ACKNOWLEDGEMENTS** ..... **X**

**CHAPTER 1 INTRODUCTION** ..... **1**
- Motivation ..... 1
- Research Question and Approaches ..... 3
- Contributions ..... 4
  - To Human-Computer Interaction ..... 4
  - To Cognitive Modeling ..... 4
  - To Education ..... 5
- Dissertation Overview ..... 6

**CHAPTER 2 LITERATURE REVIEW** ..... **7**
- Introduction ..... 7
- An Overview of Cognitive Modeling ..... 7
  - Cognitive Modeling ..... 7
  - ACT-R ..... 9
- Research Efforts on Enhancing ACT-R Model Usability ..... 11
- Similar Tools and Their Enhancements ..... 12
- Summary ..... 14

**CHAPTER 3 AN INTERVIEW STUDY OF MODELERS** ..... **16**
- Introduction ..... 16
- Method ..... 17
  - Recruitment and Participants ..... 17
  - Procedures and Questions ..... 20
  - Data Analysis ..... 22
- Results ..... 25
  - General Observation ..... 25
  - Pain Points of User Journey ..... 32
  - Reasons for Joining and Leaving ..... 45
- Discussion and Implications ..... 47
  - Suggestions ..... 47
  - Limitations ..... 52
  - Future Work ..... 52
  - The TAKLML Approach as an Example Solution ..... 53
- Summary ..... 54

**CHAPTER 4 MODELING WITH THE TAKLML APPROACH** ..... **55**
- Introduction ..... 55
  - The MENDS Task ..... 56
  - Human Performance Data ..... 59

Modeling the MENDS Task ..... 60
Descriptions of the Current Four Strategies ..... 60
How the Models Predict Time ..... 66
How the Models Learn ..... 67
Comparison of Four Strategies ..... 68
Comparing Human Performance and Model Predictions ..... 70
Without Learning ..... 71
With Learning ..... 71
Summary of Comparison ..... 73
Discussion and Conclusion ..... 74
Discussion for the MENDS Task ..... 74
Discussion for Modeling with TAKLML ..... 75
Summary ..... 76

**CHAPTER 5 MODELING ERRORS WITH TAKLML ..... 77**
Introduction ..... 77
Task Description ..... 78
Data of Human Performance Errors ..... 82
Analysis of Human Performance Errors ..... 83
Error Analysis for Each Task ..... 83
Error Analysis for Each Participant in the Test Session ..... 90
Modeling Human Errors ..... 92
Comparison of the Error Model, the Old Model, and Human Data ..... 94
Discussion and Conclusion ..... 95
Discussion of the Error Model ..... 96
Discussion for Modeling with TAKLML ..... 97
Summary ..... 98

**CHAPTER 6 DISCUSSION AND FUTURE WORK ..... 99**
Introduction ..... 99
The TAKLML Approach ..... 99
Extra Pain Points ..... 100
Discussion of Dilemmas ..... 101
Why Care About Pain Points ..... 101
Not Easy to Change ..... 102
No Need to Panic ..... 104
Summary of Limitations ..... 106
Future Work ..... 106
Summary ..... 108

**REFERENCES ..... 109**

**APPENDIX A INTERVIEW GUIDE ..... 115**

**APPENDIX B A SAMPLE OF CODED TRANSCRIPTS ..... 118**

**APPENDIX C PARTIAL PYTHON CODE ..... 120**

# LIST OF FIGURES

Figure **2-1**: *A Framework Process of a Cognitive Modeling Study (Oyewole, Farde, Haight, & Okareh, 2011).* ....8

Figure **2-2**: *Schematic Diagram of the ACT-R Cognitive Architecture* (Ritter et al., 2019). ....10

Figure **2-3**: *Screenshot of Warning Message in IntelliJ for Java.* ....14

Figure **3-1**: *Distribution of Current Locations of Participants*. ....18

Figure **3-2**: *Distribution of Current Roles of Participants*. ....19

Figure **3-3**: *Distribution of Academic Background of Participants*....20

Figure **3-4**: *Screenshot of Tutorial, Downloaded as Part of the Standalone ACT-R7 Application*. ....27

Figure **3-5**: *Screenshot of the ACT-R Website*. ....28

Figure **3-6**: *Graphical User Interface of the Standalone ACT-R Application*. ....31

Figure **3-7**: *Material-related Challenges.* ....33

Figure **3-8**: *Coding-related Challenges.* ....36

Figure **3-9**: *Community-related Challenges*. ....41

Figure **3-10**: *Why People Learn/Join/Rejoin*. ....46

Figure **3-11**: *Why People Distance from ACT-R.* ....47

Figure **4-1**: *The Schematic for the Ben Franklin Radar*. Blue lines are power; red lines are signal; purple lines are both....57

Figure **4-2**: *The MENDS Simulator's Front Panel and One Subsystem*....58

Figure **4-3**: *A Flowchart for the Simple Task Model*. An active path refers to the components receiving power and that are supposed to have lights on. ....60

Figure **4-4**: *A Flowchart of the Grey Upstream Strategy*. ....62

Figure **4-5**: *A Flowchart of the One-by-One First Grey Strategy*. ....63

Figure **4-6**: *A Flowchart of the All-Trays Strategy*....64

Figure **4-7**: *A Flowchart of the Smaller-Lights Strategy*. .....65

Figure **4-8**: *The Pseudocode of the Smaller-Lights Strategy(Screenshot from Code Comments)*..66

Figure **4-9**: *Strategy Predictions for the 35 Faults Without Learning Across Problems*. The x-axis lists all the 35 faults (not in completion order of any session). Blue line indicates prediction of the Grey Upstream Strategy; orange line indicates prediction of the One-by-One First Grey Strategy; green line indicates prediction of the All-Trays Strategy; red line indicates prediction of the Smaller Lights Strategy. .....69

Figure **4-10**: *Strategy Predictions for the 20 Problems Without Learning in the Test Session*. The x-axis shows the same order of the 20 tasks in the test session. Blue line indicates prediction of the Grey Upstream Strategy; orange line indicates prediction of the One-by-One First Grey Strategy; green line indicates prediction of the All-Trays Strategy; red line indicates prediction of the Smaller Lights Strategy. .....70

Figure **4-11**: *An Example Participant (PID 413).* A comparison of human data, the Grey Upstream Strategy model with learning, and the Grey Upstream Strategy model without learning. Blue line indicates prediction of the no-learning model; black line indicates prediction of the learning model; red line indicates observed human performance. .....73

Figure **5-1**: *The Ben Franklin Radar Schematic*. Circles and their colors are explained in the results section. .....79

Figure **5-2**: *MENDS Interface Screenshots of the Replacement Process, without Misreplacements*. .....81

Figure **5-3**: *Total Number of Broken Components Not Fixed by All Participants for the 20 Tasks of the Test Session*. Red horizontal line and bars indicates difficulty level 1; purple horizontal line and bars indicate difficulty level 2; blue horizontal line and bars indicates difficulty level 3; black horizontal line (overlapped with x-axis) indicates difficulty level 4. .....84

Figure **5-4**: *Total Number of Wrong Replacements by All Participants for the 20 Tasks*. Level 3 and 4 are relatively hard to see in this figure so no horizontal lines. Red horizontal line and bars indicate difficulty level 1; purple horizontal line and bars indicates difficulty level 2; blue bars indicate difficulty level 3; black bar indicates difficulty level 4.....85

Figure **5-5**: *Tasks in the Order of the Test Session and Their Corresponding Misreplacements with Frequencies.* The darker the cell color, the more frequently they got misreplaced. The numbers in the cell are their exact frequencies. Lines 2, 3, 6, 14, and 20 are enclosed in squares to assist in their identification.....87

Figure **5-6**: *MENDS Interface Screenshots of All the Six Trays*, with high frequently misreplaced components circled. Abbreviations are provided below each component in cases where the screenshots may be unclear. .....88

Figure **5-7**: *Total Number of Tasks in the Test Session, for Each Participant (PID)*. Blue refers to fixed number; orange refers to not fixed with the specific number at the top of the figure…91

Figure **5-8**: *Replacements in the Test Session for Each Participant (PID).* Blue bars represent the correct replacement; orange bars show the wrong replacement. .....91

Figure **5-9**: *Error to Be Modeled*. For task S2M (red solid squared), PID ignored switch information and chose S1M (red dotted squared) that had switch off (orange circled). .....93

Figure **5-10**: *Comparison of Response Times for 20 Tasks in the Test Session of the Original All-Tray Strategy Model, the Error Model, and Participant 421's Performance*. .....94

Figure **6-1**: *Why Care About Pain Points.* .....101

Figure **6-2**: *Not Easy to Change.* .....103

Figure **6-3**: *No Need to Panic.* .....105

# LIST OF TABLES

Table **2-1**: *Potential Ways to Make a System Like ACT-R Easier to Learn and Use*.....13

Table **3-1**: *Descriptions of the Themes and Codes.*.....24

Table **3-2**: *Learning Resources Mentioned by Participants* (not by any order).....26

Table **3-3**: *Material-related Suggestions.*.....48

Table **3-4**: *Coding-related Suggestions*.....49

Table **3-5**: *Community-related Suggestions.*.....51

Table **4-1**: *Time Parameters Used in the Models.*.....67

Table **4-2**: *Correlations for Participants and Strategies Without Learning*. Example well-fit participants, the four strategies, and their corresponding R2. Those with p-value <.05 have a * attached. Negative correlations are shown as 0's for clarity.....71

Table **4-3**: *Correlations for Participants and Strategies With Learning.* Example well-fit participants, the four strategies, and their corresponding R2. Those with p-value <.05 have a * attached.....72

Table **5-1**: *Comparison of the Error Model and Original Model*. Both have p-values less than .05. .....95

# ACKNOWLEDGEMENTS

Throughout this PhD journey, I have been supported by many individuals and institutions. First and foremost, I thank my advisor, Dr. Frank Ritter, for his patience, guidance, and unwavering support. His mentorship has profoundly impacted both my academic and personal growth. I am also grateful to my committee members, Dr. Xiaolong Zhang, Dr. Aiping Xiong, and Dr. Farnaz Tehranchi, for their invaluable insights and expertise. I really appreciate their help and support during my difficult times.

I would also like to thank Dr. Chris Dancy for his financial support and for introducing me to thought-provoking research topics, including DEI (Diversity, Equity, and Inclusion) of AI. His guidance broadened my perspective on AI and enriched my understanding of its multifaceted impact. Additionally, I am thankful to the College of Information Sciences and Technology for their financial support, as well as to the professors and staff in the college for their support and understanding. The opportunities to engage with faculty in both research and teaching have been instrumental in helping me discern the essence of these fields beyond diversity and improve my communication skills.

To my lab mates in the ACS lab (Applied Cognitive Science Lab) and THICC lab (the Human in Computing and Cognition lab), as well as other friends from the college, thank you for your constructive feedback on my work, and assistance beyond research—whether offering a ride to a conference or providing essential moral support.

I am profoundly grateful to my family for their patience and support, particularly as an international, first-generation college and graduate student navigating cultural and academic differences. Their utmost support has sustained me through challenging times.

I would also like to acknowledge the internet for broadening my horizons and enabling me to learn from brilliant minds around the world. Finally, I thank myself for my perseverance and resilience in moments of self-doubt and uncertainty. Navigating the decision-making process

of determining what aligns with my values and aspirations has been a significant part of my journey. The PhD experience has equipped me with a deeper understanding of research, technology, and humanity, as well as myself, and I am eager to embark on the next chapter with a commitment to making a positive impact and realizing my fullest potential.

Again, to everyone who supported me, named and unnamed (personal connections, to protect their privacy), my heartfelt thanks. This PhD journey has been transformative, and a new chapter awaits.

# Chapter 1

## Introduction

Some research communities today seem to face challenges in sustainability, with declining student engagement and fewer new members joining. Limited access to resources and the steep learning curves associated with specialized tools may further contribute to these barriers. For example, cognitive architectures like ACT-R, though valuable for modeling human cognition, seem challenging to learn and apply effectively, requiring significant time and effort from researchers. These observations require additional investigation to improve their accuracy.

Research across various communities is motivated by diverse goals, and in the case of the ACT-R community, the primary mission is to advance understanding and simulation of human cognition and behavior through programming. People may approach cognition from different theoretical perspectives, and ACT-R represents one approach within the broader scientific exploration of human thought. While this dissertation does not seek to argue whether ACT-R is superior to other cognitive modeling approaches, it eases the way for future researchers interested in ACT-R and other similar architectures/tools, addressing practical obstacles and contributing insights for sustaining the research community. Additionally, the findings here may offer lessons for other modeling architectures, of which many exist (Kotseruba & Tsotsos, 2020; Ritter & Larkin, 1994).

### Motivation

Not all valuable research problems are documented in the existing literature. The focus on theory generation in academia can sometimes lead to gaps between research and practical applications,

especially within the field of Human-Computer Interaction (HCI) (Colusso et al., 2019). Problems that practitioners experience but that have not yet been formalized in the literature are still worthy of exploration. The motivation for this dissertation stems from conversations with colleagues and senior ACT-R researchers that brought attention to some challenges: the difficulties of learning and building ACT-R models and concerns about the sustainability of this research community. Understanding these challenges, or "pain points," could lead to solutions that are beneficial and enduring.

Improving the accessibility and usability of ACT-R tools could benefit current researchers and attract new ones. Lowering the barriers to entry, much like crowdsourcing platforms do, might yield innovative and unexpected contributions to the field. For instance, Amazon Mechanical Turk has enabled new approaches to knowledge-sharing and data collection by making contributions accessible to a broader audience. By making ACT-R more approachable, this research may enable scientists and HCI researchers to incorporate computational models of cognition into their work. The project tackles challenges that have previously been too complex to address, thus opening a "blue ocean" of research opportunities.

We can draw parallels from the evolution of computing. Early computers were used solely by experts for calculations, but over time, they became personal devices that anyone could operate, giving rise to the field of HCI. Decades ago, only experts could use a computer with a high level of precision and care, as these machines were large, complex, and manually operated. Today, digital devices are ubiquitous, with children and marginalized populations alike benefiting from technology advancements. This shift underscores how lowering the learning curve has broadened access and enabled new applications across age groups and abilities. With the HCI field's emphasis on accessibility, diverse users can now engage with technology in meaningful ways.

Currently, ACT-R, like those early computers, remains the domain of trained experts with limited applications. What could happen if ACT-R, and perhaps other cognitive architectures, became easier to learn and apply? This future potential is both intriguing and worth pursuing.

Counterintuitively, researchers themselves are often overlooked as a group whose needs deserve attention. Improving usability and accessibility for them is essential; researchers should not have to endure unnecessary obstacles in their work to make scientific contributions.

The objective of this dissertation is to highlight emerging research areas that are often overlooked and encourage more students and researchers to engage with computational cognitive models. More specifically, it seeks to comprehensively understand the challenges faced by ACT-R modelers and contribute to discussion, as well as insights to support the sustainability and growth of this research community.

## Research Question and Approaches

To address the objective of this dissertation, the core research question is as follows:

**What are the pain points experienced by ACT-R modelers, and what steps can we take to alleviate these challenges?**

To explore this question, semi-structured interviews were conducted with ACT-R modelers worldwide, focusing on their experiences learning and using ACT-R, as well as their broader concerns for the sustainability of the ACT-R research community. Based on insights gathered, as an example, a potential solution was introduced, and its effectiveness was demonstrated through two cognitive modeling case studies. These studies illustrate how the proposed solution can support researchers in overcoming identified obstacles and can contribute to a more accessible and sustainable research environment.

## Contributions

This section outlines the expected contributions of this research to the fields of Human-Computer Interaction (HCI), Cognitive Modeling, and Education.

### To Human-Computer Interaction

This research emphasizes the importance of addressing often-overlooked populations, particularly researchers themselves. It represents an initial comprehensive effort to understand the pain points and concerns within the broader research ecosystem. By highlighting the gap between academic research and practical applications, this dissertation encourages ongoing dialogue in both the Human-Computer Interaction and Cognitive Modeling fields, fostering collaboration and innovation.

Specifically, the semi-structured interview results in Chapter 3 reveal numerous design and redesign opportunities to address the challenges currently faced by modelers and the broader community.

### To Cognitive Modeling

The dissertation introduces a novel approach to modeling through the framework of TAKLML (Task Analysis with Keystroke-Level Modeling and Learning), supported by two performance demonstrations. It generates multiple cognitive models that incorporate diverse strategies, learning processes. Modeling human errors is considered challenging within the field. In addition to the TAKLML methodology, one of the demonstrations presents a technique for simulating human performance errors, contributing to the development of error models.

By using the ACT-R community as a case study, this research outlines directions for fostering a sustainable research environment. It identifies pain points across three major themes: learning materials, coding challenges, and community-related issues. The insights gained, along with suggestions from participants to mitigate these obstacles, provide a framework with details for community members to discuss and address these challenges. This framework applies not only to ACT-R but also to other cognitive architectures and smaller research communities.

Ultimately, this dissertation opens the discussion on removing barriers that currently hinder the learning and use of ACT-R, thereby unlocking its potential. Making ACT-R more accessible encourages broader engagement and exploration of cognitive modeling in innovative ways.

**To Education**

This research offers a reflective examination of educational pathways, discussing the advantages and disadvantages of specializing in specific majors. It identifies the lack of cohesive thinking in the current educational landscape, thereby contributing to discussions about educational structures and their implications for future scholars.

The learning trajectory of ACT-R in Chapter 3, the individual strategies explored in Chapter 4, and the various error categories in Chapter 5 collectively highlight the importance of diversity. This emphasis on respecting diversity and individual differences with an open and growth mindset is essential in education, so that educators do not interrupt students when they are applying different approaches, whether in problem-solving or learning.

## Dissertation Overview

This dissertation deepens the understanding of the challenges ACT-R modelers face and help address sustainability issues within the broader research community. Chapter 1 introduces the research motivation, articulates the research questions and methodologies, and outlines the intended contributions to the field. Chapter 2 reviews relevant literature in ACT-R and cognitive modeling, detailing usability issues in modeling tools and improvements made in other similar AI tools. Chapter 3 presents an interview study that captures ACT-R modelers' experiences and insights. Chapter 4 showcases a practical demonstration of the TAKLML approach, which integrates hierarchical task analysis, the Keystroke-Level Model (KLM), and the power law of learning within ACT-R, to simplify modeling processes. Chapter 5 presents a second demonstration using the TAKLML approach to address previously challenging tasks, illustrating further simplification of cognitive modeling. Finally, Chapter 6 discusses key insights, outlines limitations, and suggests directions for future research.

# Chapter 2

# Literature Review

## Introduction

This chapter provides the foundational literature review for the dissertation, organized into three key themes. First, it introduces cognitive modeling by defining its role in understanding and simulating cognitive processes, with a focus on the ACT-R framework. This discussion situates ACT-R within the broader cognitive modeling landscape and underscores its potential for advancing knowledge in human cognition. The second theme examines efforts to improve the usability of ACT-R, reviewing prior studies and tool modifications aimed at reducing the steep learning curve associated with ACT-R. The third theme explores improvements made in other AI and modeling tools to identify best practices and potential solutions that could benefit ACT-R usability. Collectively, this literature highlights the importance and necessity of this dissertation. The review justifies the need for an initial interview study with ACT-R modelers to identify and understand their pain points and explore viable pathways for developing solutions that could alleviate these challenges.

## An Overview of Cognitive Modeling

### Cognitive Modeling

Cognitive science is a field dedicated to understanding the human mind and developing models that predict and explain human behavior. Within this field, cognitive modeling is the process of building cognitive models in computer programs to help explain human cognitive processes and

predict behaviors that involve cognitive processes. Cognitive architectures refer to the infrastructures once those cognitive models are implemented in computer programs. Several cognitive architectures, such as ACT-R, Soar, EPIC, and ICARUS, have been developed, each tailored for specific applications and areas of research (Choi & Langley, 2018; Kieras, Wood, & Meyer, 1997; Koripi, 2021; Laird, 2012). In practice, the term "cognitive architecture" encompasses more than a single, strict definition, as surveys of the field reveal. Most surveys characterize cognitive architectures as frameworks for intelligence, emphasizing mental representations and the computational mechanisms that enable complex behavior (Kotseruba & Tsotsos, 2020). These applications cover ten primary categories: human performance modeling, games and puzzles, robotics, psychological experiments, natural language processing (NLP), human-robot and human-computer interaction (HRI/HCI), computer vision, categorization and clustering, virtual agents, and other miscellaneous areas (Kotseruba & Tsotsos, 2020). Figure 2-1 is the template process of a cognitive modeling study after choosing a task.

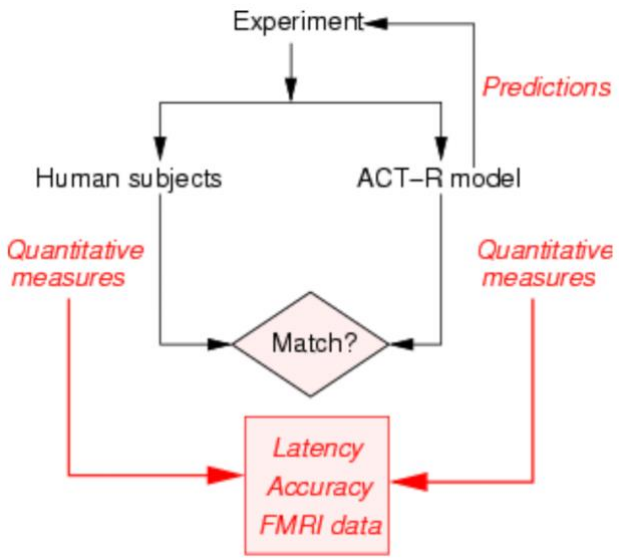



Figure **2-1**: *A Framework Process of a Cognitive Modeling Study (Oyewole, Farde, Haight, & Okareh, 2011).*

Cognitive modeling allows researchers to explore theories of cognitive processes in various tasks (Pew & Mavor, 1998; Pew & Mavor, 2007; Ritter et al., 2003; Ricupero, 2024). These theories are operationalized through models programmed to perform tasks. Those models may be simulated and implemented in the same programming language or in another software environment to solve the task. Additionally, some studies have explored methods that allow models to interact directly with interfaces without pre-specified instructions or simulations, which is particularly useful in testing user interfaces (Jones & Ritter, 2000; Tehranchi & Ritter, 2016, 2017, 2018a, 2018). Cognitive models are used as user models to test user interfaces (Tehranchil, Bagherzadehkhorasani, & Ritter, 2023; Ritter, Baxter, Jones, & Young, 2000). Modeling approaches range from straightforward, paper-based concepts to complex software systems (Ritter et al., 2018).

Collecting human data in cognitive modeling is often challenging. For example, in the Ben Franklin Radar system fault-finding task, researchers used a circuit simulator combined with a user input recorder to capture participants' interactions, including mouse movements and clicks (Ritter et al., 2022). These interactions provided insights into participants' cognitive activities while working with the interface. In this dissertation, human data from a previous Ben Franklin study is used to validate cognitive models, with task completion time serving as the primary quantitative measure. By comparing task times between models and human participants, high correlation coefficients would suggest that the models accurately reflect cognitive processes involved in task performance.

## ACT-R

Cognitive architectures function as both unified theories of the mind and as computational infrastructures for constructing intelligent agents (Choi & Langley, 2018). A Unified Theory of

Cognition is an integrated approach to understanding human cognition (Newell, 1993). Originating from cognitive science, ACT-R is one of the architectures developed to support these unified theories of cognition, realized as a computer program (Ritter et al., 2018). Cognitive models have been used in various contexts. They serve as agents in synthetic environments (Best et al., 2002; Pew & Mavor, 1998; Ritter et al., 2012) and are used in system design to predict and analyze user behavior (Booher & Minninger, 2005; Pew & Mavor, 2007).

ACT-R itself may refer to the theory, the architecture, or the implemented computer program, all of which contribute to a range of applications. For example, ACT-R models help researchers understand memory, build cognitive tutors, understand vehicle operators, test interface designs, simulate agents, and study language, multitasking, cognition changes, etc. (Ritter et al., 2018). Figure 2-2 presents a schematic diagram of the ACT-R cognitive architecture.

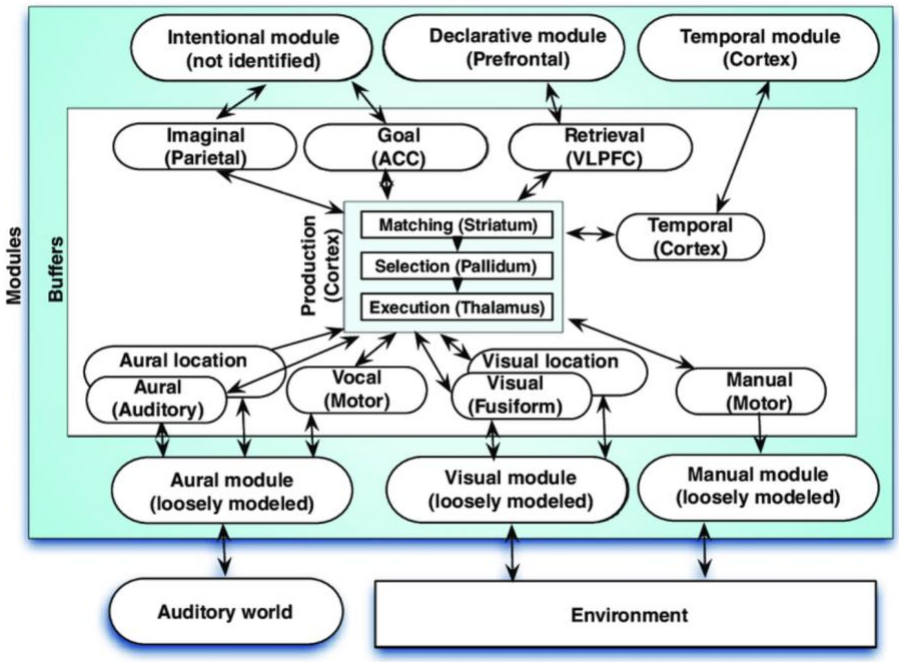



Figure **2-2**: *Schematic Diagram of the ACT-R Cognitive Architecture* (Ritter et al., 2019).

Also, ACT-R is a rule-based language that operates at a low level, where domain-specific knowledge is encoded as rules. This distinct style of programming is designed to represent cognitive processes directly, enabling researchers to simulate and study a wide range of human behaviors and cognitive phenomena.

## Research Efforts on Enhancing ACT-R Model Usability

Design errors are not unusual, and this issue has been explored in research, such as the study by Baxter, Churchill, and Ritter (2014), which addresses the fundamental error of design using the ABCS framework. ACT-R tools also contain design flaws, which result in usability challenges in various modeling tasks (Pew & Mavor, 1998; Pew & Mavor, 2007; Ritter et al., 2003; Ricupero, 2024). Various efforts have been made to improve the usability of ACT-R for cognitive modeling by addressing both learning and model-building processes (Amant et al., 2005; Büttner, 2010; Ritter et al., 2012; Paik et al., 2015; Brasoveanu & Dotlačil, 2020). While this dissertation primarily focuses on enhancing the learning experience and model-building aspects, expanding model connectivity with external task environments—such as the work by Tehranchi and Ritter (2018, 2017) in updating eye and hand models to facilitate ACT-R's interaction with user interfaces—remains an important area for future research. This section highlights two primary directions of research: (a) developing high-level model compilers and (b) re-implementing ACT-R in alternative programming languages.

One approach to simplify model building has been the development of high-level model compilers. For example, Amant et al. (2005) created the GOMS to ACT-R compiler, G2A, which allows users to develop ACT-R models by leveraging the GOMS framework. Additionally, Ritter et al. (2012) introduced, a high-level cognitive architecture aimed at easing model development. Paik et al. (2015) introduced a high-level behavior representation language (Herbal) and reported

that Herbal reduces model development time by a factor of 10 when compared to working directly in Soar, ACT-R, or Jess.

Another approach has been to re-implement ACT-R in programming languages beyond Lisp to address usability issues associated with Lisp. For example, Büttner (2010) introduced a Java simulation environment for ACT-R, while other researchers translated the architecture into Julia, providing an alternative to Lisp's limitations. More recently, Brasoveanu and Dotlačil (2020) introduced Python ACT-R, which was designed to improve accessibility and usability for researchers who prefer Python over Lisp.

Despite these advancements, nearly all of these tools and adaptations remain underutilized, often limited to use by the original developers or their research teams. As a result, this dissertation adopts a human-centered approach, aiming not only to address the usability of ACT-R tools but also to better understand the core pain points encountered by ACT-R modelers. This focus on user experience within the ACT-R research community seeks to identify broader solutions that could encourage wider adoption and collaboration.

A framework (Table 2-1, next page) that outlines strategies for making ACT-R more accessible was suggested in a book (Ritter, 2024). This dissertation incorporates and expands upon these strategies, integrating them into a structured analysis of pain points and user-based suggestions, detailed in Chapter 3.

## Similar Tools and Their Enhancements

Lessons on improving ACT-R's usability can be derived from advancements in other cognitive architectures and fields, particularly where ease of learning and application are prioritized (Koripi, 2021; Tsoi & Back, 1994). Soar, a cognitive architecture with similarities to ACT-R, has

received updates to improve its usability, including the Herbal compiler (Ritter et al., 2005). Although other architectures like Epic and Icarus lack extensive literature on usability improvements, Soar's case provides insights into leveraging compilers and programming aids to enhance user experience (Ritter & Larkin, 1994). Machine learning (ML) is an example of the other type of artificial intelligence. IntelliJ IDEA and VS Code are popular integrated development environments (IDEs) with strong debugging help.

Table **2-1**: *Potential Ways to Make a System Like ACT-R Easier to Learn and Use*.

| ID | Suggestions |
|---|---|
| 1 | FAQ |
| 2 | Compiler |
| 3 | Macro language |
| 4 | Descriptive languages |
| 5 | Building models from instructions |
| 6 | Building models from data directly, like LLM |
| 7 | Providing ACT-R in python/java |
| 8 | Manual |
| 9 | Tutorial |
| 10 | Examples |
| 11 | A set of models to build from |
| 12 | Video explanation of architecture showing the gears clanking |
| 13 | Updated GUI |
| 14 | Structured editor |
| 15 | Helpdesk |
| 16 | Video tutorials |
| 17 | More helpful tutorial reading materials, the little ACT-R-er |
| 18 | Summer workshop/summer school |
| 19 | Spring summer school |
| 20 | A unified theory of how this type of programming language is learned |

Machine learning (ML) offers another informative comparison. With its rapid adoption across academia and industry, ML has generated extensive educational resources, online courses, and community forums that make it accessible to diverse learners. Resources such as beginner courses, hands-on projects, and even no-code ML tools allow people with limited technical backgrounds to create ML models, emphasizing the importance of inclusive, varied educational

paths. ML's widespread adoption is driven by practical applications, media visibility, and promising job prospects, all of which foster an inclusive and accessible learning ecosystem.

For programming support, IDEs like IntelliJ and VS Code offer valuable lessons in user experience and debugging support. These IDEs enhance the programming process by providing autocomplete features, code formatting, color-coding, and efficient debugging tools. Debugging is simplified by feedback that pinpoints exact error locations, as shown in Figure 2-3, as well as breakpoints that allow targeted code testing, thus saving time and reducing frustration. These tools demonstrate the value of integrated debugging and user-friendly interfaces in making programming and modeling environments more approachable.

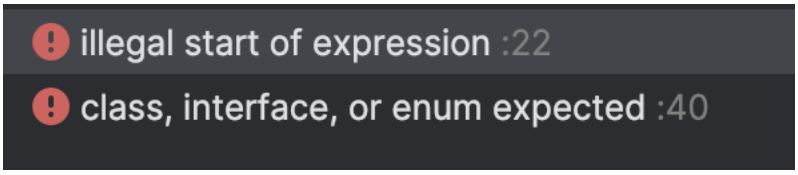



Figure **2-3**: *Screenshot of Warning Message in IntelliJ for Java.*

In summary, drawing from Soar's compiler use, ML's broad resource availability, and the debugging capabilities of modern IDEs highlights practical methods for improving the accessibility and usability of complex tools like ACT-R.

## Summary

This chapter reviewed the literature foundational to cognitive modeling and the ACT-R architecture, along with efforts to improve ACT-R's usability and lessons from related fields. It covered research aimed at simplifying ACT-R, including alternative architectures (e.g., Soar) and innovations from fields such as machine learning and programming, which demonstrated successful usability enhancements. Examples include machine learning's diverse learning

pathways and accessible resources and advanced IDEs like IntelliJ and VS Code, which streamline coding with tools like auto-completion, error pinpointing, and targeted debugging. These external examples provide useful insights into possible usability improvements for ACT-R.

The existing literature provides useful themes regarding ACT-R usability, but it remains limited in scope and detail. This gap indicates the need for further research, especially in documenting the pain points that ACT-R modelers encounter. Documenting these challenges would not only enhance the relevance of this dissertation but also offer valuable insights for future research aimed at improving the usability and accessibility of ACT-R.

To address this need, this dissertation includes a semi-structured interview study with ACT-R modelers from various regions. These interviews aim to provide a comprehensive understanding of the usability challenges and community-wide concerns, yielding insights that are currently underrepresented in academic literature. This approach supports the development of actionable suggestions for future improvements and strengthens the foundation for a more sustainable ACT-R research community. The following chapter provides details on the semi-structured interview study.

# Chapter 3

# An Interview Study of Modelers

This chapter presents a semi-structured interview study designed to understand the challenges experienced by ACT-R modelers and to explore concerns within the broader research community. The chapter covers the study's introduction, interview methods, recruitment approaches, participant backgrounds, and an analysis of the learning and application journey of ACT-R. It concludes with a discussion of the pain points identified along this journey, as well as potential solutions, limitations, and future directions for the study.

## Introduction

To investigate the challenges that arise when learning and using the ACT-R cognitive architecture, the author conducted a semi-structured interview study focused on three primary areas: (a) participants' overall experiences, (b) specific difficulties encountered, and (c) participants' suggestions for addressing these difficulties.

As described in Chapter 2, previous efforts to improve ACT-R usability have included initiatives like the Herbal compiler and implementations in Python and Java. However, these tools often remain limited in their impact, primarily benefiting their developers rather than the wider research community. Consequently, many researchers continue to struggle with ACT-R's usability, indicating a need for more comprehensive and accessible solutions.

This study documents some of these usability challenges in a systematic way to generate insights for improving ACT-R's accessibility. The findings provide a foundation for future efforts to support the research community, facilitating a more inclusive and sustainable environment for cognitive modeling advancements.

# Method

To investigate the experiences and challenges of ACT-R cognitive modelers, semi-structured interviews were conducted with participants from five countries, all with relevant experience in learning and using ACT-R. The interviews were semi-structured to effectively explore participants' experiences and challenges (Lazar, Feng, & Hochheiser, 2017; Cairns & Cox, 2008). This format allowed for follow-up questions to delve deeper into individual experiences and better understand specific challenges. Most interviews lasted between 30 and 60 minutes, while two were extended to one or two hours due to participants' enthusiasm. The following subsections provide detailed descriptions of the recruitment and participant profiles, interview procedures and questions, and data analysis.

## Recruitment and Participants

Participants were recruited through convenience sampling (Etikan, Musa, & Alkassim, 2016) and snowball sampling (Goodman, 1961; Naderifar et al., 2017). Recruitment efforts included emails and messages sent to personal contacts, individuals listed on the ACT-R website, and members of the ACT-R email list. Eligibility criteria for participation were broad, encompassing anyone with ACT-R experience, whether they were new learners or experienced users capable of teaching the system.

Initial contact was made with individuals who were known to meet these criteria. In these recruitment messages, potential participants were provided with an overview of the study, the criteria for participation, and a note confirming their eligibility. During an initial round of outreach, a pilot interview was conducted, and then five additional participants were successfully recruited. A subsequent recruitment round expanded the participant pool to a total of fifteen

individuals, including the pilot study participant. Figures 3-1, 3-2, and 3-3 illustrate the diversity in participant backgrounds. Details of the figures are described below.

A total of fifteen individuals participated in the interview study, residing across five countries at the time of the interviews: nine were based in the United States, three in Japan, one in Germany, one in Canada, and one in Sweden. These countries, depicted in Figure 3-1, represent the participants' places of residence rather than their countries of origin.

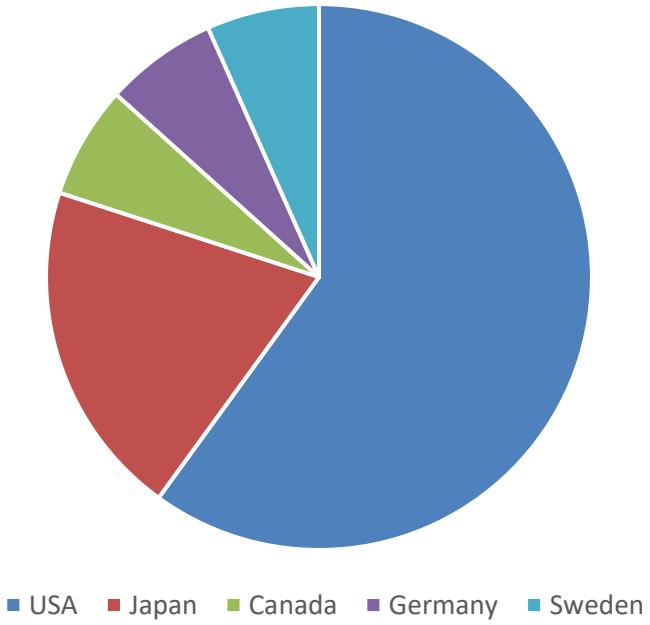



Figure **3-1**: *Distribution of Current Locations of Participants*.

As shown in Figure 3-2, the participants spanned a range of professional stages. Using the words of participants, those include two master's students, four doctoral candidates, three postdoctoral researchers, three assistant professors, one tenured professor, one government employee, and one lecturer. Their institutional affiliations covered a range of fields, such as cognitive psychology, cognitive neuroscience, engineering, behavioral informatics, education science, and philosophy.

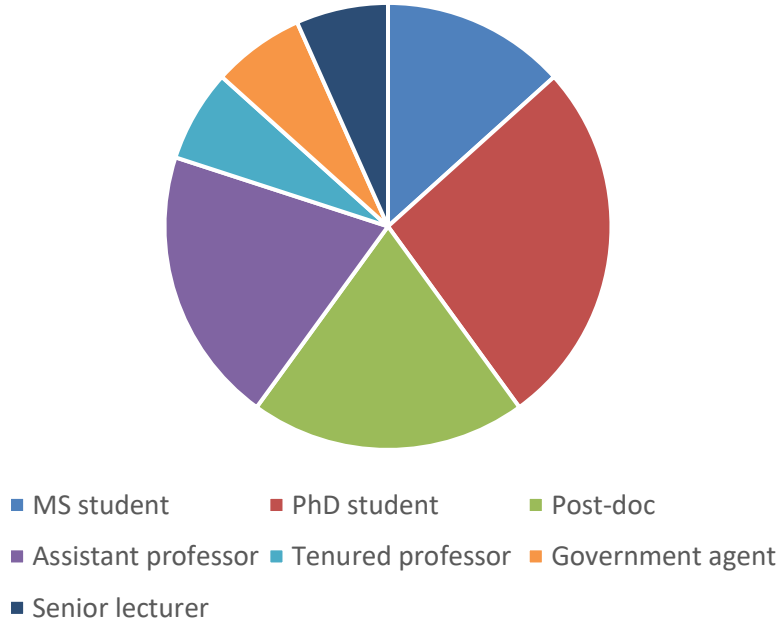



Figure **3-2**: *Distribution of Current Roles of Participants*.

Additionally, the participants' educational backgrounds reflect two primary threads, as shown in Figure 3-3. One group emerged from the cognitive psychology domain, with specializations in cognitive psychology, general psychology, and human factors. The other group came from programming-oriented disciplines, including computer science, engineering, informatics, mathematics, and physics.

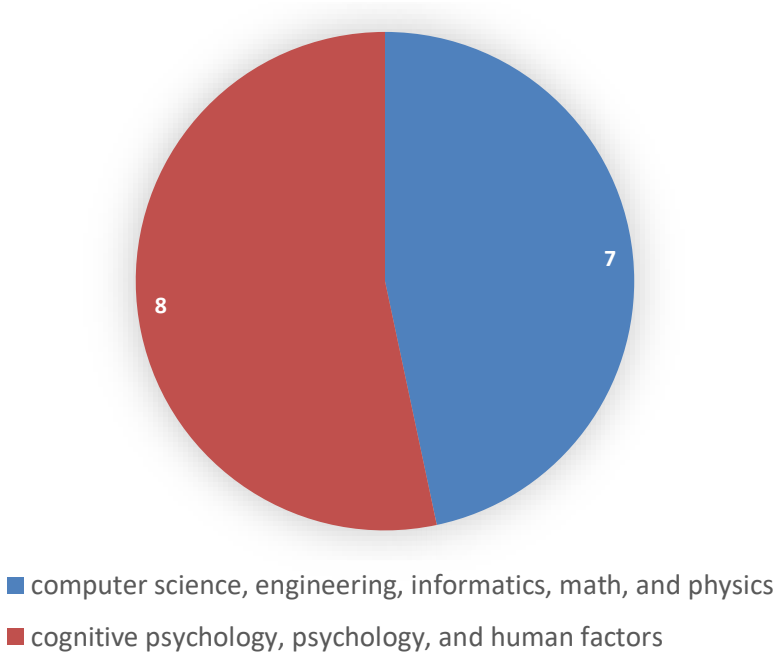



Figure **3-3**: *Distribution of Academic Background of Participants*.

Participants were not compensated for their time; however, they were informed that the findings of the study could contribute to advancements that would benefit the entire ACT-R research community. Common challenges and concerns emerged, such as difficulties with debugging, which are discussed in detail in a later subsection. Participants also contributed diverse perspectives shaped by their unique experiences and professional roles, offering valuable insights into the usability challenges and varying needs within the ACT-R modeling field.

**Procedures and Questions**

The interview study employed a semi-structured format (Adams, 2015; Andren et al., 2008) and was conducted online via Zoom (Ritter & Ricupero, 2023) to explore participants' experiences in learning and using the ACT-R. Due to the flexible nature of semi-structured interviews, the interviewer asked follow-up questions, when participants' responses invited deeper exploration.

For participants based in Japan with limited English proficiency, a translator was present to help as needed.

Each interview began with a greeting and an expression of gratitude for the participant's involvement. The interviewer provided an overview of the interview focus, explaining that the session would cover their experiences with ACT-R. Consent for recording was requested before proceeding. At the interview's conclusion, participants were invited to offer any questions or feedback, after which the interviewer stopped the recording and extended final thanks.

The interview questions (Appendix A) were designed as open-ended prompts across four main categories: (a) learning experiences, (b) usage of ACT-R, (c) task-related model building, and (d) insights for the ACT-R community. These categories, while loosely defined to avoid limiting responses, encouraged detailed accounts and follow-up inquiries as relevant themes emerged. Below is a summary of each category with the prepared questions:

**Learning Experiences**: This category aimed to capture participants' learning journeys with ACT-R through four prepared questions with any follow-up questions. Participants were asked about their initial approach to learning ACT-R, including resources, guidance from mentors, and any challenges encountered. They were also invited to suggest improvements based on their learning experiences.

**Using ACT-R**: In this section, "using ACT-R" broadly encompassed building models with the ACT-R architecture and using any related tools. Participants described their usage methods, timeframes for building models or publications, and difficulties they encountered. They also provided suggestions for improvements.

**Task-Related Model Building**: This section focused specifically on participants' experiences in building models for research tasks. Participants discussed tasks they had previously modeled and the time required for coding.

**Insights for the ACT-R Community**: Finally, participants provided reflections on the broader ACT-R research community. They shared perspectives on why some researchers may avoid or leave ACT-R research and offered suggestions for further participant recruitment.

Each category was designed to elicit comprehensive responses, facilitating a nuanced understanding of the challenges faced by ACT-R users and insights for future improvements within the community.

In addition to the four primary themes initially prepared for this study, additional themes emerged from participants' responses, revealing new insights into the experiences and challenges of ACT-R modelers. These emergent themes are thoroughly analyzed in the Results section, contributing to a more comprehensive understanding of the broader landscape of ACT-R usability and community concerns.

## Data Analysis

To prioritize practical impact within the field, this study extracts specific challenges and suggestions from the data, using this goal to guide the analysis process and maximize its applied relevance. The prepared interview questions were organized into four thematic categories: (a) learning experiences, (b) usage of ACT-R, (c) task-related model building, and (d) insights for the ACT-R community.

The interviewer reviewed the digital audio recordings and transcripts multiple times, which provided familiarity with the content and an initial sense of emerging themes. The initial step involved transcribing and organizing key phrases from each interview transcript into an Excel sheet. To ensure the analysis produced practical findings, codes and themes focused on identifying specific pain points and practical solutions, such as materials and tools that could ease common challenges. Key themes and subthemes within the interview data were identified (Hsieh

& Shannon, 2005). This coding and grouping process was iterative, repeated until the codes and themes fully captured the main points of the data (Glaser & Strauss, 2017).

The interviewer iteratively renamed and organized the themes and subthemes with codes to identify logical connections and patterns. This iterative process supported a coherent organization of findings, shaping the structure of the Results section. An example of coded transcripts is provided in Appendix B.

Additional themes emerged from the participants' responses, expanding the initial four categories and reflecting the broader experiences within the ACT-R modeling community. The emerging themes include: (a) general observations about the ACT-R user experience, (b) specific pain points or challenges in the user journey related to materials, coding, and community, (c) reasons for joining the ACT-R community and for leaving the field, and (d) suggestions.

Table 3-1 outlines themes and subthemes with codes and descriptions. Note that the study involved only one coder, and the codes were not verified by an external coder. Themes, like “coding-related challenges” have intricate layers that provide details for further analysis and exploration.

These themes provide a structured understanding of the complex landscape of ACT-R modeling, which will be explored in the Results section to highlight insights for enhancing user experiences and community engagement. Suggestions will be presented and discussed in the Discussion section.

Table **3-1**: *Descriptions of the Themes and Codes.*

| Theme and Sub-theme | Codes | Description |
|---|---|---|
| **General observation** | Book, summer workshop, teach self, … | Contextual insights into diverse ACT-R modeling journeys. |
| **Material-related challenges** | | |
| Understanding the concept | Tutorial, for dummies, … | Challenges related to the clarity of educational materials. |
| Information search | Look up, a lot to read, … | Difficulties in accessing relevant resources. |
| Gap between mimicking examples and creating new | Hard to create own, do not know how, … | Struggles transitioning from mimicking to original modeling. |
| Lack of guidelines and standards | Good model, … | Absence of guidelines for good modeling. |
| **Coding-related challenges** | | |
| Tool setup | Install, download, setup, … | Obstacles in configuring ACT-R tools and software. |
| Debugging | Lisp syntax, production rules, stepper, traceback | Technical issues in code debugging. |
| Code sustainability and compatibility | Lisp update, ACT-R update, system update, … | Issues with code longevity amidst updates. |
| **Community-related challenges** | | |
| Apprentice and centralization | Limited access, word of mouth, … | Barriers to educational resources, community engagement, or core architecture. |
| Education and career paths | Major, program, … | Variability in educational backgrounds affecting engagement. |
| Communication with external members | other fields, students, … | Communicating with other research fields, with students and the public |
| **Reasons for joining and leaving** | Course requirement, personal interests, no support from school, … | Factors for joining ACT-R and reasons for disengagement. |
| **Suggestions** | Recommendation, suggestion, … | Participants' recommendations for improving the experience. |

# Results

This section presents the findings from the interview study, highlighting the resources participants utilized in their journeys of learning and using ACT-R, as well as the pain points encountered and factors contributing to attrition within the research community.

The learning paths of participants varied significantly, and a summary of their experiences, challenges, and concerns provides a comprehensive overview of the research ecosystem. Understanding these diverse perspectives is crucial, as it allows for a better understanding of how various issues may be interconnected within the ecosystem and facilitates more effective solutions to address these challenges.

## General Observation

This subsection summarizes the learning resources utilized by participants and their diverse educational backgrounds. As detailed in Table 3-2, the resources mentioned include formal education courses, ACT-R summer workshops, the ACT-R website, tutorials for the Lisp version which can be download from the ACT-R website, academic papers, books, guidance from advisors, peer support, the ACT-R mailing list, and contributions from renowned experts in the field.

Table **3-2**: *Learning Resources Mentioned by Participants* (not by any order).

| No. | Resource |
|---|---|
| 1 | Course by formal education |
| 2 | ACT-R summer workshop |
| 3 | ACT-R website |
| 4 | ACT-R tutorial |
| 5 | Papers |
| 6 | Books |
| 7 | Academic advisor |
| 8 | Peers |
| 9 | ACT-R emailing list |
| 10 | Other famous experts |

Participants had varying access to these resources, with self-directed learning being a common practice among ACT-R modelers, regardless of the presence of advisors or expert assistance. Recent students often benefited from a more structured learning path, aided by their advisors and formal courses, while more experienced modelers typically followed a less linear trajectory, learning ACT-R intermittently.

The ACT-R tutorial emerged as the most widely recognized and utilized learning resource, as depicted in Figure 3-4. Some participants were required to engage with foundational books, such as Newell (1993) and Anderson (2007), whereas others had never encountered these works. Three participants who attended the ACT-R Summer Workshop emphasized its value, noting that it not only enhanced their understanding of ACT-R but also provided valuable networking opportunities within the community, enabling them to ask questions and build connections.

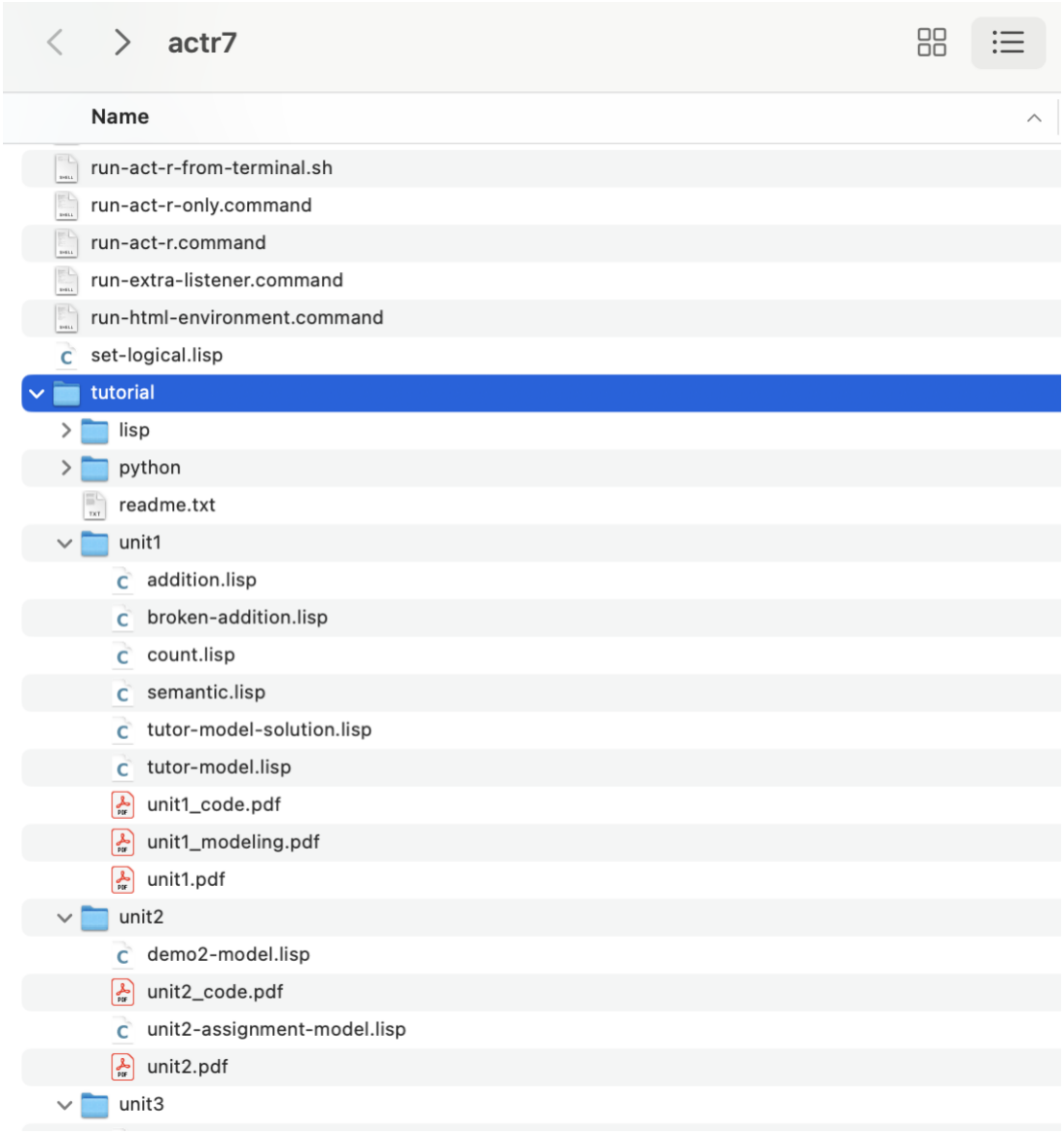



Figure **3-4**: *Screenshot of Tutorial, Downloaded as Part of the Standalone ACT-R7 Application*.

The ACT-R mailing list is not widely recognized or used among modelers. Only one senior modeler mentioned it prior to my inquiry. While senior members of the community expected the mailing list to serve as a communication platform for juniors seeking assistance, many junior researchers preferred to tackle challenges independently rather than posting questions to the list. This behavior may stem from the struggles that junior researchers often face. For instance, P10 noted frequent debugging challenges and felt a sense of responsibility for

resolving these issues on their own, not wanting to trouble others with what they perceived as "low-level" problems. A more detailed discussion of these debugging issues will be provided in later subsections.

The ACT-R community is characterized as a "small community," both globally and locally. This characterization aligns with insights gained from my conversations with senior members and feedback from interview participants. Quantitative evidence supports this observation, as there are about 240 contacts listed on the ACT-R website under the "People" section, as illustrated in Figure 3-5. However, the website may not have been updated when the contact information was extracted, some individuals may have opted not to list their details, and some may no longer be active (active: engaging in ACT-R related activities).

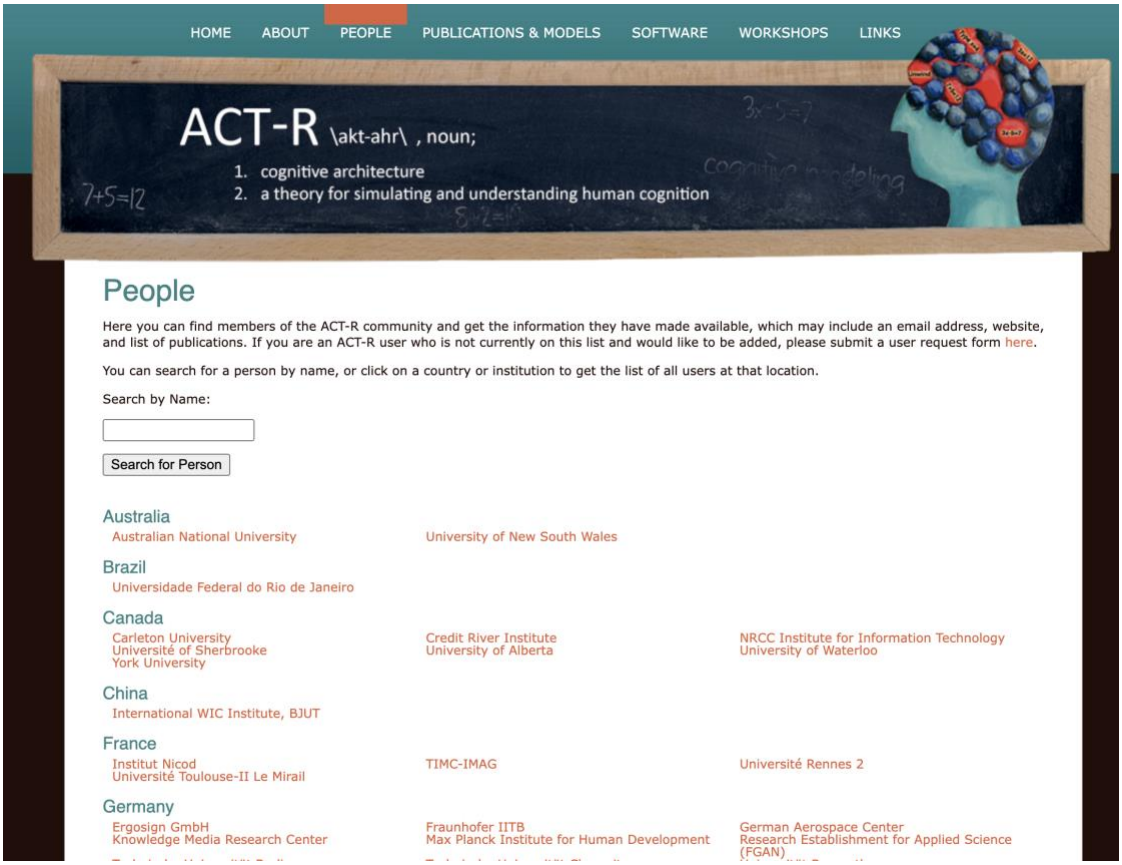



Figure **3-5**: *Screenshot of the ACT-R Website*.

Interview invitations were distributed to email addresses with valid formats containing "@" through my university email account. Apart from one automated response indicating "I am retired," there were 63 delivery failures encountered. These failures included messages stating that the email address might be misspelled or nonexistent, that the message could not be delivered, or that there was a communication failure during delivery. More specific examples included messages such as "the email address might be misspelled or may not exist," even though the email addresses were copied and pasted accurately without any errors on my part. Those messages included "Your message to ***@*** couldn't be delivered", "Your message wasn't delivered because the recipient's email provider rejected it", "A problem occurred while delivering your message to this email address", "The recipients weren't found at ***", and "A communication failure occurred during the delivery of this message".

For students facing challenges, their available resources are limited to advisors and peers. In the ACT-R community, Dan Bothell, the maintainer, is a prominent expert and often serves as the primary point of contact for ACT-R-related issues. Many participants have sought assistance from him. In contrast, more senior researchers and professors often lack colleagues with shared research interests, which further isolates them. This disconnection can extend to manuscript submissions, where they find it necessary to explain their work to both the computer science and cognitive psychology communities.

The limited understanding and frequent misconceptions surrounding ACT-R have resulted in significant challenges for the research community. Potential new members may hesitate to join, and existing researchers might leave due to insufficient support, especially concerns of the application and the job market. Those who remain often face not only isolation but also the burden of educating others about ACT-R, including reviewers of their manuscripts for computer science or cognitive science conferences. For example, they have to persuade the reviewers that their papers belong to them when submitting their manuscripts to other computer

sciences or cognitive science conferences or journals. More details will be provided in the subsequent section.

Participants also employed various approaches to building ACT-R models, using editors like Emacs and VS Code for coding. They executed ACT-R models using terminals or standalone applications with graphical user interfaces, as illustrated in Figure 3-6. Cockburn et al. (2015) emphasized the importance of user interface for novice-to-expert transitions. ACT-R interface could be improved. The programming languages involved in implementing ACT-R include Lisp and Python. No participant had tried ACT-R modeling in Java. In some courses, students are taught using the Python implementation of ACT-R, while models written in Lisp are referred to as the “canonical” version, according to P8. P11 noted that contemporary course content in computer science differs significantly from that of previous decades, with many recent students lacking a strong foundation in Lisp due to the high-level languages that are now emphasized.

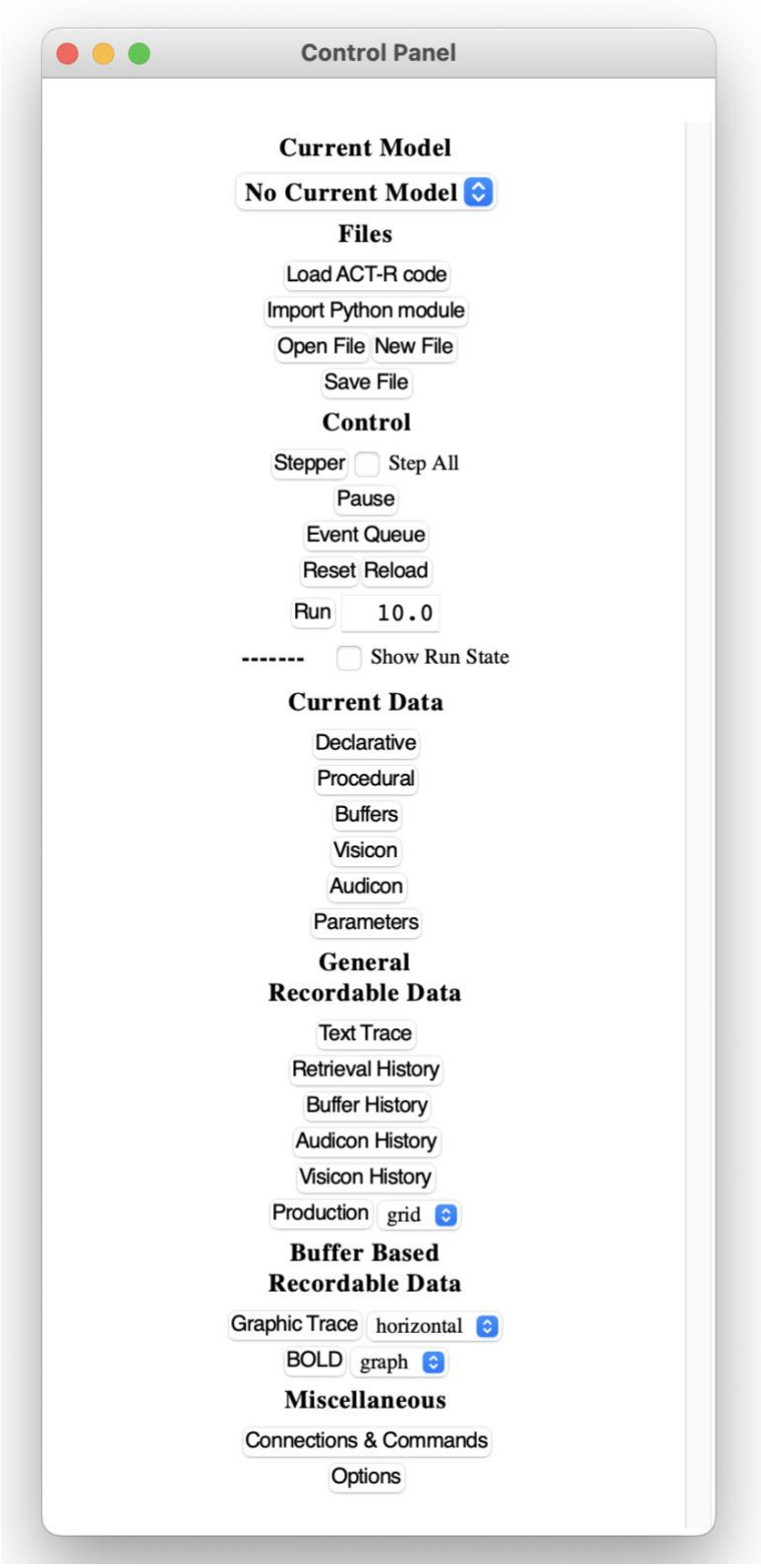



Figure **3-6**: *Graphical User Interface of the Standalone ACT-R Application.*

Considerable diversity was observed in the participants' journeys of learning and model development. This study primarily focuses on identifying pain points, and therefore, participants were not pressed for information regarding any additional learning resources they may have utilized after their initial responses. The distribution of various learning materials used was not systematically recorded.

Overall, ACT-R is recognized as complex and challenging to master when building models. Participants with backgrounds in computer science expressed that the time they devoted to learning ACT-R could have made them experts in other areas. Even professors and senior lecturers noted their hesitance to claim comprehensive knowledge of ACT-R due to its complexity and ongoing evolution. A common coding approach among participants was writing code incrementally and conducting frequent tests, to ease the debugging challenges encountered in ACT-R model building.

The following subsections will explore these difficulties and pain points in greater depth, addressing broader concerns within the research community and the factors contributing to individuals' departures from it.

**Pain Points of User Journey**

In this subsection, the concept of pain points is identified based on a thorough analysis of the participants' transcripts. These pain points have been categorized into three primary themes: material-related issues, coding-related issues, and community-related issues. These three themes were derived from the motivation of this dissertation to understand users' learning and usage experiences with ACT-R, as well as to explore concerns related to the community. The usage experience of ACT-R primarily involves coding. Each of these major pain points will be elaborated in the following subsections.

***Material-related Challenges***

Numerous resources can be accessed alongside the ACT-R application from the ACT-R website, including the tutorial and user manual. Among these, the tutorial is the most widely recognized and utilized resource, with P3 describing it as "the most helpful." Participants also expressed various challenges they encountered while engaging with the tutorial. In this subsection, (a) *Challenges in Understanding the Concept*, (b) *Challenges in Information Retrieval*, (c) *The Gap Between Mimicking Example Models and Creating Original Models*, and (d) *Lack of Guidelines and Standards for Effective Modeling*, are described in detail. These issues are depicted in Figure 3-7.

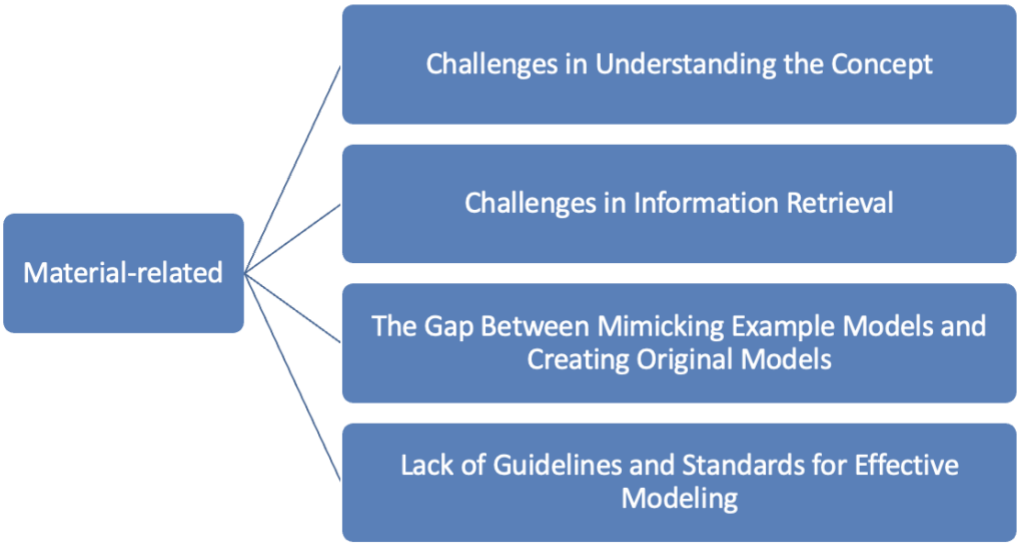



Figure **3-7**: *Material-related Challenges.*

**Challenges in Understanding the Concept:** The tutorial associated with the standalone Lisp version of the ACT-R application is characterized by its detailed and dense content. It covers various aspects of the ACT-R cognitive architecture, providing one or more example models at the end of each chapter. P15 remarked, “when you start, you don’t really have a clue of whatever it is that you are reading, where that fits in the whole picture.” The tutorial integrates materials from both cognitive science and computer science, necessitating foundational knowledge in both fields to fully grasp the cognitive architecture. Readers specializing in either cognitive science or computer science reported difficulty in comprehending certain sections due to their lack of background in the other discipline. For instance, P15 expressed confusion about the imaginal module and the buffer, stating, “like, it doesn’t make sense. I don’t know how I found out, but there was this specific thing (buffer) that took me a long time to figure out.” Understanding the architecture and how the modules and buffers relate to actual brain functions proved challenging. P6 noted that it was “a little tricky to understand because the brain does not have modules.” P9 further shared, “For Python, there was a different tutorial, and I think that was easy to follow and easy to understand. But when coming back to Lisp, that kind of transition has not been very easy.”

Participants also commented that the examples provided in the tutorial served primarily for learning purposes and lacked coherence, stating they did not align with specific scientific or research concepts but were instead intended to showcase the capabilities of ACT-R.

**Difficulties in Information Search:** Regarding the challenge of information retrieval, participants often struggled to locate the necessary content within the provided materials. P15 recounted their desire to remove certain chunks but faced difficulties achieving this goal, ultimately having to design an algorithm independently. They noted that a better understanding of the manual would have clarified the commands available for removing chunks. Reflecting on their experience, P15 expressed uncertainty about what the tutorial contained, stating, "it would have been beneficial for me to know the existence of the imaginal buffer a little bit earlier, but I guess that's not really in the tutorial. I'm not sure."

**The Gap Between Mimicking Example Models and Creating Original Models:** Concerning the gap between replicating example models and developing original models, P15 remarked, "the tutorial provides the theories and the example models but does not provide the guidelines on how you can come up with your own models. There is a huge gap." Many participants indicated that they often modified existing models for their own purposes, feeling lost when attempting to create a model independently.

**Lack of Guidelines and Standards for Effective Modeling:** Regarding the absence of guidelines and standards for assessing model quality, multiple participants expressed uncertainty about the validity of their models upon completion. They were unsure how to improve or optimize their work after finishing. P15 articulated, "you have no sense of what good modeling is. You only get that by talking to more experienced people, I guess." This participant further elaborated, "I think one part of the learning process that is difficult is that, especially because I did it on my own, I had a very, very weak sense of what is good modeling and bad modeling practice." They also acknowledged, "that is a type of knowledge that is also very difficult to write down in text."

***Coding-related Challenges***

Participants reported several challenges related to coding and writing ACT-R models. For some, this process can be quite difficult; for instance, one professor mentioned a student who was unable to complete their first model after two years and ultimately had to leave the graduate program. In this subsection, detailed descriptions of the difficulties encountered in (a) *tool setup*, (b) *debugging*, (c) *code sustainability and compatibility* issues are provided. These challenges are illustrated in Figure 3-8.

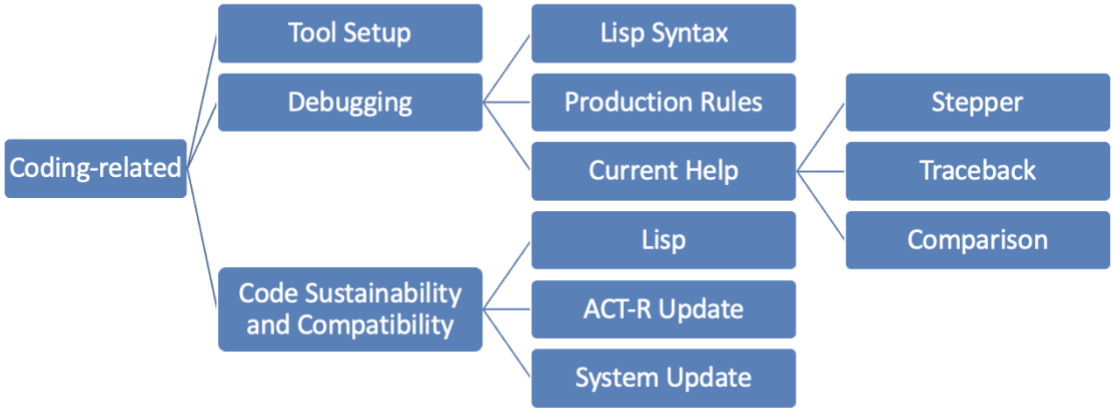



Figure **3-8**: *Coding-related Challenges.*

**Setup Difficulties:** Participants encountered significant challenges during the setup process, which fall at the intersection of material-related and coding-related issues. Successfully coding and running ACT-R models involves several setup steps, particularly concerning basic Lisp, which includes Clozure CL (CCL) and Quick Lisp, the foundational language for ACT-R. P8 emphasized that it would be beneficial for instructional material developers to assume that new researchers may have limited programming knowledge, stating, "I'm guessing because it's made for a very specialized community of researchers, they assume a lot of technical background." While participants eventually managed to complete the setup, they noted that providing clearer guidelines, particularly video tutorials, would expedite the process. P1 remarked, "I never would have known how to do the steps I did except that I asked my advisor about it. Lisp is actually pretty, I think, not user-friendly." P11 expressed reluctance to repeat the setup process after upgrading their system, noting, "if you have to reinstall everything, it is really challenging." Similarly, P5 shared their frustration with Emacs, stating, "I feel it's difficult to install or begin to use these tools. If someone could help me install them, it would be easier to start running." Overall, participants indicated that they required hands-on assistance from at least one other person, as they struggled to install the necessary tools using only online resources.

**Debugging Challenges:** Once the setup was complete, debugging became a significant issue in the model-building process. P8 asserted, "the debugging part is the most difficult part when you build your models," adding that "debugging is the only difficult part." This challenge is more pronounced in Lisp compared to Python, where integrated development environments (IDEs) offer more user-friendly debugging assistance. P3 shared that while they could complete the first draft of their model in less than an hour, the debugging process consumed several weeks.

In the following subsections, the debugging issues arising from Lisp syntax and ACT-R production rules will be discussed. Current debugging tools will also be highlighted, including the stepper function from the standalone application and commands for examining trace details.

Additionally, a comparison will be made between the debugging support available for ACT-R models and that found in other programming environments.

As for syntax, most participants reported difficulties with Lisp syntax. P10 remarked, “the syntax is not easy to understand. It does not align very well with the theory.” Modelers often found themselves stuck on syntax errors for hours or even weeks, which could stem from simple mistakes like misspelling a word or switching letters. P10 noted, “the overall flow of the code is easy enough to understand, but you can make syntax errors very quickly, and you can’t trace it. It is difficult to trace back.” The trace functions will be elaborated later in this subsection.

As for production rules, Participants also highlighted the challenges of correctly implementing ideas through production rules, particularly within larger models. P15 commented, “you spend so much time on debugging. It is crazy.” They encountered situations where the model appeared to run successfully, but incorrect production rules were activated, leading to the selection of the wrong chunks. P15 found this particularly frustrating, stating, “that is actually the most annoying because you don’t get error messages.” Consequently, they had to meticulously review the entire trace, which is time-consuming and complicates the debugging process.

Modelers have access to various tools to assist with debugging, such as the stepper function from the standalone application, depicted in Figure 3-6, and commands for adding or examining trace details. The stepper function allows modelers to execute their models step-by-step through each production rule, enabling them to understand the model's behavior. However, participants found the stepper function to be very time-consuming, especially with larger models, particularly if the error occurred at the end of the process or during a later trial. They also provided suggestions for improving the stepper function, which will be addressed in the recommendations section.

Participants utilize trace commands to gain insights into their models' behavior. However, the effectiveness of the trace tool diminishes as model complexity increases. P3 stated, “the

output of the trace is hard to read. It is full of words." While error messages are generated when code fails to run, they are not always helpful. P10 shared that when they received 40 warnings for a single model, they simply thought, "oh, ignore that." Suggestions made by participants will be discussed in the recommendations section.

While debugging issues are common across programming, the challenges specific to debugging ACT-R models deserve emphasis. "When I learn, when I try to build my first model, I just feel everything is so outdated and is not updated. They are they're being left behind by the majority of other like programmers or programming languages," said P6. Other participants also shared similar impressions. For example, people may find Google, stackoverflow.com, and Gen AI very helpful when they debug for their coding tasks. However, those tools are not very helpful in problem solving for ACT-R issues. ACT-R is a small community. P10 shared that they do not know where other ACT-R modelers would share their coding information, and they do not have access to the personal small groups of ACT-R to ask for help. "It is not connected across the world", they said. The debugging functions for other languages, such as those in VS code for Python and those in IntelliJ for Java, are very strong. They have debugging mode and can add breakpoints. The error warnings can direct programmers to certain lines, which really saves time and energy. The code environment for Lisp may have similar features but is not as strong. P6 tried a distortion plugin for VS code about Lisp. "I did not know if it was working or not, but obviously it cannot work with ACT-R. My background was not in computer science. I'm still trying to understand how the plugin would do the work", they said.

Even though modelers successfully build their models, they face code sustainability and compatibility problems. The sustainability and compatibility are also tied to Lisp programming language updates, ACT-R updates, and computer system updates, both Mac and Windows. A participant shared their concern that the Lisp coding interface is going to die fairly soon and that there will not be any developers because there is not enough demand or users. There is also not

enough money as incentives for those Lisp coding interface developers. This maybe a reasonable concern. The concern was shared with a Lisp developer who has a strong belief in the capabilities of Common Lisp. They nominated ACT-R for "the most important Common Lisp project they had never heard of". To them, it is concerning that the ACT-R website is not mobile-optimized and says, "Copyright @2002-2013", indicating that this "seem like an old, mostly unmaintained project".

**Code Sustainability and Compatibility:** Even when modelers successfully build their models, they encounter issues related to code sustainability and compatibility. These challenges are connected to updates in the Lisp programming language, ACT-R, and computer systems, whether Mac or Windows. Participants expressed concerns about the longevity of the Lisp coding interface, fearing it may become obsolete due to insufficient demand and a lack of incentives for developers. This concern was echoed by one graduate student, whose advisor advised them to abandon thoughts of maintaining their model and instead move on. A senior researcher noted that they had not kept pace with the latest ACT-R updates.

System compatibility issues arise when CCL (Clozure Common Lisp) is not updated as soon as the computer system. P11 shared that they had to downgrade their iOS system version to continue make CCL work. To avoid further sort of upgrade problems, they had to switch to a Windows machine. They also compromised and started to use less libraries that can get out of date to avoid compatibility issues.

### *Community-related Challenges*

Participants identified several pain points related to the research community, which are categorize into three key areas: (a) *Apprentice and centralization*, (b) *Education and career paths*, and

(c) *Communication with external members*. Those issues are interrelated and can impact the sustainability of the research community. They are illustrated in Figure 3-9.

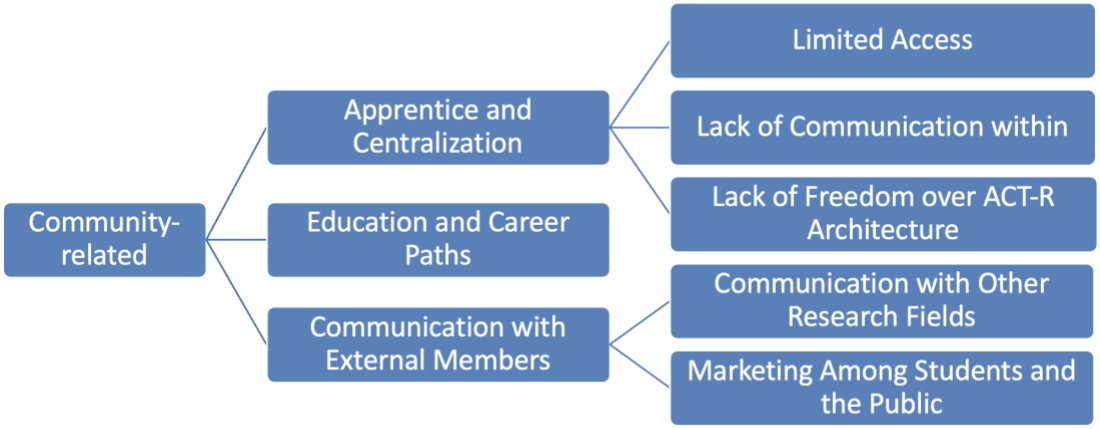



Figure **3-9**: *Community-related Challenges*.

**Apprentice and Centralization:** In the context of apprenticeship, participants highlighted the limited access to knowledge and assistance for those new to ACT-R, while centralization pertains to the concentrated control over the ACT-R architecture. Participants recognized that learners of ACT-R often assume the role of apprentices, meaning that those outside of formal apprenticeship networks struggle to access essential resources and support. As P10 noted, "A lot of components are still very much word of mouth." Beginning research in ACT-R as an independent researcher is "not impossible, but it would be really, really difficult." Without the guidance of a teacher or mentor, many participants expressed that learning independently could lead to significant challenges. For example, P4 said "It's going to be really struggling." Participants with backgrounds in other research fields and coding experience observed that individuals within the ACT-R community often seem disconnected. This lack of connection not only hampers the learning process for apprentices but also contributes to poor communication within the community. Although researchers can share their findings through publications and conferences, such as the International Conference on Cognitive Modeling (ICCM), many participants reported that they do not know many other ACT-R modelers beyond their immediate networks.

Regarding centralization, several participants expressed concern that the core knowledge of the ACT-R architecture and the authority to modify it are concentrated within one university or limited to just one or two individuals. While they acknowledged and appreciated the support from these experts, they also voiced worries about the transmission of this core knowledge. Training opportunities on critical aspects of ACT-R, such as altering module functionality or editing the cognitive architecture, are scarce. P3 articulated the challenge, stating it is difficult to "learn to fully control the code." Consequently, participants raised concerns about the long-term sustainability of the ACT-R research community.

**Education and Career Paths:** Participants with backgrounds in ACT-R modeling typically come from psychology or engineering disciplines. For many individuals, awareness of ACT-R and its connection to unified theory emerges only if they enroll in a course that includes ACT-R content; otherwise, they may not encounter this information in their general psychology or engineering education.

Concerns about future career paths can be a source of anxiety for students. P4 noted, “this is a big worry among students, especially because I’m in the cognitive science department now. We don’t train students for any particular job, but we have a very high hiring rate.” They further explained that the traditional career path is becoming increasingly misleading, as there is now considerable flexibility. For example, ACT-R graduates do not have a designated job title; instead, they are encouraged to leverage their transferable skills to effectively communicate their abilities, as job titles are subject to change.

One participant, who struggled to find employment, was advised to market themselves as a data scientist despite lacking a formal degree in the field. P4 also shared insights from a recruiting friend at a consulting firm, who stated, “he has only one rule when he is hiring a data scientist. If they have a degree in Data Science, he won’t hire them. He is looking for a set of skills, not the name of the degree”. A better understanding of the recruitment process behind job titles could greatly benefit students. Additionally, participants noted a growing crossover between academia and industry, emphasizing that “there is a lot of crossovers now between academia and industry.”

ACT-R is utilized in various projects for the U.S. government and military. Some ACT-R modelers are not U.S. citizens, leading to concerns about their employability in related fields. P4 commented, “They are still hired, but they have to be hired in a different way. If they really want you, they will figure out a way: you can be a consultant.” Junior students tend to express less

concern, particularly if they are deeply committed to ACT-R and have a strong personal interest in understanding cognitive processes, aiming for a career in academia.

**Communication with External Members:** Communication challenges with researchers in other fields, particularly when publishing, are a significant concern for ACT-R modelers. With the increasing specialization of academic fields, interdisciplinary researchers often face a unique challenge: they must master and integrate knowledge from multiple disciplines, as P4 explained, often doubling their research efforts. Despite the rigor and value of their interdisciplinary contributions, ACT-R researchers frequently struggle to find receptive outlets for publication, as their work may not align clearly with the expectations of many fields.

This challenge is compounded by misunderstandings from reviewers in fields like developmental or social psychology, who may not fully grasp ACT-R's contributions to cognitive modeling. P4 shared that submissions are sometimes dismissed with remarks such as, "that is not what we do," or by oversimplifying ACT-R as merely "code writing." Even when ACT-R work is accepted, researchers often find themselves as one of the few representing this approach outside of specialized venues like the MathPsych or ICCM conferences. As a result, ACT-R modelers often find themselves in the position of having to educate audiences from both psychology and engineering fields about their research, where others in more traditional fields may already have established understanding and rapport with their audience.

Outreach and marketing to students and the general public are also important. Currently, there is limited positive visibility for ACT-R, while negative perceptions circulate from those less supportive of the framework. For instance, P4 shared that some psychology professors openly criticize cognitive modeling, telling students that "ACT-R is a trick," or reducing it to "just writing code." This skepticism can lead to confusion among students, who might ask questions like, "you don't have neurons in your model. Why is it not wrong?" Additionally, P7 highlighted that ACT-R's applications are not widely recognized, especially in comparison to more

commercially visible AI techniques. This limited visibility is further complicated by constraints on public sharing of certain applications, as P4 noted, due to the confidential nature of work for government or military clients.

**Reasons for Joining and Leaving**

People learn ACT-R or join the community for various reasons and leave for a range of others. Based on collected data, individuals generally have three main motivations for starting to learn ACT-R or engaging in related activities, such as model building, publishing, and conference participation. As shown in Figure 3-10, these motivations include: (a) *Course requirements*, where students encounter ACT-R as part of a required course in their degree program; (b) *Job requirements*, where individuals in graduate programs or research roles need to work with ACT-R as part of their responsibilities, even if they were previously unfamiliar with it; and (c) *Personal interest*, where people are drawn to ACT-R due to a genuine curiosity about scientific questions related to the "mind" or "brain." This intrinsic interest also leads some to rejoin the community or return to ACT-R research after time spent in other fields.

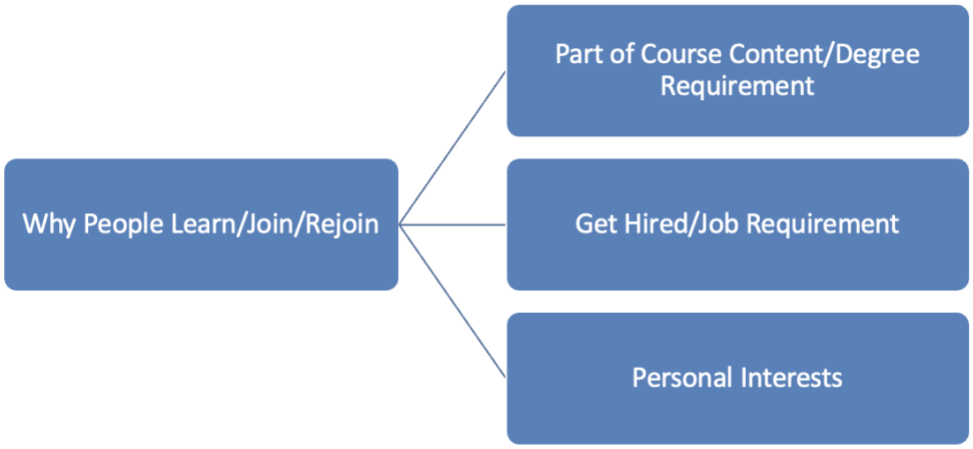



Figure **3-10**: *Why People Learn/Join/Rejoin*.

Based on the data collected, there are five main reasons people distance themselves from ACT-R. As shown in Figure 3-11, these include: (a) *Obtaining unrelated jobs*: ACT-R-specific roles are limited, particularly for recent graduates. Many students find non-ACT-R-related employment upon graduating and thus stop working on ACT-R, even if they wish to continue; (b) *Lack of institutional support or school requirements*: Faculty, especially early-career professors, may face pressure from their institutions to pursue other research areas. To establish independence and advance academically, they may shift toward innovative topics, diverging from ACT-R to build a distinct research portfolio; (c) *Difficulty in continuing with ACT-R*: Participants mentioned knowing students who struggled so much with ACT-R modeling that they ultimately had to leave their programs; (d) *Lack of awareness*: Many simply have no exposure to ACT-R, and without knowledge of its existence or relevance, they cannot engage with the community. As P1 noted,

> I think a lot of people don't know what ACT-R is. It's a pretty small community of people. The skills to do ACT-R and it takes a really long time to learn. learning ACT-R would take them, you know, months to figure out before they could actually do something with it that it integrates well with their work, so I feel like it's just not worth the time trade-off for a lot of people.

(e) *Lower appeal compared to other interests*: While some remain interested in ACT-R, various obstacles may make it less appealing compared to other pursuits.

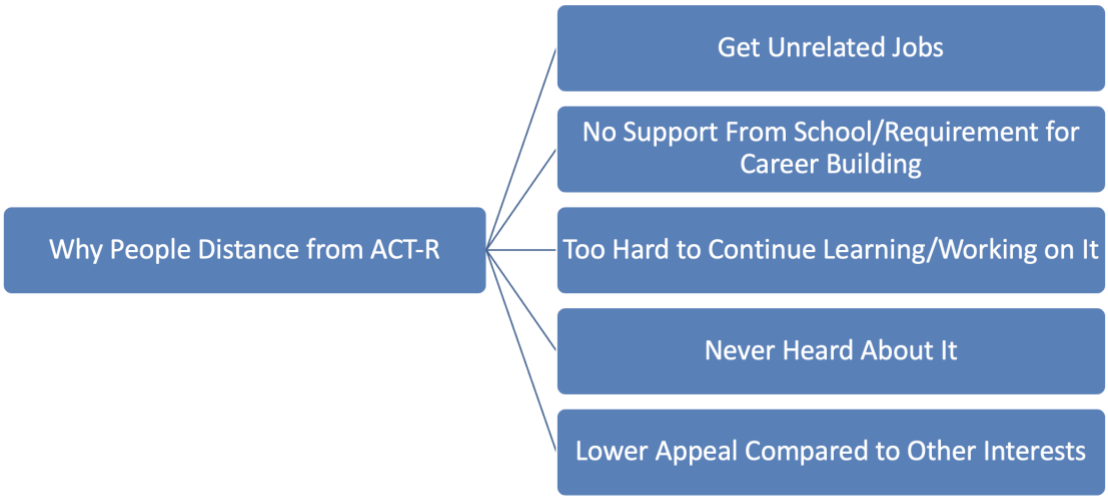



Figure **3-11**: *Why People Distance from ACT-R*.

## Discussion and Implications

This section collects suggestions from participants, including learning related suggestions, coding related suggestions, and community related suggestions. This section then discusses limitations of this interview study and talks about future work.

### Suggestions

The suggestions provided by participants are organized into three main themes: material-related, coding-related, and community-related improvements to address existing challenges and pain points.

***Material-related Suggestions***

Participants shared several ideas to improve tutorials and educational resources. Table 3-3 details these suggestions, which aim to address common difficulties, including understanding ACT-R concepts, bridging the gap between following example models and creating one's own, and the lack of guidelines and standards for constructing a robust model. Not all challenges have solutions at this time, but the suggestions collected here represent the current feedback from participants.

Table **3-3**: *Material-related Suggestions.*

| Pain Points | Suggestions |
|---|---|
| The difficulty of understanding the concept | 1. Reorganize content structure<br>2. Visualize the content structure<br>3. Provide videos |
| The difficulty of information search | N/A |
| The gap between mimicking example models and creating the first model | 1. Provide a precritical guideline to generate a model, with videos |
| The lack of guidelines and standards for a good model | N/A |

Three main suggestions were frequently mentioned by participants to enhance understanding of ACT-R concepts: restructuring tutorial content, visualizing ACT-R materials, and providing more instructional videos. Participants expressed a need for an organized mental model of ACT-R's different modules before delving into the details. P15 recommended a preview of the tutorial structure to "have a better sense of what is useful for readers and what is not so useful at the current moment." Beginning with an explanation of ACT-R's significance was also seen as beneficial.

Participants also emphasized the need for practical guidelines on creating models, suggesting that additional step-by-step videos demonstrating tool usage and guiding the modeling process would be highly valuable.

While there were no specific suggestions addressing difficulties in information search or the lack of standardized guidelines for quality modeling, P11 noted that defining what constitutes a "good model" can be challenging to articulate.

### *Coding-related Suggestions*

Participants offered several suggestions related to coding and building ACT-R models. As outlined in Table 3-4, detailed recommendations on tool setup, debugging, and addressing code sustainability and compatibility issues were provided. Although not all difficulties had corresponding suggestions, current feedback from participants has been compiled in this subsection.

Table **3-4**: *Coding-related Suggestions*.

| Pain Points | Suggestions |
|---|---|
| The difficulties in tool setup | 1. Video help |
| Debugging | 1. A virtual tutor<br>2. An interface that visualizes modules and buffers of the model<br>3. A developing and debugging tool like Jupyter notebook for Python<br>4. Moved all equations in Julia programming language |
| Code sustainability and compatibility issues | N/A |
| The lack of guidelines and standards for a good model | N/A |

Participants recommended creating videos to assist with tool setup, emphasizing that step-by-step guidance would be far more effective than written instructions alone. They prefer accessible, online resources over relying on one-on-one support.

For debugging, participants offered several suggestions, such as developing a virtual tutor, an interface for visualizing model modules and buffers, and a debugging tool similar to Jupyter Notebook for Python. More specifically, P3 described their suggestion of an interactive, visual interface that would enhance the usability of ACT-R modeling. They envisioned a drag-and-drop system allowing users to visualize modules and assign chunks in real time. The system could have a dynamic presentation of what was happening about chunks and production rules. It could also have the activation values on top of the visualized chunks. P7 suggested improvements to the stepper function, requesting that it display not only the fired production rule but also the conflict set and the next production to be selected. Participants noted that, compared to other coding experiences, ACT-R modeling can feel especially challenging without familiar debugging tools. They expressed a desire for ACT-R or Lisp to support debugging features commonly found in platforms like Jupyter Notebook, VS Code, and IntelliJ. Lisp has debugging tools, but participants may find it less convenient to use compared to Python in VS Code. One participant shared that they ported all ACT-R equations to the Julia programming language to avoid Lisp debugging complexities.

Regarding code sustainability and compatibility issues, participants acknowledged the difficulty of addressing these challenges and noted that no ideal solutions currently exist. These dilemmas will be discussed in detail in a later section.

### *Community-related Suggestions*

Participants shared their recommendations for fostering a sustainable research community. Table 3-5 details these suggestions, which address issues related to apprenticeship and centralization, education and career paths, and communication with external members. While not all challenges have proposed solutions, the suggestions gathered from participants provide valuable insights.

The bottom lines in Table 3-5 are used to enhance table structure for easier reading and avoid confusion.

Table **3-5**: *Community-related Suggestions.*

| Pain Points | Suggestions |
|---|---|
| Apprentice and centralization | 1. Encourage communication within the community<br>2. Decentralize ACT-R sources from certain people of CMU to more branches |
| Education and career paths | 1. More available resources to inform students about the flexibility of career paths, as well as job hunting skills<br>2. More job openings in ACT-R |
| Communication with external members | 1. Make clear the “why”<br>2. Make applications out there |

To address issues related to apprenticeship, particularly limited resources and social support, participants recommended a more organized online repository of resources and a dedicated forum for communication. They frequently suggested video tutorials, noting the value of YouTube videos in learning other programming languages. For centralization issues, participants proposed decentralizing ACT-R resources and knowledge from CMU to foster growth in more branches.

Regarding education and career paths, participants highlighted the need for more ACT-R-related job opportunities. Additionally, providing resources to inform students about flexible career paths and effective job-hunting skills could help alleviate anxieties, enabling them to better focus on their studies and research.

In terms of communication with external members, participants suggested that the ACT-R community could clarify the foundational "why" behind ACT-R research—such as its significance and rationale. P3 further emphasized the importance of increasing visibility for ACT-R applications.

## Limitations

This semi-structured interview study has several limitations. For the interview study, ACT-R modelers from the Netherlands were invited but did not respond. The University of Groningen is a place where a group of researchers gathered and are interested in cognitive modeling. Their learning and using experience of ACT-R might be slightly different. Additionally, this interview focuses more about coding in Lisp, less about Python. Future research could explore Python usage in ACT-R modeling in greater depth. As a limitation of the interview study, the participants' stories could not be verified, as the results only reflect their inputs. Furthermore, more experienced interviewers might have been able to guide the conversations more effectively, potentially uncovering deeper insights. This work arose from the author's intellectual curiosity and was conducted without any internal or external funding, ensuring the independence and impartiality of the findings. However, the study was not funded, implying no compensation for participants, not to mention getting a second coder. The absence of a second coder could also be seen as a limitation.

## Future Work

This study has highlighted key challenges faced by ACT-R modelers and broader concerns within the research ecosystem, along with potential solutions and recommendations. These insights provide a clearer understanding of the difficulties confronting the field. Future studies could delve into any of the identified pain points and propose solutions to explore them in greater depth. Each issue and suggested solution present an opportunity for future targeted research and improvement. It would also be useful to gain insights into potential directions by combining this results with Agre’s (2005) article on how to find research topics.

## The TAKLML Approach as an Example Solution

Many of the challenges identified from the ACT-R modeling community extend beyond usability concerns and into broader, systemic issues that cannot be resolved through a single dissertation. Addressing these community-level barriers requires a collective, long-term effort that transcends any one project's scope or resources. Given the practical limitations of this dissertation, it narrows down to the coding difficulties found in Chapter 3 and focuses on one feasible, impactful suggestion: leveraging ACT-R equations and coding in Python to streamline the modeling process. By using Python instead of the full ACT-R architecture in Lisp, this approach seeks to bypass the steep learning curve associated with traditional ACT-R model implementation and to ease the debugging process, offering a more accessible entry point for modelers while still preserving the theoretical foundations of ACT-R.

The TAKLML approach, which stands for Task Analysis, Keystroke-Level Modeling, and Learning using the power law from ACT-R, implemented in Python, was developed. Comparing to Keystroke-Level Modeling, the TAKLML approach applies to a large and complex task and also includes learning. By building on core principles of ACT-R, such as the power law of practice, and combining them with Python's flexibility, TAKLML simplifies model creation while maintaining alignment with cognitive theory.

In the following two chapters, two applications of the TAKLML approach are presented to illustrate not only the effectiveness of this method but also its capability to model complex tasks that have traditionally posed significant challenges in the field. These examples demonstrate how the approach addresses usability and accessibility issues within cognitive modeling, enabling researchers to engage in intricate modeling tasks more readily. By simplifying complex processes, the TAKLML approach exemplifies a viable solution for expanding the scope and efficiency of cognitive modeling practices.

## Summary

This chapter presented an interview study aimed at identifying the pain points experienced by ACT-R modelers and their broader concerns within the research ecosystem. A total of 14 participants, along with one pilot interviewee, were recruited for semi-structured interviews conducted via Zoom. The questions focused on participants' learning and usage experiences with ACT-R, as well as their perspectives on the overall research community. This chapter covered the findings, including participant backgrounds, an overview of their learning and application journeys with ACT-R, and the specific challenges they encountered. This chapter summarized the pain points noted and the suggestions provided by participants, along with additional insights identified. Finally, this chapter introduced the TAKLML approach, which will be utilized as a partial solution as an example in the following Chapter 4 and 5.

# Chapter 4

# Modeling with the TAKLML Approach

This chapter presents a cognitive modeling study designed to demonstrate the performance of an alternative approach that approximates ACT-R but is fast, the TAKLML approach (Task Analysis, Keystroke-Level Modeling, and Learning using the power law from ACT-R, implemented in Python). How much faster remines to be explored. This approach serves as a potential solution to the coding and debugging difficulties from identified pain points. The task modeled in this study is the MENDS task, which involves identifying a faulty component within a 35-component electrical circuit. Four strategies were successfully implemented for this task, illustrating how Python-based modeling with task analysis works. This chapter was based on a publication (Wang & Ritter, 2023a).

## Introduction

It is possible to construct a learning model that can capture individual differences in strategies as these strategies develop. Predictable sources of variability can be found in learning across time and learners. Modeling human behavior using different strategies and comparing them with the time course of individual learning behavior remains difficult and rare, but not impossible. A model that learns helps describe how humans use knowledge to solve problems and how the repetitive application of the same knowledge leads to faster performance.

A good overview of the categories of human learning is provided by Ashby and Maddox (2005). These models have had their predictions compared to averaged data but have not had their predictions compared to individual learners. Understanding individual differences can help understand the acquisition and application of complex skills. Averaging over strategies will

distort results and hide important and reliable effects (Elio & Scharf, 1990; Siegler, 1988b). Models of individual differences have been built by previous researchers in various ways, such as differences in global system parameters (Card, Newell, & Moran, 1983; Daily et al., 2001; Kase et al., 2017), differences in knowledge (Anzai & Simon, 1979; Best & Lovett, 2006; Kieras & Meyer, 2020) , and in how differences arise through learning (Larkin et al., 1980; Siegler, 1988a, 1988b).

Thus, previous work has argued for the need and the usefulness of representing strategies These models have shown that individual differences in problem-solving arise in multiple tasks, and that modeling individual strategies can lead to a better understanding of how the task is performed and how learning strategies arise (Daily et al., 2001; Delaney et al., 1998; VanLehn, 1991). Friedrich and Ritter (2020) also presented the case that strategies are important for understanding behavior, and that learning is important as a component to shift between and to evolve strategies. They pointed out that little research had been done on modeling different strategies for a task, because to do so, tasks need to be complex enough to allow multiple strategies, and individual behavior must be traced and analyzed, and multiple strategies must be modelled. Previous researchers have often reported one strategy of the task (e.g., Wang & Ritter, 2023a). To extend this line of work, this study reports how multiple strategies were modeled while they were being learned, how they matched participants' performance and how they inform us about behavior, transfer, and learning.

**The MENDS Task**

A complex electrical troubleshooting task was used to study problem solving, with a much more complex system, the Ben Franklin Radar System that has five times more components. Figure 4-1

shows the Ben Franklin Radar system schematic. It has 36 components, including a power supply that does not break: five times as many components as the Klingon Laser Bank task.

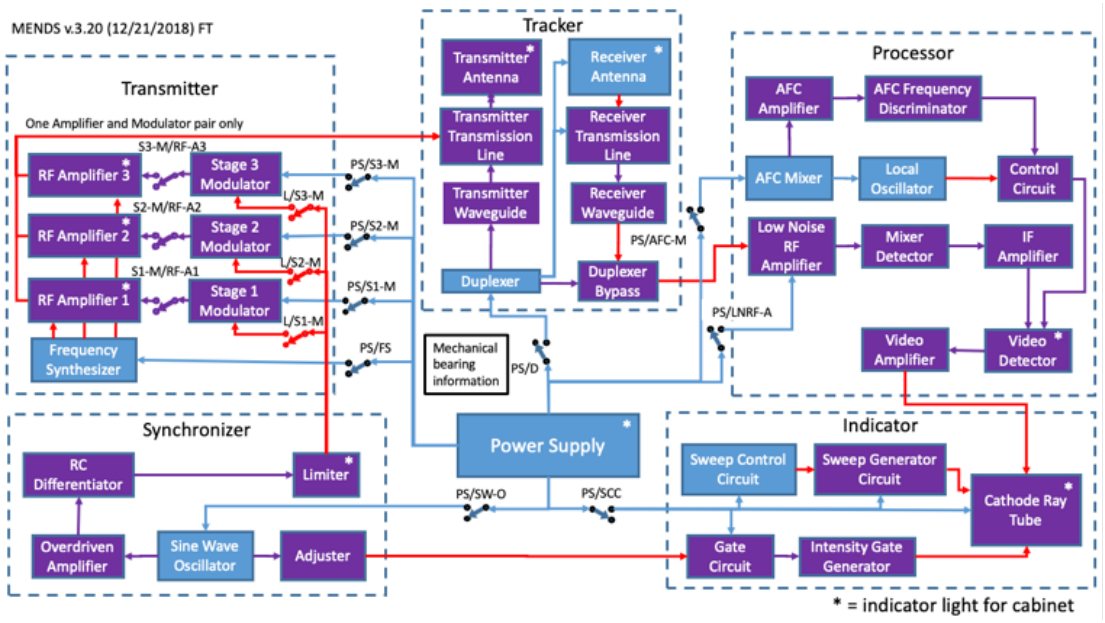



Figure **4-1**: *The Schematic for the Ben Franklin Radar*. Blue lines are power; red lines are signal; purple lines are both.

In the Synchronizer tray, components include Sine Wave Oscillator (SW), Adjuster (AR), Overdriven Amplifier (OA), RC Differentiator (RC), Limiter (LI). In the Transmitter tray, components include Frequency Synthesizer (FS), Stage 1 Modulator (S1), RF Amplifier 1 (R1), Stage 2 Modulator (S2), RF Amplifier 2 (R2), Stage 3 Modulator (S3), RF Amplifier 3 (R3). In the Tracker tray, components include Duplexer (DU), Transmitter Waveguide (TW), Transmitter Transmission Line (TT), Transmitter Antenna (TA), Receiver Antenna (RA), Receiver Transmission Line (RT), Receiver Waveguide (RW), Duplexer Bypass (DB). In the Processor tray, components include AFC Mixer (AM), AFC Amplifier (AA), AFC Frequency Discriminator (AF), Local Oscillator (LO), Control Circuit (CC), Low Noise Amplifier (LN), Mixer Detector (MD), IF Amplifier (IF), Video Detector (VD), Video Amplifier (VA). In the Indicator tray, components include Sweep Control Circuit (SC), Sweep Generator Circuit (SG), Gate Circuit (GC), Intensity Gate Generator (IG), Cathode Ray Tube (CR).

MENDS, in Figure 4-2, is a simulator created by Charles River Analytics for the Ben Franklin Radar system, used under license. In MENDS, participants can click and open the subsystem (trays) to see the components in each subsystem. Based on their schematic knowledge, the light and switch conditions, participants can decide and click the component that they think is the broken component. Components without power are in grey.

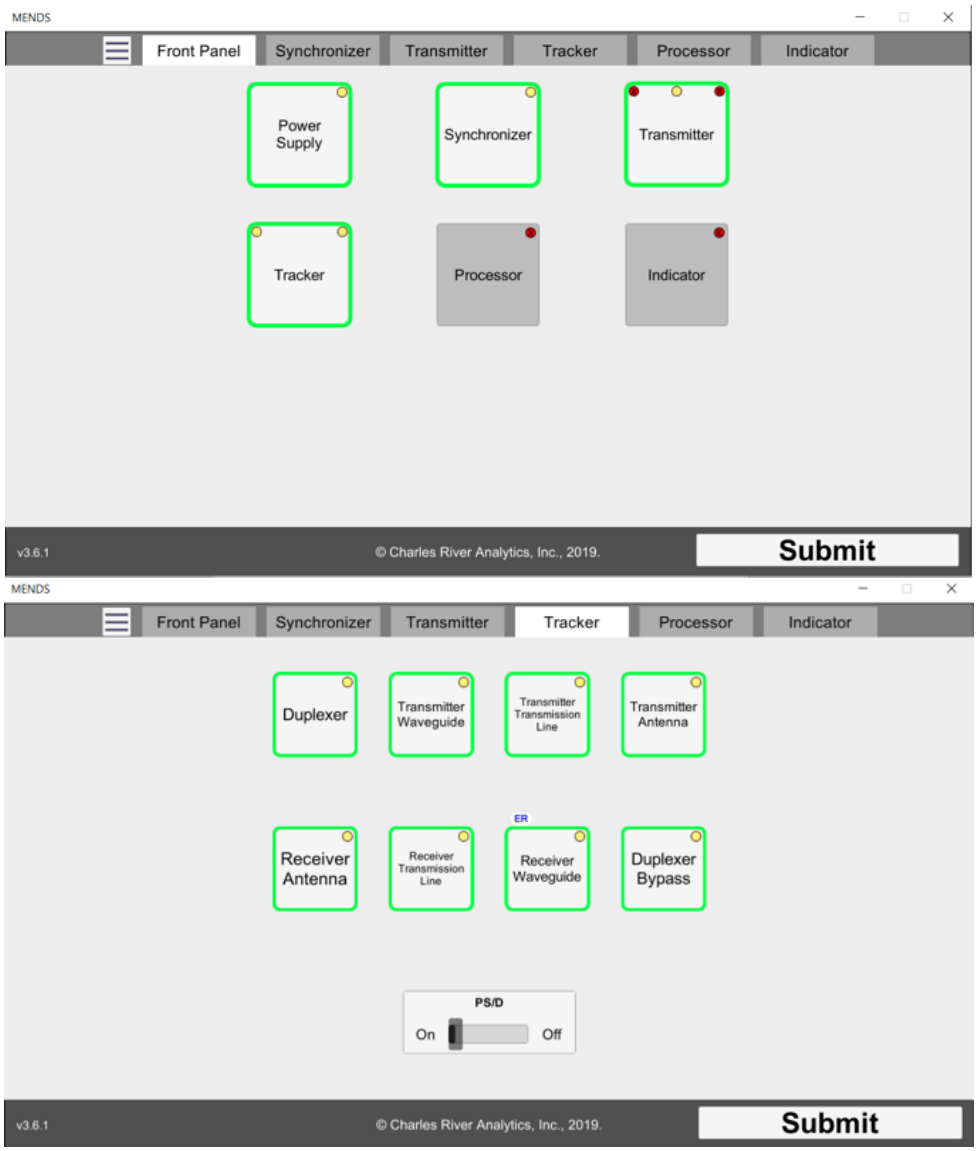



Figure **4-2**: *The MENDS Simulator's Front Panel and One Subsystem*.

**Human Performance Data**

Participants' mouse moves were collected for the modeling and data analysis using the RUI logger (Kukreja et al., 2006; Morgan et al., 2013). To gather data, a user study was run (Ritter et al., 2020). The goal (in addition to studying learning and retention on a complex task) was to identify strategies with a larger number of participants. In the larger study, sessions 1, 2, 3, and 4 are practice sessions; Session 5 is the test session. Data from the test session, when participants tend to have developed their strategies, were used. After data cleaning, 111 participants' data were in the test session. Participants have 5 minutes in Session 1 to practice MENDS and are asked to finish 20 problems in the test session. The number of finished tasks varied, and not all participants finished 20 problems in the test session.

From the mouse clicks of the top six participants in the test session, four strategies were developed and implemented in the Ben Franklin Radar task. The observed time for participants' performance was compared with the predicted time of the strategy models, without and with learning.

In the remainder of this chapter, the modeling of the MENDS task will be outlined, followed by detailed descriptions of the four current strategies, with the Grey Upstream Strategy serving as a technical example. A comparison of the various strategies will then be presented, along with an analysis of human performance in relation to model predictions, both without and with the incorporation of learning. Finally, an example of a participant's match will be included to demonstrate the accuracy of the strategy model in predicting response times. (Participant IDs ranged from 100 to 500, representing different running sessions.)

## Modeling the MENDS Task

Among the suggestions to make modeling easier, as an example, one approach can be modeling with equations extracted from ACT-R and coding in Python. More specifically, this approach builds models using task analysis, keystroke-level modeling, and learning using the power law from ACT-R, implemented in Python, which is named TAKLML.

Figure 4-3 shows how to build a simple task model for MENDS in Python (Ritter et al., 2022). The simple task model used a Panda data frame to store and reflect components' broken status (1: component is fine; 0: broken), light status (1: component has light on; 0: light off), downstream components to give power, upstream components to receive power, switch condition before them (1: switch on; 2: switch off), the number of times the required schematic knowledge have applied by participants. More code details for the functions are in Appendix C.

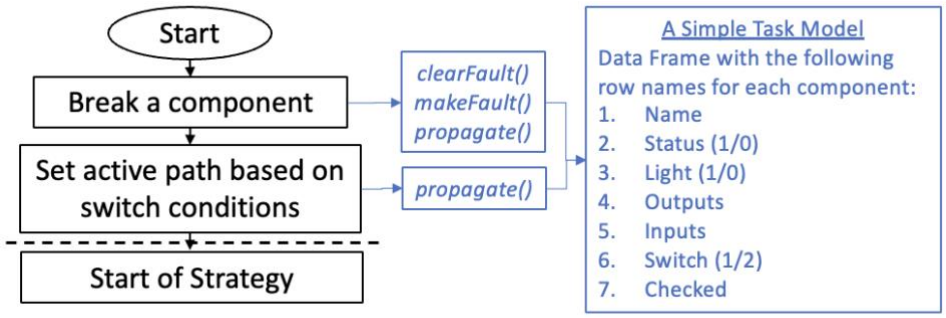



Figure **4-3**: *A Flowchart for the Simple Task Model*. An active path refers to the components receiving power and that are supposed to have lights on.

### Descriptions of the Current Four Strategies

From the mouse moves of participants who had 100% correct rate in the test session, four strategies analyzed by the author were implemented to identify the broken component. All strategies find the

broken component successfully. Those strategies are categorized by four features: their starting point, how the front panel information was used, degree of schematic knowledge used, and degree of interface information used. Variations within one strategy are also possible.

### *The Grey Upstream Strategy*

The Grey Upstream Strategy (GreyUp), shown in Figure 4-4, was based on participants P324, 420, 451, 453. The strategy involves two major steps, finding a grey component as a starting point and tracing upstream in the schematic till finding the broken component. To locate the starting point, users click into the first grey tray using light information from the front panel and identify the first grey component by interface order from left to right and up to down within the clicked tray. Starting from the first grey component, participants use their schematic knowledge to trace up until they identify the broken component, which is the only one with its light off but all its upstream components in the active path are with their lights on.

The front panel information is stored in an array following the tray order on the interface from left to right. The Grey Upstream Strategy model, based on the front panel order, accesses to the components' light conditions from the data frame Some components in the tray are the indicator lights for the tray, so the tray lights on the front panel depend on them. The five trays with their components are stored in five arrays. The model will check the switch conditions to first redefine the active path by modulating the items in the array, which are the components about to be checked. Then the model will check the lights of components in the tray to find the first grey component with light = 0. The upcheck() function takes the index of the first grey component in the data frame and uses breadth search to its "Inputs" that it gets its power from. The function uses a for loop until it finds the component with light = 0 but with all its inputs light = 1, implying that it is the broken one. Function ifChangeTray() takes arguments of the current

and next components and is used inside the upcheck() function to decide if needing to update the tray information. Prediction time is added manually through those process whenever visual encoding, mouse moves and clicks, and mental operation are involved.

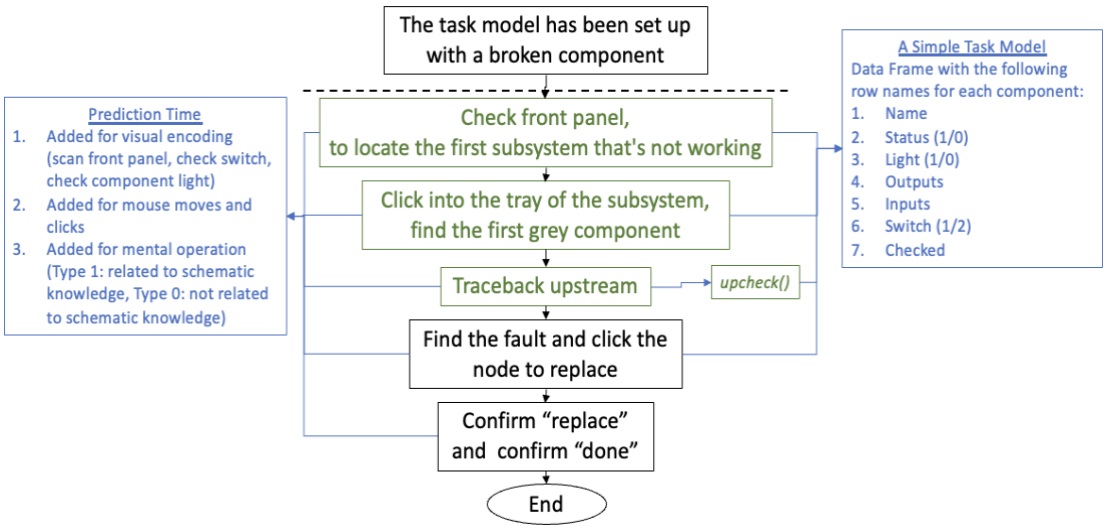



Figure **4-4**: *A Flowchart of the Grey Upstream Strategy*.

### *The One-by-One First Grey Strategy*

The One-by-One First Grey Strategy (OByO), shown in Figure 4-5, was developed based on P448. The strategy walks through trays one by one, stops and clicks on the component when the first grey component in the active path is found. This strategy ignores front panel information and mainly uses interface information — whether the lights are on, and whether the switches are on. The components on the interface follow the order of the circuit flow, so this strategy is a shortcut that takes advantage of the interface component order and can find the broken component successfully without a good schematic knowledge.

The interface information of the front panel and five trays are stored in a two-layered array named subsystem. The active path in the subsystem array is updated based on the switch

conditions. Then a for loop in the subsystem is used to locate the first component with light = 0 based on the interface order. Through the steps, the prediction time was added for visual encoding, mouse moves and clicks, and mental operation time.

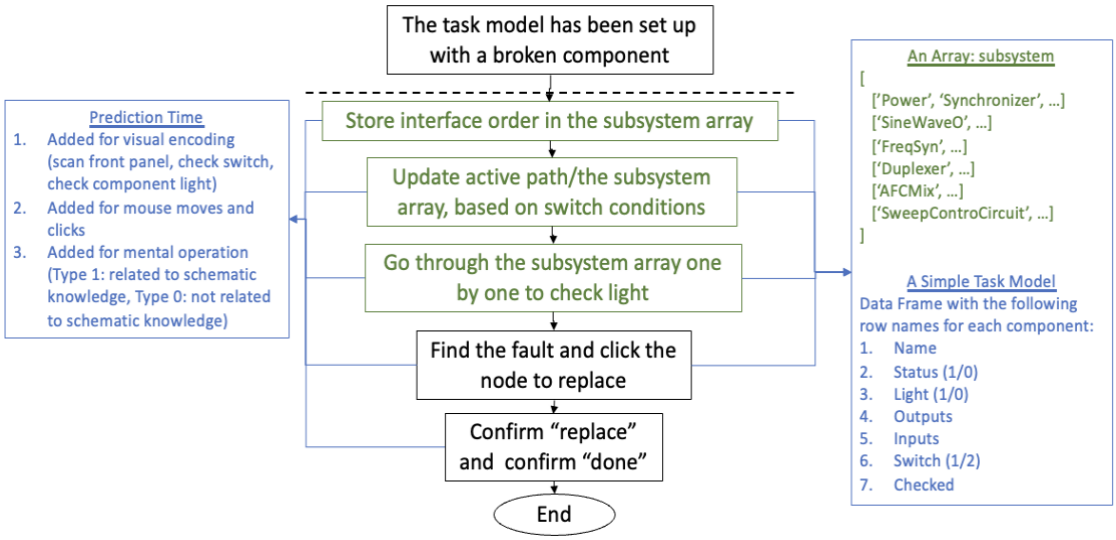



Figure **4-5**: *A Flowchart of the One-by-One First Grey Strategy*.

### *The All-Trays Strategy*

The All-Trays Strategy (AllTrays), shown in Figure 4-6, was developed based on P347. The strategy walks across all trays while deciding the fault, then directly goes to a certain tray to locate the broken component. Components were assigned a number based on their distance from the power supply. The smaller the number, the closer the component to the power supply. The one that has the smallest number among the grey components is the broken one. This strategy ignores front panel information and uses both schematic knowledge and interface information.

The interface information and components' relative distances from the power supply are stored in dictionaries. The relative distances can be 0, which is assigned to the power supply

itself, 1, which is assigned to those directly attached to the power supply, …, and 11, which is 'Cathode' and the furthest from the power supply. A for-loop is used among the dictionaries to search grey components and get the one with the smallest relative distance at the end. Through the steps, the prediction time was added for visual encoding, mouse moves and clicks, and mental operation time.

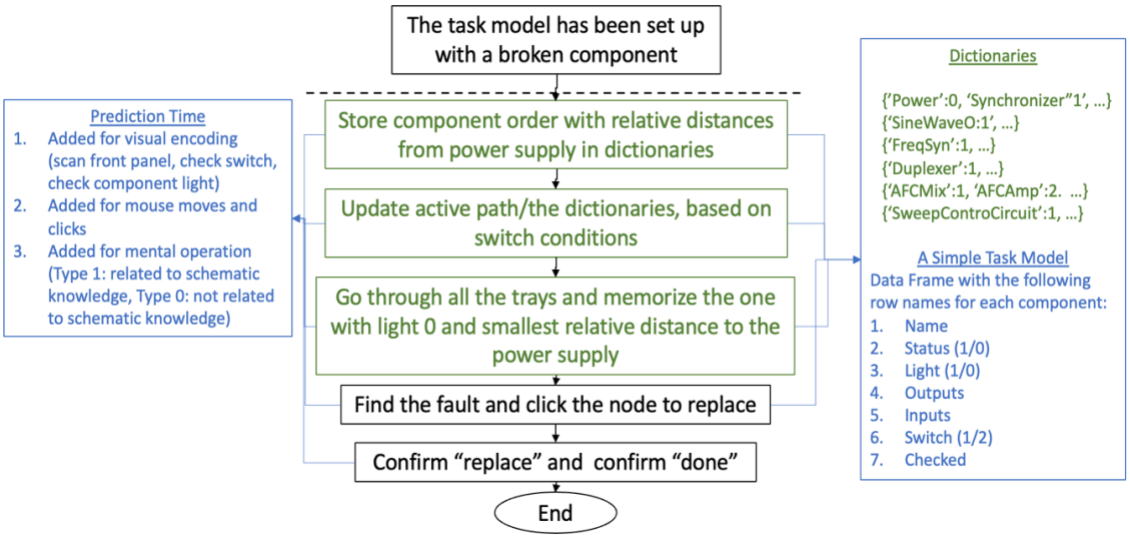



Figure **4-6**: *A Flowchart of the All-Trays Strategy*.

### *The Smaller Lights Strategy*

The smaller lights strategy (SLights), shown in Figure 4-7, was developed based on P396. The strategy uses smaller indicator lights on the front panel to decide the first tray to go; components are grouped by smaller indicator lights. By the combination of indicator lights on the front panel, participants know which tray the broken component is in, then directly go to that tray to click it. This strategy uses smaller indicator lights on front panel and uses both schematic knowledge and interface information. This strategy requires better and higher-level use of schematic knowledge.

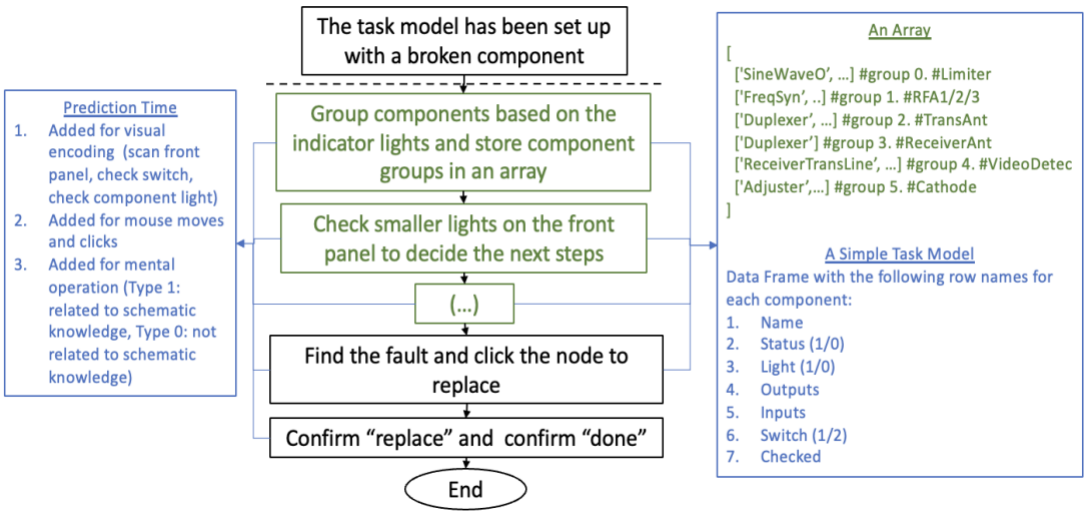



Figure **4-7**: *A Flowchart of the Smaller-Lights Strategy*.

The components are grouped by smaller indicator lights on the front panel. Figure 4-8 shows the detail of how each indicator light influences the next component to check. The prediction time was added for visual encoding, mouse moves and clicks, and mental operations.

```
# set up smaller indicator lights
    # 8 indicators
    # Group 1: Limiter, RFA1/2/3, TransAnt in a line
    # Group 2: ReceiverAnt,VideoDetector, Cathode in a line
    # only Duplexer in both groups. if TransAnt & ReceiverAnt both off, must be Duplexer in Tracker
    # Limiter in Synchronizer
        # if off,
            # in Synchronizer. check Limiter, RCDiff, OverAmp, SineWaveO
        # else. AF1, AF2, AF3 in Transmitter
            # if all off,
                # in Transmitter, check FreqSyn, then check swtich, then S1M/S2M/S3M correspondingly
            # else. TransAnt in Tracker
                # if TransAnt off,
                    # in Tracker.  check TransTransLine, TransWave, Duplexer
    # ReceiverAnt in Tracker
    # if off,
        # in Tracker. check DuplexerBy
    # else. VideoDetector in Processor
        # if VideoDetector off,
            # in Tracker. DuplexerBy, ReceiverWave, ReceiverTransLine
            # in Processor. check IFAmp, ControlCircuit, MixerDetec, LocalOsc, AFCFreqDis, LowNoiseRFA, AFCMix,
                #AFCAmp
        # else. Cathode in Indicator
            # if Cathode off,
                # in Processor. check VideoAmp
                # in Indicator. check SweepGeneratorCircuit, IntensityGate, SweepControlCircuit, GateCircuit
                # in Synchronizer. check Adjuster
# add visual encoding/orienting time on front panel and checking smaller light time
```

Figure **4-8**: *The Pseudocode of the Smaller-Lights Strategy(Screenshot from Code Comments)*.

**How the Models Predict Time**

The models take steps to solve the task based on each strategy, and time is added to each type of action. The time for each step depends on the learning level. The times added with no learning condition are shown in Table 4-1. Under the learning condition, it is assumed that knowledge is transferred to later tasks and the participants solve the later tasks faster based on the repetitive application of the same knowledge. The strategies seem complicated but can be run as a task analysis at the keystroke level and predict times and learning much faster than doing it directly in ACT-R.

Table **4-1**: *Time Parameters Used in the Models.*

| Name | Time (s) |
|---|---|
| Visual encoding | 0.4 |
| Mouse move | 1.1 |
| Mouse click | 0.2 |
| Mental operation | 1.35 |
| Learning rate | 0.4 |

**How the Models Learn**

Models represent learning by modifying the mental operation time. The mental operation time was set as a function and not a constant. The function *mental(type, var2)* gives different times for different types of mental operation and learning levels. The *type* variable includes *type 1* and *type 0*. *Type 1* refers to the mental operation time retrieving a component by using schematic knowledge. *Type 0* refers to any other mental processing not related to schematic knowledge. *Type 1*, retrieving a component, can be faster with learning. For *Type 1*, *var2* is a parameter to assist prediction with learning. For *type 0*, *var2* is a random number because the time is assumed to be fixed, and no learning happens. For situations that assume no learning happens, the mental operation function in models uses constant 1.35 s.

For situations that assume learning happens, the mental operation function in models uses the following formula (Anderson & Schooler, 1991):

$$\boldsymbol{Prediction\ Time} = \mathbf{1.35} \cdot \boldsymbol{n}^{-l} \qquad \textbf{(1)}$$

Where n is the number of times the component has been checked or the number of times the required knowledge of circuit has been used. The value of *l* represents the learning rate, 0.4.

**Comparison of Four Strategies**

Those four strategies have different starting points and use different degrees of schematic knowledge and interface information. They take different times to finish the same task. Figure 4-9 shows the models' predicted times on the 35 faults as if they solved each for the first time (without learning). The figure shows that the strategies are different, and that the time differences they spend also vary by different tasks.

The Grey Upstream Strategy takes the most time in the Transmitter tray, because the switches conditions in the Transmitter tray are the most complicated tray. After deciding which tray to click into, the model directly goes into the certain tray and starts checking from left to right. In each tray, the further the components from the power supply, the more time they take. The One-by-One First Grey Strategy follows the order on the interface and checks their lights one by one, so the time it uses for each component increases and takes a tiny leap when changing trays. The All-Trays strategy takes relatively similar time for all the components because it goes through all the trays and components. The Smaller Lights strategy checks indicator lights from the Synchronizer to the Indicator tray so it takes a longer time to locate the broken components in later trays.

For the two strategies that use light information on the front panel, both the Grey Upstream Strategy and the Smaller Lights strategy take a longer time if the broken component is "Adjuster (AD)" because it does not influence the indicator light on its own tray—the Synchronizer tray but influences the "Cathode"—the indicator light on the Indicator tray.

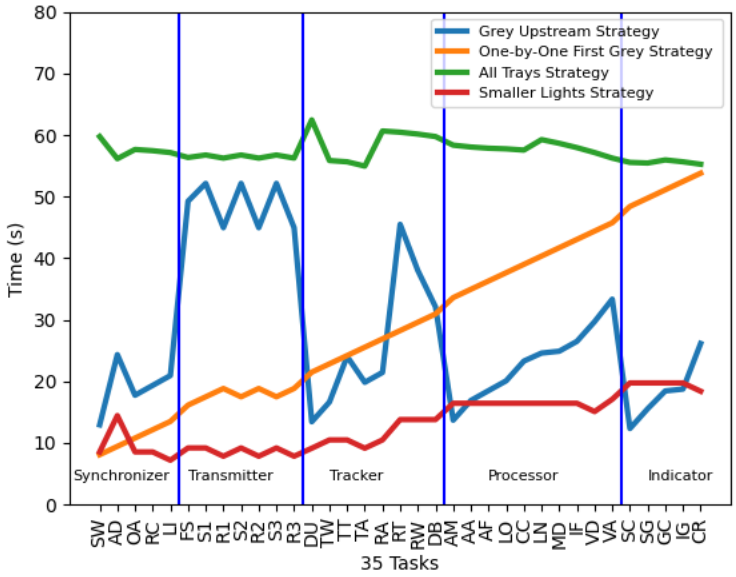



Figure **4-9**: *Strategy Predictions for the 35 Faults Without Learning Across Problems*. The x-axis lists all the 35 faults (not in completion order of any session). Blue line indicates prediction of the Grey Upstream Strategy; orange line indicates prediction of the One-by-One First Grey Strategy; green line indicates prediction of the All-Trays Strategy; red line indicates prediction of the Smaller Lights Strategy.

Figure 4-10 shows the predictions for the strategies for the 20 problems without learning for the problem series in the test session, showing again that the strategies are different and predict different amounts of learning based on what parts of the schematic and interface participants use. For example, the four strategies take from less than ten seconds to near sixty seconds to finish the same task "SW". Those strategies give different predictions. Their plots are very different.

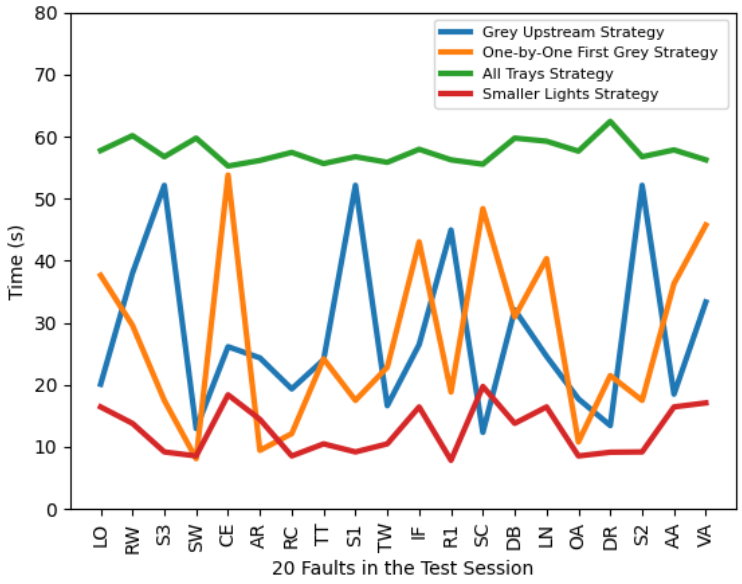



Figure **4-10**: *Strategy Predictions for the 20 Problems Without Learning in the Test Session*. The x-axis shows the same order of the 20 tasks in the test session. Blue line indicates prediction of the Grey Upstream Strategy; orange line indicates prediction of the One-by-One First Grey Strategy; green line indicates prediction of the All-Trays Strategy; red line indicates prediction of the Smaller Lights Strategy.

## Comparing Human Performance and Model Predictions

The results of the comparison between the strategies and participants, both without and with learning, will be discussed next. The developed strategies were compared to the human performance data collected during the test session. The models were used to solve tasks in this session, having been trained on the previous sessions that participants experienced prior to the test. The predicted times were then compared to the observed times of the participants. A participant is considered to fit well to a strategy if they have $R^2$ with a p-value < .05.

**Without Learning**

14 participants' behaviors match a strategy ($p < .05$; $R^2$ varied) without learning in the test session. All the other participants did not have a well-matched strategy from current available strategy models. Table 4-2 shows only part of the data, under the no learning condition, as an example. These represent the best ones, ranking from high to low correlations for AllTrays, GreyUP, OByO, and SLight. Some strategies were matched well, such as the Grey Upstream Strategy (GreyUp) and all trays strategy (AllTrays). The one-by-one strategy (OByO) and the smaller light strategy (SLight) did not match well. Negative correlations are shown as 0's in the table for clarity. These negative correlations were not significant correlations.

Table **4-2**: *Correlations for Participants and Strategies Without Learning*. Example well-fit participants, the four strategies, and their corresponding R2. Those with p-value <.05 have a * attached. Negative correlations are shown as 0's for clarity.

| PID | AllTrays | GreyUp | OByO | SLight |
|---|---|---|---|---|
| 378 | .322* | 0 | .024 | .026 |
| 387 | .067 | .315* | 0 | 0 |
| 404 | .021 | .233* | 0 | 0 |
| 413 | 0 | .492* | 0 | 0 |
| 429 | 0 | .342* | 0 | 0 |
| 428 | 0 | .428* | 0 | 0 |
| 352 | .325* | .068 | .004 | .015 |

**With Learning**

69 participants' behaviors match a strategy with learning in the test session. All the other participants did not have a well-matched strategy from current available strategy models. Table 4-3 shows only part of the data, under the learning condition, as an example. These represent the best ones, ranking from high to low correlations for AllTrays, GreyUP, OByO, and SLight. Some strategies were matched well, such as the Grey Upstream Strategy (GreyUp) and all trays strategy

(AllTrays). The one-by-one strategy (OByO) and the smaller light strategy (SLight) did not match well. Negative correlations are shown as 0's in the table for clarity.

Table **4-3**: *Correlations for Participants and Strategies With Learning.* Example well-fit participants, the four strategies, and their corresponding R2. Those with p-value <.05 have a * attached.

| PID | AllTrays | GreyUp | OByO | SLight |
|---|---|---|---|---|
| 361 | .763* | .082 | 0 | 0 |
| 407 | .737* | .012 | .033 | .019 |
| 350 | .594* | .095 | .055 | .012 |
| 355 | .577* | .062 | 0 | 0 |
| 352 | .572* | .078 | .004 | .017 |
| 385 | .563* | .095 | .009 | 0 |
| 413 | .009 | .526* | 0 | 0 |

Table 4-3 shows that models with learning match participants' performance better than without learning, implying that the participants were learning within and across sessions and that the power law can help the prediction performance of models. The strategies are very different, so only one strategy per participant fit well if there is a match.

Figure 4-11 shows the match of PID 413 as an example. PID 413 has $R^2$ of .498, without learning and an $R^2$ of .518, with learning for the Grey Upstream Strategy. The red dotted line is the observed time from human data; the blue solid line is the predicted time without learning; the black solid line is the predicted time with learning and was trained with previous sessions' faults.

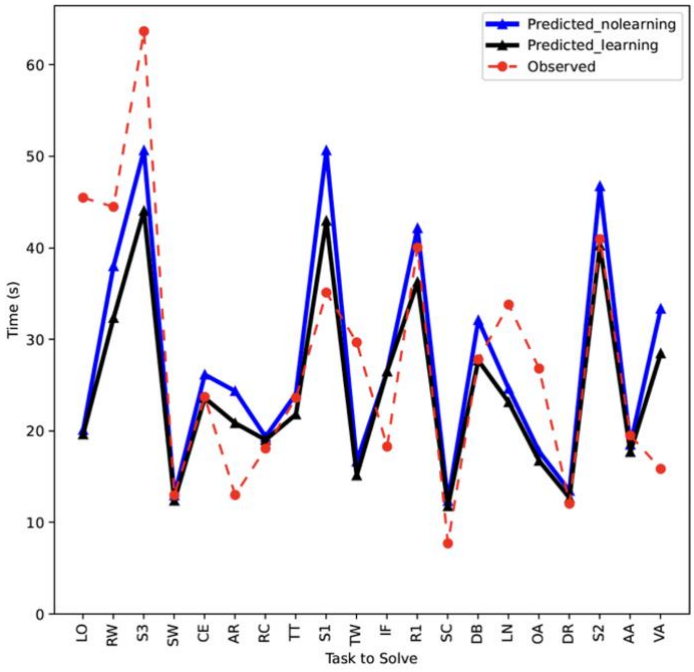



Figure **4-11**: *An Example Participant (PID 413).* A comparison of human data, the Grey Upstream Strategy model with learning, and the Grey Upstream Strategy model without learning. Blue line indicates prediction of the no-learning model; black line indicates prediction of the learning model; red line indicates observed human performance.

**Summary of Comparison**

The models predict the time required to replace the broken component. Based on the mouse clicks of the top participants in the test session, four strategies were developed and implemented in the Ben Franklin Radar task. These strategies utilized hierarchical task analysis, the Keystroke-level model, and the power law. Each strategy was then executed in the test session under both the without-learning and learning conditions to obtain predicted times. Human performance data indicate that, over time, participants learned and developed their strategies, resulting in a higher correct rate. 14 participants' behaviors reliably match a strategy without learning in the test session. 69 participants' (62%) behaviors reliably match a strategy with learning in the test

session. The numbers of matches is much larger with learning, supporting the power law of learning by strategy (Delaney et al., 1998).

## Discussion and Conclusion

The previous section shows the diversity of strategies employed by participants in a fault-finding task. Four strategies were identified, each with distinct approaches from starting points, the use of schematic knowledge, and engagement with the task interface. This section further discusses the implication of this modeling task and the TAKLML approach.

### Discussion for the MENDS Task

Four strategies were identified in problem-solving and in different learning conditions. Different strategies have different starting points, different ways of using schematic knowledge, and interface information. They took different times, and all found the correct solution. With the assumption of only one standard strategy, we may mistakenly regard many of the participants as misbehaving when they apply a different strategy. In a worse case, we may interrupt them at the wrong time, believing that we are correcting and helping them.

One should indeed be careful averaging behaviors. Participants use different strategies within this task. Averaging over strategies will distort results and hide important information. Four strategies were identified and used for the complex fault-finding task to predict human performance. While more than four strategies exist, some significant negative correlations were also observed, suggesting that further exploration may reveal counterintuitive strategies.

Additionally, a fifth strategy, called the random-start strategy, has been developed. Participants using this strategy would pick a random tray to click and then decide to go down or

up the circuit based on the light conditions on the tray. This strategy requires further refinement to evaluate all possible starting points for calculating the correlations between predicted time and human performance time.

The current strategy models are from the top six well-performed participants. The rest of the participants made many more errors during the fault-finding process. If the strategy models consider errors, the match of participants and strategies may increase. A simple error model was implemented and indeed got higher $R^2$ (Wang & Ritter, 2023b). More and complicated error models are worth further explored.

Moreover, currently, it is assumed that participants were using the same strategy across a session, but some participants may have applied various strategies within one session for different tasks. It is also assumed that they apply the strategy perfectly without variation. It requires them to have the necessary working memory and schematic knowledge to help them finish all the steps of a strategy. In this case, errors, or lapses, or changes of strategies within the same session were not modeled. Modeling those can be my future steps.

This study urges an awareness of understanding diversity of behavior. Modeling and understanding different strategies will help understand the breadth of human behavior. It will also help model individuals more accurately rather than just an average, which may not exist.

**Discussion for Modeling with TAKLML**

This study showcases that modeling with TAKLML (Task Analysis with Keystroke-Level Modeling and Learning) can work. Compared to traditional frameworks like ACT-R, TAKLML simplifies the model-building process, making it more accessible to researchers. The user-friendly nature of Python and the Visual Studio (VS) Code integrated development environment (IDE) enhances the experience, allowing for an intuitive engagement with the modeling tasks. This ease

of use can lower the barriers for researchers new to cognitive modeling. A quantitative comparison could be conducted in the future through structured experiments, allowing researchers to systematically assess the efficacy of different modeling approaches. Moreover, researchers can change the learning constant for different subtasks (Kim &Ritter, 2016).

## Summary

This chapter presented a cognitive modeling study to show that modeling in Python with task analysis can accomplish a cognitive modeling task. This chapter first introduced the MENDS task to be modeled and human performance data to be compared. Then, the chapter presented details of modeling the MENDS task, including descriptions of the current four strategies for the task, how the models predict time with explanations and equations used, how the models learn, and comparison among the four strategies. The chapter then presented the results of comparing human performance and model predictions, under different modeling situations of “without learning” and “with learning”.

# Chapter 5

# Modeling Errors with TAKLML

This chapter documents another cognitive modeling study as a performance demonstration. A cognitive modeling study is presented to illustrate how a difficult task was modeled in Python and how the difficulty of modeling participants' errors was minimized through the use of task analysis and Python. Building on the MENDS task from the previous performance demonstration, the match between the error model and the human performance data has been successfully improved. This chapter demonstrates that modeling in Python, combined with data analysis, can effectively address tasks that are typically considered challenging within the modeling field. This chapter is based on a publication (Wang & Ritter, 2023b).

## Introduction

Performance errors can be a great source for understanding how people learn and improve in fault-finding tasks. Error generation and recovery are important determinants of real performance with underappreciated and understudied implications, in that errors can on one task be 3% of actions yet making and correcting them take 20% of time (Landauer, 1995). In a text-editing task, errors attribute to 35% of expert's time and over 80% of the variance in time. Considering errors would thus be very helpful to understand where time goes in tasks and help build models. It would be important to add error generation into those models to better predict time.

Norman (1981) and Reason (1990) had studied action slips and human errors. Norman categorized action slips into three: (a) errors in the formation of the intention (e.g., mode and description errors); (b) faulty activation of schemas (e.g., loss of intention and misordering of action components); and (c) faulty triggering (e.g., spoonerisms, blends, and intrusions of

thoughts). Reason provided a broader framework that include three basic error types: skill-based skips and lapses, rule-based mistakes and knowledge-based mistake. This dissertation appreciates the existing framework and categorization. However, for practical purposes, this dissertation introduces a bottom-up approach and encourages researchers to analyze their own data and extract errors from their own data to make the modeling process smoother.

In this chapter, the process of building an error model is presented as an example. First, error patterns are analyzed, and an appropriate error type is chosen for modeling, using data from a large study (Ritter et al., 2022). Errors are analyzed for each task and each participant. For each task, the frequency of misreplaced components is presented, which serves as one definition of error, and misreplacements are further categorized into four types. An error model is also introduced—an updated fault-finding strategy model that generates errors and corrects them, resulting in a higher correlation with participants' actual performance compared to the original model.

Error modeling is recognized as a challenging area within the research field. The decision to model errors further demonstrates the performance of the approach outlined in Chapter 4, showcasing its capability to address difficult tasks. In Chapter 4, the approach utilizes an equation from ACT-R in Python, based on suggestions for simplifying modeling. This approach incorporates Python alongside task analysis, keystroke-level modeling, and learning using the power law from ACT-R.  With the same approach, this chapter continues to model errors, detailing both the process and the results.

## Task Description

The task is to find a single broken component in the Ben Franklin Radar system (Ritter et al., 2019). Figure 5-1 shows the schematic that is based on an actual amateur radar system. It is the

same schematic in Figure 4-1 with some components circled. Those circles are explained in the result. This study uses the same task as the study in Chapter 4. It is organized into five subsystems, including the Synchronizer, Transmitter, Tracker, Processor, and Indicator. The human performance data were collected from a previous study (Ritter et al., 2022).

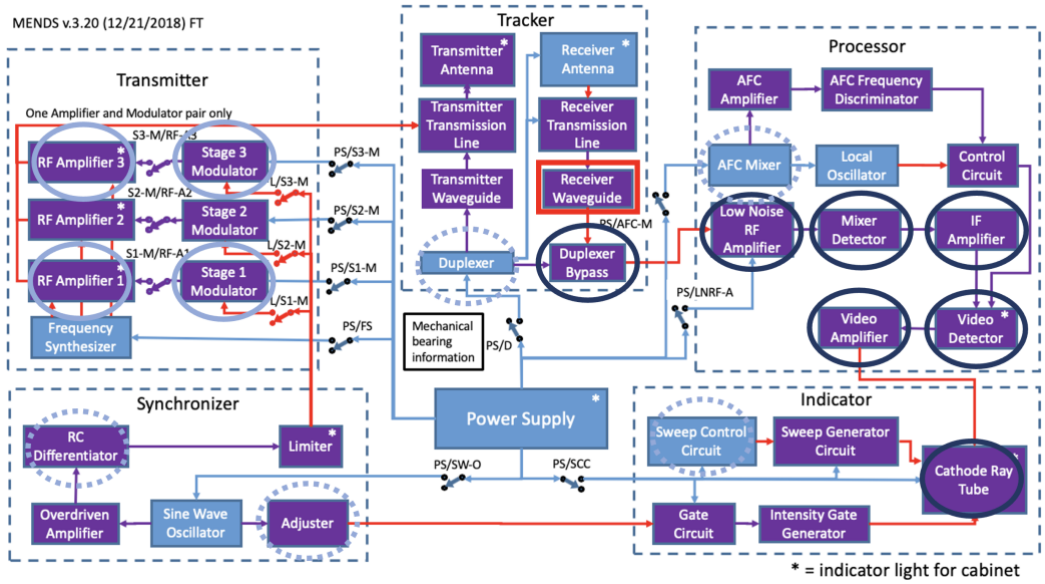


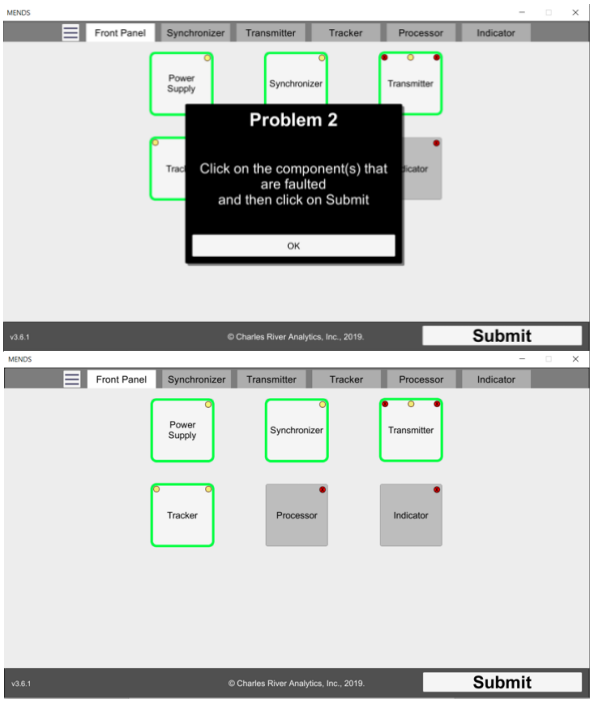

Figure **5-1**: *The Ben Franklin Radar Schematic*. Circles and their colors are explained in the results section.

The Ben Franklin Radar system is implemented as a simulator (MENDS) with two levels of fidelity in Unity. The simpler 2D version was used in this study. Participants were asked to study the schematic and then finish tasks on MENDS.

For each task, one of the components in the circuit, excluding the power supply, was broken. Participants were asked to identify the broken one. When identifying, they used a mouse to change trays (similar to tabs) on MENDS to gain the interface information (the light and switch conditions). They also recalled their circuit schematic knowledge to help decide the broken one. Then, they clicked on the component they identified, confirmed replacing the component, and confirmed finishing the task. If a participant found the real broken component, after clicking

"Replace", the component would be updated with a green square line and a yellow circle, implying that the broken component was successfully replaced (see detailed steps in Figure 5-2). Replacing an unbroken component made no change to the light condition, the component of which would stay grey with red circle at upper right corner.

If the participant was sure they replaced the broken component based on the light change, they could press the button "OK" when the screen popped up a question "Are you sure you've done?". The screen would then pop up a message saying "You replaced 1 faulty component. You left 0 faults un-fixed. You replaced 0 components that were not faulty" with an "OK" button. When the participant clicked it, the trial ended.



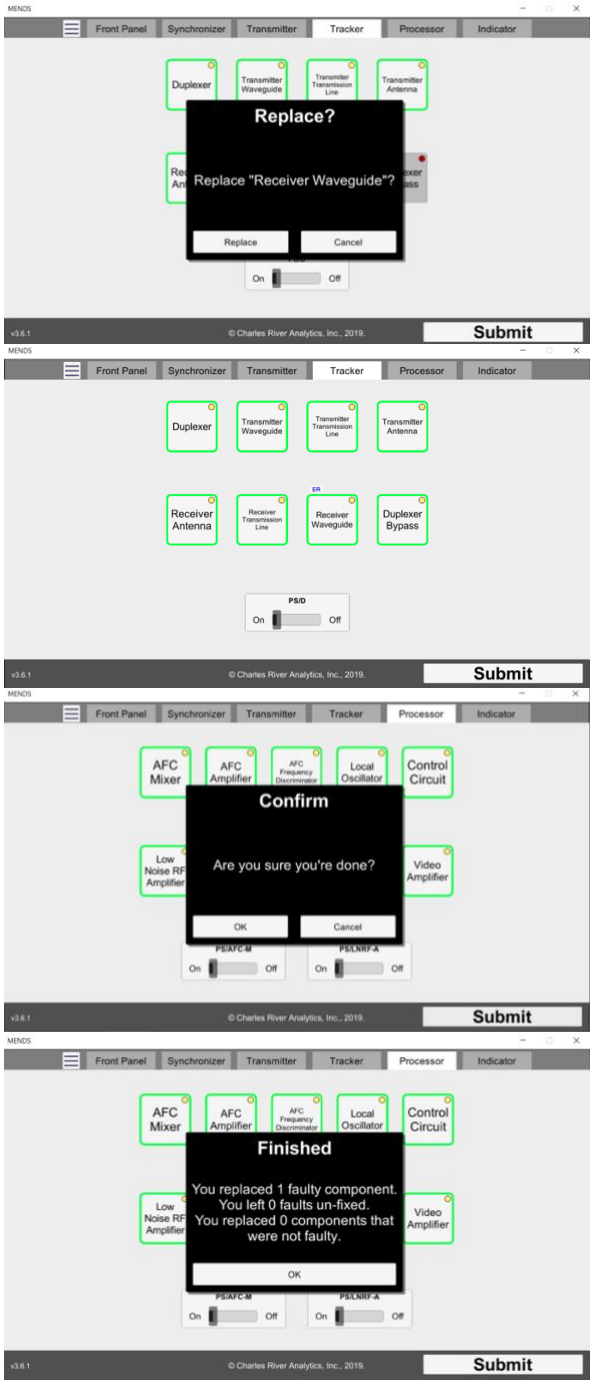



Figure **5-2**: *MENDS Interface Screenshots of the Replacement Process, without Misreplacements.*

**Data of Human Performance Errors**

Participants in the study (N = 111) learned about the troubleshooting task with the task system and its interface through an online tutor (Ritter et al., 2022). Data from the last session, the testing session where they saw 20 fault problems, were used. Further detail not necessary for this paper are in the technical report (Ritter et al., 2022). This previous report only reported averaged error rates.

Some strategies that participants may use were analyzed (Wang & Ritter, 2023a) and we presented a partial error model (Wang & Ritter, 2023b). Here is a more detailed process of how researchers could dig deeper into errors and build the error generation and correction model.

Participants made errors by replacing an unbroken component. The worst case is that participants could replace several unbroken components and click "OK" to confirm that they were done, without ever finding and replacing the broken component. In this case, the feedback by the simulator interface would show "You left 1 fault(s) un-fixed". Those are errors not corrected by the participants. my definition of errors does not include variations of the strategies nor repetitive or extra steps in the process. Those extra or wrong steps in the process, as long as those do not lead to the final replacement with feedback from the simulator interface, are so far not considered as errors but still just part of the problem-solving journey and may be part of the strategies they use (e.g., a more thorough search that opens unnecessary subsystems). The analysis begins with an examination of misreplacement, specifically focusing on instances when unbroken components were replaced.

## Analysis of Human Performance Errors

This section begins with an error analysis for each type of fault encountered during the test session, detailing the difficulty level of tasks, the most frequently misreplaced components, and a specific case involving the RW component. Subsequently, an analysis for each participant (PID) is presented, which informed the development of the error model constructed for this study.

### Error Analysis for Each Task

The task acronyms are taken from the schematic (See Figure 5-1). When a component is the broken one, and the task is to find it, its acronym is used as the task name. Figure 5-3 and Figure 5-4 help understand errors for each task through the unfixed and the wrong attempts (participants can make several wrong attempts during the process and leave the broken component unfixed once for each task). Figure 5-3 shows the total number of unfixed tasks. Figure 5-4 shows the total number of wrong attempts that participants made trying to solve the task. The comparison of the two images reveals the correlations between the number of unfixed attempts and incorrect attempts. The tasks with higher number of wrong attempts are more likely to not end up being fixed. In other words, those that are not fixed at the end usually have gone through several wrong attempts. Potential reasons might be that those tasks are in general harder to fix. They may require the part of the schematic knowledge harder to memorize and/or may require more steps in the problem-solving strategy that participants tend to miss.

#### *Difficulty Levels of Tasks*

The tasks were categorized into four difficulty levels (Level 1 is hardest and Level 4 is easiest), in the color patterns of Figure 5-3 and 5-4.

More specifically, Figure 5-3 shows the total number not being fixed for each task in Session 5. For example, the total number of not being fixed for component S3 was 14, representing those 14 participants failed to identify the broken S3 to fix it. The total number of not being fixed for TW was 1, implying that only 1 participant failed to identify TW to fix it. The total number of not being fixed for TT was 0, implying that all participants successfully identified the broken component TT. The tasks could be categorized into some difficulty levels, using the total number of not being fixed. Assuming that the more difficult a task was, the more participants failed the task, the total number of not being fixed gives rise to the difficulty levels.

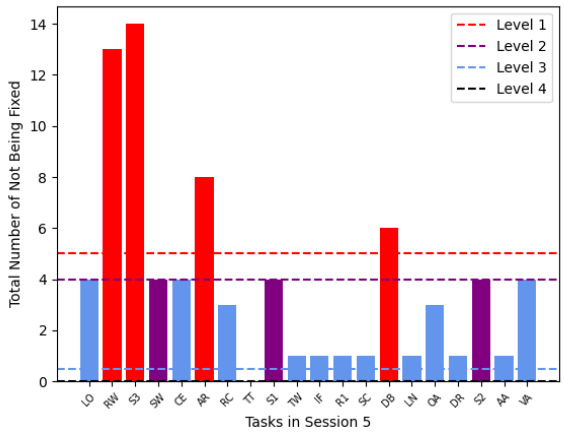



Figure **5-3**: *Total Number of Broken Components Not Fixed by All Participants for the 20 Tasks of the Test Session*. Red horizontal line and bars indicates difficulty level 1; purple horizontal line and bars indicate difficulty level 2; blue horizontal line and bars indicates difficulty level 3; black horizontal line (overlapped with x-axis) indicates difficulty level 4.

Figure 5-4 shows the total number of wrong replacements in Session 5. For example, the total number of wrong replacements for S3 was more than 200, implying that all the participants tried more than 200 times in total to identify the broken S3 to fix it. The total number of wrong replacements for TW was around 50, implying that all the participants tried around 50 times in total to identify the broken TW to fix it. The total number of wrong replacements for TT was around 25, implying that all the participants tried around 25 times in total to identify the broken

TT to fix it. Assuming that the more difficult a task was, the more wrong replacements participants made in the task, the total number of wrong replacements could imply the difficulty levels. The tasks could be categorized into some difficulty levels, using the total number of wrong replacements.

Therefore, considering the cues in both figures, four distinct difficulty levels emerged.

**Level 1**: RW, S3, AR, DB. (Red: on top of both figures.)

**Level 2**: SW, S1, S2. (Purple: popped out in Figure 5-4 as Level 2.)

**Level 3**: LO, CE, RC, TW, IF, R1, SC, LN, OA, DR, AA, VA. (Blue: those in between Level 2 and 4.)

**Level 4**: TT. (Black: the only one that got fixed by all participants, and thus has 0 not fixed in Figure 5-3.)

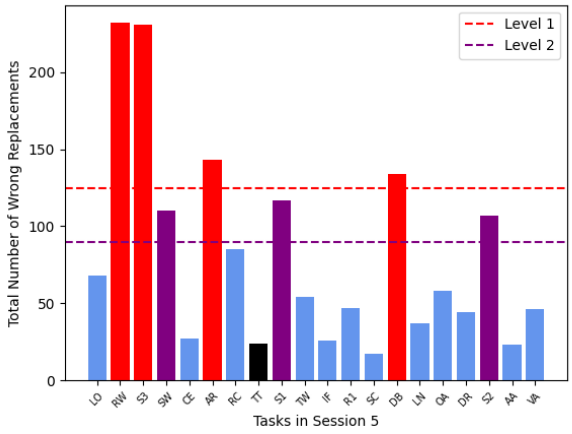



Figure **5-4**: *Total Number of Wrong Replacements by All Participants for the 20 Tasks*. Level 3 and 4 are relatively hard to see in this figure so no horizontal lines. Red horizontal line and bars indicate difficulty level 1; purple horizontal line and bars indicates difficulty level 2; blue bars indicate difficulty level 3; black bar indicates difficulty level 4.

### *Most Likely Misreplaced*

The frequency of misreplaced components for each task was calculated. Analyzing these misreplaced components enables the examination of behavior patterns related to errors. Figure 5-5 provides a summary of these frequencies. The y-axis lists 20 tasks in Session 5. Each line refers to a task. For example, line VA refers to the task that participants need to identify the broken component VA to fix it. The x-axis lists all the components in the circuit that participants could click to fix. If they fixed the broken component VA when VA is broken, they finished the task successfully. However, if they clicked other components that were not broken or VA, they misreplaced a working component. Figure 5-5 collected the frequency of misreplaced components for each task.

In Figure 5-5, a broken S3 component (line 3) is often wrongly replaced with S1, R1, S2, and R2 components. A broken AR component (line 6) is mostly misreplaced with GC, IG, CR. A broken DB (line 14) is mostly misreplaced with LN. A broken VA (line 20) is mostly misreplaced with CR. As a more specific example, a broken RW (Receiver Waveguide, line 2) is mostly misreplaced with LN (Low Noise RF Amplifier) (62), CB (25), VD (25), CR (24), MD (23), VA (23), IF (20), S3 (7), S1 (6), R1 (5), R3 (4), AM (2), AD, RC, SU, OC, VA (1). The numbers in parenthesis are their frequency of being misreplaced. Those frequencies may be due to their relative positions on the schematic and/or interface. The closer they are to the broken components in either the schematic or the interface, the more likely they can be misidentified as the broken one.

To analyze reasons of misreplacement, RW (line 2) with the highest frequency of misreplacement could serve as an insightful case. The following subsection delves into RW in greater detail.

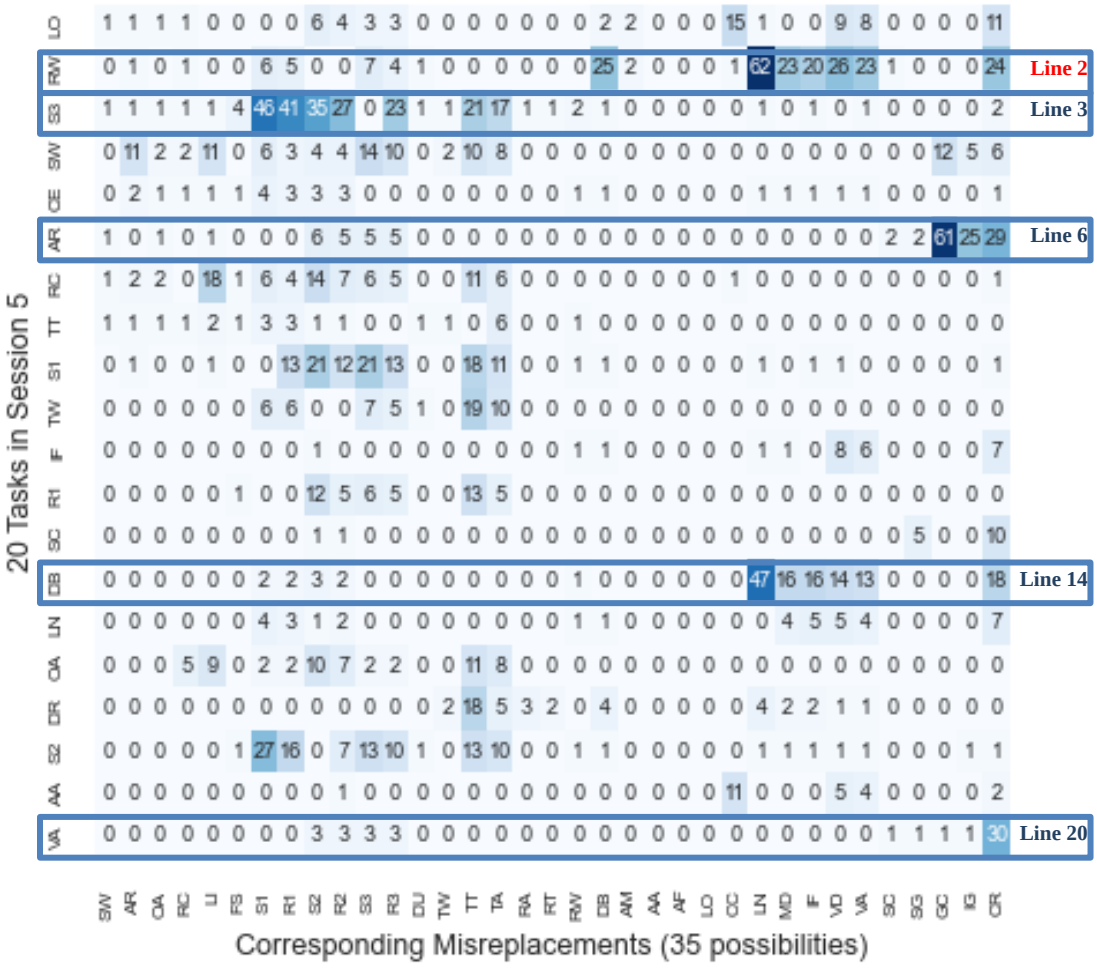



Figure **5-5**: *Tasks in the Order of the Test Session and Their Corresponding Misreplacements with Frequencies.* The darker the cell color, the more frequently they got misreplaced. The numbers in the cell are their exact frequencies. Lines 2, 3, 6, 14, and 20 are enclosed in squares to assist in their identification.

***The Case of RW***

Figure 5-1 and 5-6 provided detailed presentations of misreplacements of the RW (Receiver Waveguide, Line 2) fault. Different colored and solid/dotted lines indicate varying levels of misreplaced frequency. Red lines outline the broken component. Dark blue circles outline those that have misreplacement frequency from 62 to 20, light blue solid lines circle those with

frequency between 7 to 4, and light blue dotted lines circle those with frequency of 2 or 1. Circle colors apply to both Figure 5-1 and 5-6. Figure 5-1 may still be unclear as to understand why participants mistakenly clicked those other components for the task of the broken component RW, so Figure 5-6 of the MENDS interface screenshots was provided to help clarify and understand those errors. Errors were categorized into four types, based on the light condition and the relative position of the misreplaced component on the tray. Details of the categorization are described below.

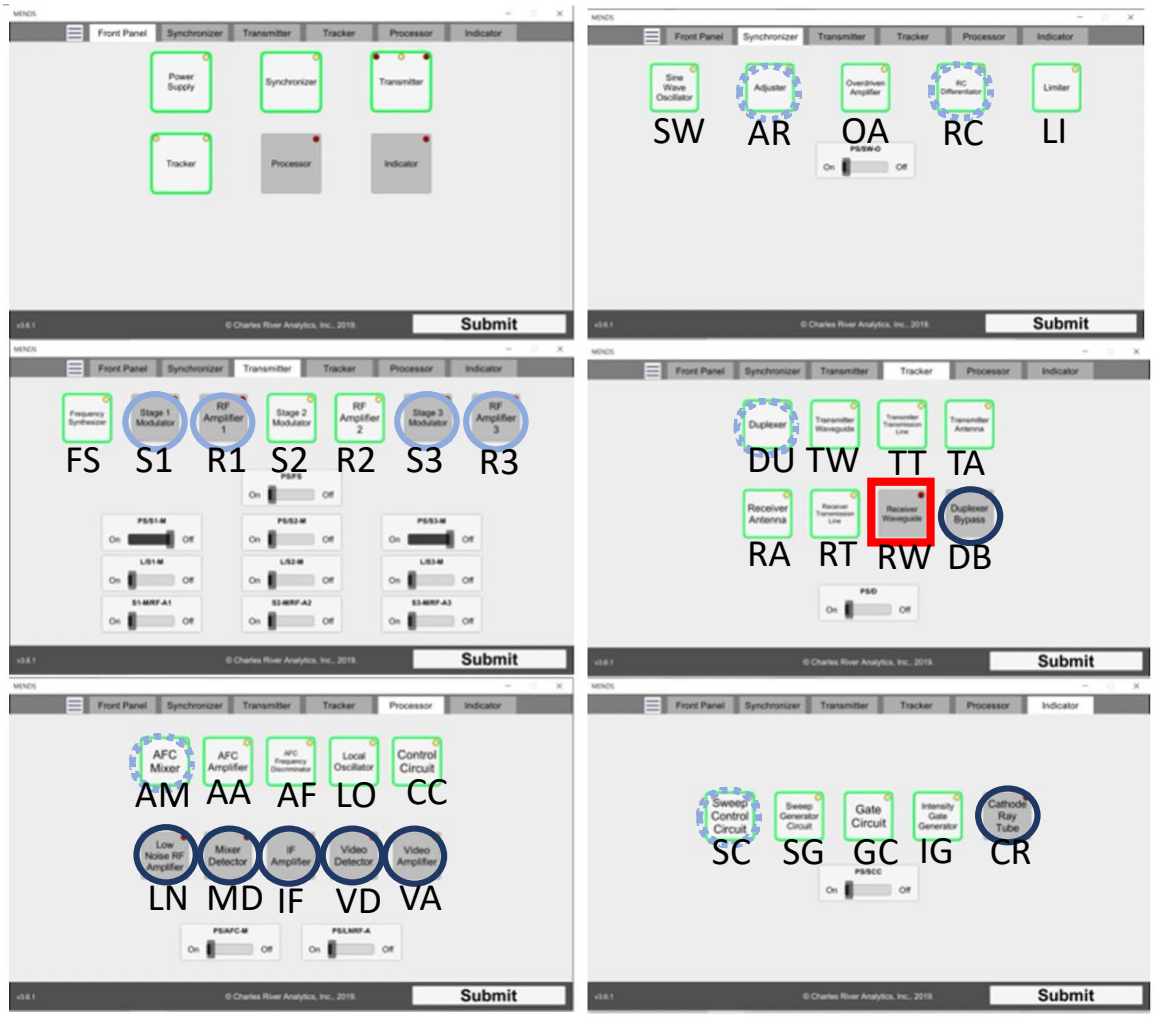



Figure **5-6**: *MENDS Interface Screenshots of All the Six Trays*, with high frequently misreplaced components circled. Abbreviations are provided below each component in cases where the screenshots may be unclear.

### *Categorization of Errors*

Errors were categorized to help decide the specific and appropriate type of errors to model. The process was presented as a pattern. It may better show how later researchers can come up with their own error categorization and build their error model, rather than using an already existing categorization that seem generalizable but may be hard to apply to each different task and their modeling process. By analyzing the case of RW (Receiver Waveguide) in Figure 5-6, four potential types of errors were categorized based on their source, schematic knowledge errors, strategy errors, interface information ignorance, and other errors. The following are a more detailed description of those error types:

**Schematic Knowledge Errors**: the participant failed to fix the broken component due to not being able to recall the correct schematic knowledge. They may forget the detail they need or may memorized the wrong one.

**Strategy Errors**: the participant failed to fix the broken component because they did not have a working strategy. They made wrong decisions about which tray or component to click even though they may have the schematic knowledge and interface information ready.

**Interface Information Ignorance**: the participant ignored some necessary interface information. For example, they may forget to check the switches. And they still clicked components that are not receiving power/with an off switch.

**Other Errors**: this type includes all the other errors, including that the participant may intentionally mess around, try out, just randomly click without paying attention, etc. For example, the participant may click on unbroken components that have lights on and are functional. They may just not care at all or not pay attention.

Participants may just click the wrong component, but it may be due to a range of reasons. The types of errors are distinguished based on the source or intentions, and this helps us establish

my error model. There are more unnecessary steps that participants may take, and they may also be due to the above four listed errors. For example, switching between trays several times may be due to schematic knowledge errors, but the participants autocorrect themselves before they take further actions.

**Error Analysis for Each Participant in the Test Session**

Next errors for each participant were examined, which helps to decide whose error to model. Figure 5-7 shows the number of fixed and unfixed tasks in the test session, for each participant (PID). Blue refers to repaired number; orange refers to not fixed with the specific number at the top of the figure. Most of the participants fixed all faults at the end, leaving only a few orange/unfixed. However, the amount of time participants spent making incorrect attempts during the fault-finding process may not be immediately apparent.

Therefore, Figure 5-8 is included to illustrate this analysis. Figure 5-8 shows the number of right and wrong replacements or attempts during the process. The number of incorrect attempts made by many participants exceeded their correct attempts, indicating a need to examine these wrong attempts more closely to gain insights into time allocation and to improve the modeling of their performance.

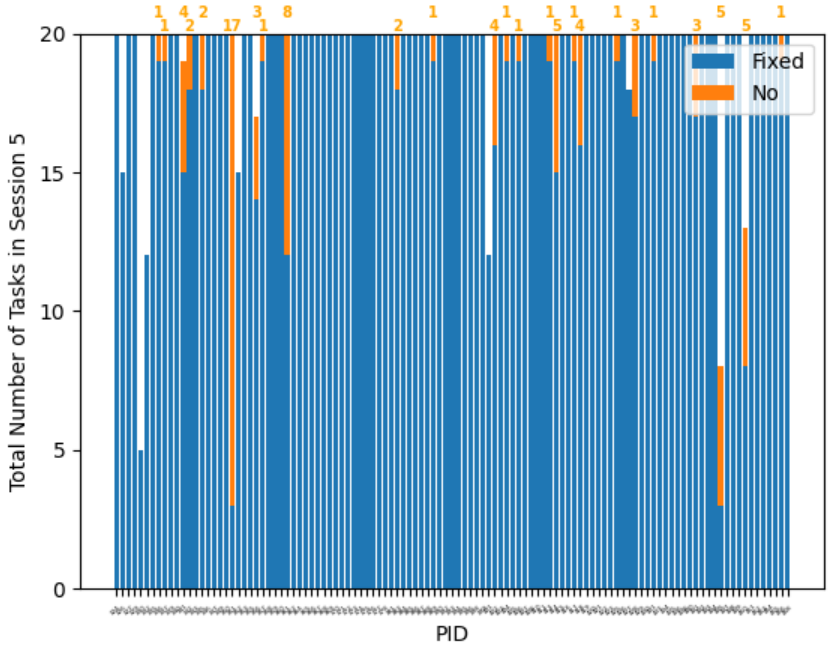



Figure **5-7**: *Total Number of Tasks in the Test Session, for Each Participant (PID)*. Blue refers to fixed number; orange refers to not fixed with the specific number at the top of the figure.

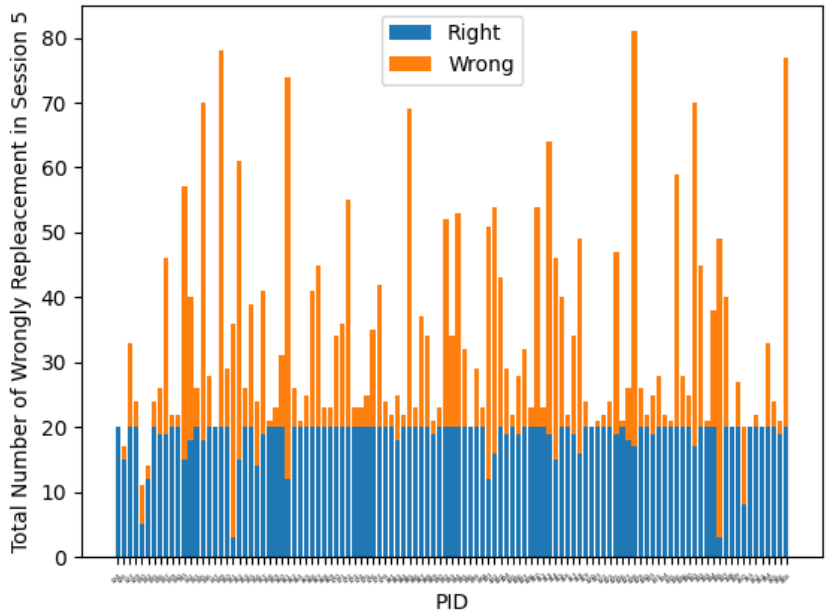



Figure **5-8**: *Replacements in the Test Session for Each Participant (PID).* Blue bars represent the correct replacement; orange bars show the wrong replacement.

Considering both the figures, we may understand better participants' performance. For those that do the best, 6 out of 111 participants replaced all the broken components and only the broken components, with a low frequency of misreplacements and high frequency of fix rate. my implemented strategies were from analyzing their mouse clicks to understand their good strategies used.

For those that do good, they had on average three wrong replacements per task, but still managed to find the correct broken component because their fixed number was also near 20. Those with a high misreplacement and high fix rate did not give up.

For those that are not doing well, some seem to easily give up or not pay enough attention so that they did not try much. Some seem to try or guess very hard that they made many wrong attempts but still failed to fix the broken component at the end. They may lack schematic knowledge or a good strategy if they have a high frequency of misreplacements and low frequency of fix rate.

PID 421's behavior was modeled, because the participant had only one unfixed component across the session and may have a well-established strategy.

## Modeling Human Errors

This section outlines the modeling of the single error for Participant ID 421. PID 421 matches with All-Trays Strategy with trained learning in the test session. In round 18, the participant chose S1M mistakenly when the correct broken component was S2M, shown in Figure 5-9. By using the correct strategy without error, the participant is supposed to check the switch condition after entering the tray, to decide the active path, then to see if the components in the active path have lights on. From those components that are grey but in the active path, participant should use schematic knowledge to decide their order and find the broken component. PID 421 in round 18

of the test session made a mistake of choosing a grey component that is not in the active path (with the switch off). No matter if it is due to interface information ignoration (ignoring switches) or due to strategy error (PID 421 did not check the switches as they were supposed to), this behavior is observed and can be modeled.

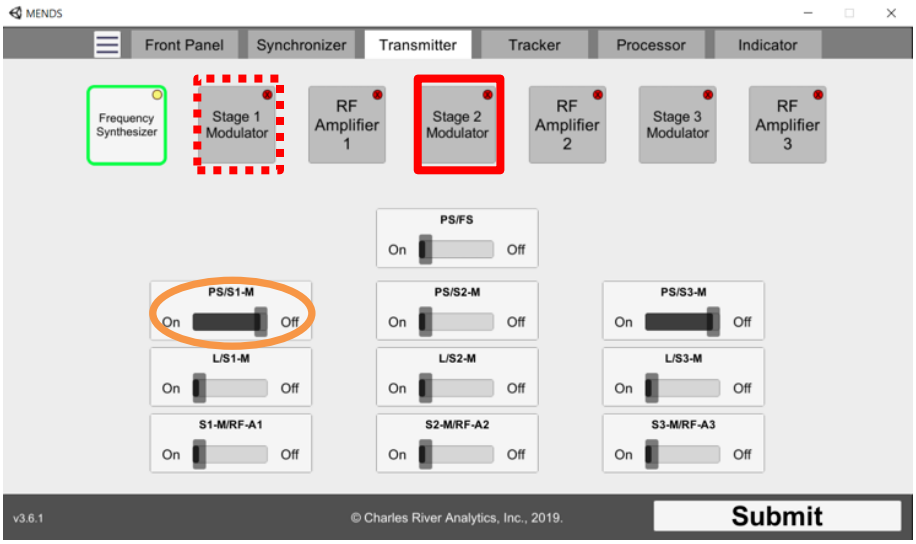



Figure **5-9**: *Error to Be Modeled*. For task S2M (red solid squared), PID ignored switch information and chose S1M (red dotted squared) that had switch off (orange circled).

To better model Participant 421's error, their mouse click record was analyzed in detail. Besides the error of clicking S1M and being notified by the system, PID 421 also made other unnecessary moves, such as going back and forth between two trays, Synchronizer and Transmitter. Those moves may be part of the strategy process but also part of the need to double check due to lack of working memory or lack of self-confidence, and so on. Some time may go to those behaviors, though they are not necessarily self-corrected errors. A deeper analysis of the time spent on these behaviors may be conducted later; currently, the focus remains on errors involving the replacement of incorrect components. Though there are differences among different strategies, there are also variations when participants implement the same strategy. The tab changes implied

that PID 421 may have some variations from the standard All-Trays Strategy, but in general went through all the trays. Therefore, the All-Trays Strategy model was updated to include error and error correction. This would be better for models to explain where time goes, but those behaviors are not yet carefully explored.

## Comparison of the Error Model, the Old Model, and Human Data

For illustration, the response times across the 20 tasks in the test session are compared to the predictions of the original All-Tray strategy model, the error model, and participant 421's actual performance (Figure 5-10). The Error model is more of a "glove and hand" shape to the data than the original model for observed time, and it also takes more time due to additional steps. The error model also shows a better power law of learning together with the participant data.

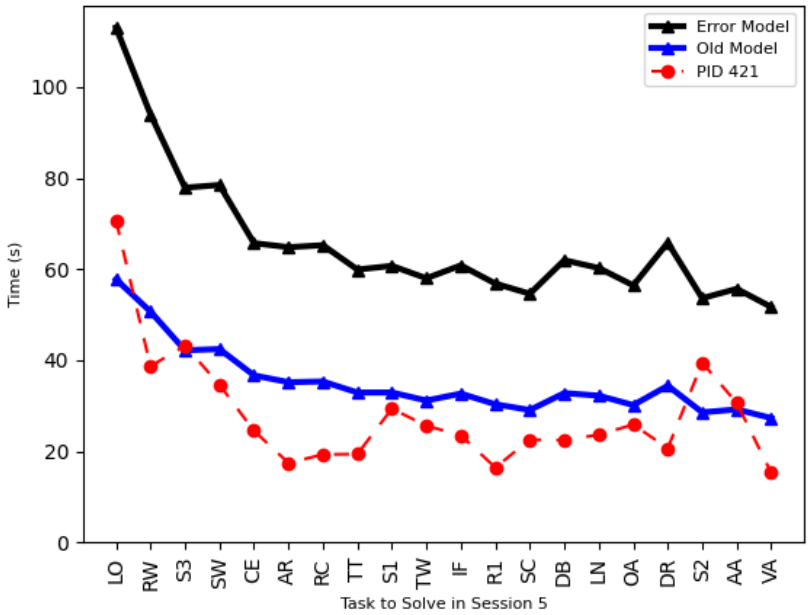



Figure **5-10**: *Comparison of Response Times for 20 Tasks in the Test Session of the Original All-Tray Strategy Model, the Error Model, and Participant 421's Performance*.

Table 5-1 shows the number of tasks, correlation, r-squared, mean squared error (MSE), and the root of mean squared error (RMSE) for the original All-Tray model and the updated error model. Their $R^2$ are statistically significant ($p < .05$). The error model has a higher positive correlation and $R^2$ than the original model, showing that my error model indeed catches more of participants' behaviours. At the same time, both the MSE and RMSE of the error model are much larger than the original model, which is expected from the gap of black and red lines in Figure 5-10. The gap arises from the addition of extra knowledge to simulate errors, which predicted longer response times. One potential solution to address this gap could be the use of an automatic error model (AEM) (Bagherzadehkhorasani & Tehranchi, 2023).

Table **5-1**: *Comparison of the Error Model and Original Model*. Both have p-values less than .05.

| Model | N | $R^2$ | MSE | RMSE |
|---|---|---|---|---|
| Error | 20 | .65 | 1490.27 | 38.60 |
| Original | 20 | .60 | 116.06 | 10.77 |

## Discussion and Conclusion

This section discusses the role of error models in enhancing understanding of participant learning and performance. It advocates for a task-specific approach to model building, using error analysis to create tailored models rather than relying on fixed categories. The error model improves correlations with human data, indicating the error model's potential, such as in high-stakes environments like aviation training. Furthermore, the TAKLML approach facilitates practical error modeling by simplifying complex tasks, and future research could compare its effectiveness with other methodologies to identify best practices. Details are described below.

**Discussion of the Error Model**

The patterns and trends of the errors tell us more about participants' learning and performance improvement. An unavoidable journey to success is to learn from errors, from many perspectives and not just in this fault-finding task.

A pattern of error categorization and modeling is presented here. There are other ways of categorization from the literature. This cognitive modeling study used task analysis to allocate time to mental process and behaviors to calculate performance time for prediction. To build error models, the error pattern is straightforward after error analysis and specific for this task. A practical method for building error models is proposed here: analyze the specific task at hand and develop a tailored error categorization for that task. This process eases the practical model building process. The discussion of the gap between theories and practical uses is not uncommon.

While emphasizing previous categorizations and their application to various tasks for the purpose of theoretical generalization, an alternative level of generalization is suggested: focusing on the method or process level. For the purpose of error model building, the process includes analyzing the errors specific to your task, figuring out the error patterns emerged through the analysis, and building your error model. This is useful when you use the modeling approach of hierarchical task analysis, the Keystroke-Level Model (KLM), and the power law.

A better understanding of errors can indeed help us understand better of not only with the tasks themselves, but also participants' performances of those tasks. Adding error generation and correction into a strategy model indeed increased correlation with human data. A relatively novel model of error generation is presented, which improves the correlation of the model's predictions, although it does not reduce the root mean square error (RMSE). It is believed that the root mean square error (RMSE) can be addressed in future work by adjusting parameters. More importantly, this model suggests where errors come from and predicts errors better than the error free model—

which did not predict errors at all. The error model shows its potential in performance prediction and can be further modified to match the red dotted human data line in Figure 5-10. The error model and its error analysis process show the potential of considering performance errors in aviation training and other high-stakes environments.

The errors categorized and modeled are only those leading to the final failure of tasks. If participants corrected themselves during the task and found the broken component at the end, the mistakes or "errors" were not considered nor modeled in this study. An expanded exploration of errors corrected by participants during the fault-finding process can reveal the existence of more errors and additional types of errors, such as unnecessary replacements and examinations of subsystems beyond what is required. Further analysis of where time goes can be done with those expanded analysis of errors in the process. Understanding and showing participants' learning and improvements through the analysis is also another exciting further analysis.

**Discussion for Modeling with TAKLML**

This study showcases that modeling in Python with task analysis can not only work but also accomplish difficult tasks, such as modeling errors. The TAKLML approach facilitates practical and systematic error modeling by breaking down complex tasks into manageable components. Researchers interested in error modeling are not required to apply a fixed error category within their model. Instead, they can analyze the errors observed and develop a custom error model tailored to the specific data and context.

This study represents an initial exploration and does not adequately demonstrate the comparative advantages of the TAKLML approach relative to other modeling methodologies. Future research could investigate the simplicity, speed, and efficiency of the TAKLML approach compared to alternative modeling methods. Structured experiments using newly collected data

could elucidate the strengths and weaknesses of each approach, thereby informing best practices in modeling strategies.

## Summary

This chapter presented a cognitive modeling study to show how modeling with the TAKLML approach can accomplish tasks that are previously considered very difficult. This chapter introduced error modeling as a difficult modeling action in the field, with task description and data of human performance errors. The chapter showcased how to model errors by analyzing human performance errors and gaining insights from the error analysis for each task and for each participant. This chapter then presented the modeling detail and the comparison results of the error model, the old model, and the human performance data.

# Chapter 6

# Discussion and Future Work

## Introduction

In the previous chapter, an interview study was presented to identify the pain points experienced by participants, along with two cognitive modeling studies that demonstrate the performance of a potential solution: the TAKLML approach (Task Analysis with Keystroke-Level Modeling and Learning). This approach employs task analysis, keystroke-level modeling, and learning using the power law from ACT-R, implemented in Python. This chapter discusses the impact of the TAKLML approach and talks about the new pain point identified from the performance demonstrations. The following discussion includes an examination of the limitations encountered in the three studies, insights gained from the findings, and considerations for future work.

## The TAKLML Approach

The TAKLML (Task Analysis with Keystroke-Level Modeling and Learning) approach offers a practical solution to coding-related challenges. Chapters 4 and 5 demonstrate its effectiveness, showcasing how this approach can make cognitive modeling more accessible for beginners, model individual differences to reveal diversity and simulate errors in high-stakes contexts. Although modest in scope, TAKLML addresses specific challenges, illustrating how researchers can make a meaningful impact within their manageable limits. By lowering the learning curve and entry barriers for cognitive modeling, TAKLML fosters greater inclusivity and usability in this field.

It extracted the equations from ACT-R to avoid the debugging issues of ACT-R modeling. It serves as a supplementary tool rather than a replacement for ACT-R models, catering to users with varying levels of technical expertise and their familiarity with ACT-R. By providing alternative methods, especially for new learners, this strategy enhances accessibility and usability within the modeling community.

## Extra Pain Points

Chapter 3 lists various pain points and challenges based on interview data, while Chapter 5 introduces additional concerns. Chapter 3 specifically talks about the usability issues associated with the ACT-R application interface. Chapter 5 brings up another interface, the simulated task interface. These two interfaces serve quite different purposes. The ACT-R application interface is used to load model files and run ACT-R models. The simulated task interface is used to help build the model and compare participant performance. It is developed by the researchers to mimic physical tasks and capture user interactions through keyboard presses and mouse clicks.

Chapter 5 demonstrates that the positioning of elements within the simulated task interface could affect participant performance. One error, named "Interface Information Ignorance," refers to a lack of awareness regarding the layout of components on the simulated interface. While comparing participant performance on the simulated task interface can yield valuable insights, it may introduce biases when contrasted with the real physical circuits. Additionally, the complexities associated with using the simulated task interface present further challenges, thus contributing to the overall pain points experienced by modelers.

## Discussion of Dilemmas

This section discusses various topics that emerged from the findings. Participant feedback revealed additional reasons why this dissertation is necessary for fostering communication within the community. Furthermore, the complexities of implementing changes were acknowledged, and this discussion will explore the challenges associated with such efforts. Additionally, this section explores how the new information may address and alleviate some of the previously expressed concerns.

### Why Care About Pain Points

This subsection addresses the significance of understanding pain points and the necessity of this dissertation. Key subtopics include: (a) unlocking the potential of ACT-R versus selectively engaging talented individuals, (b) facilitating an understanding of interconnected issues, and (c) fostering communication within the community by providing relevant topics, building common understanding, and supporting potential future collaborations. These topics are illustrated in Figure 6-1.

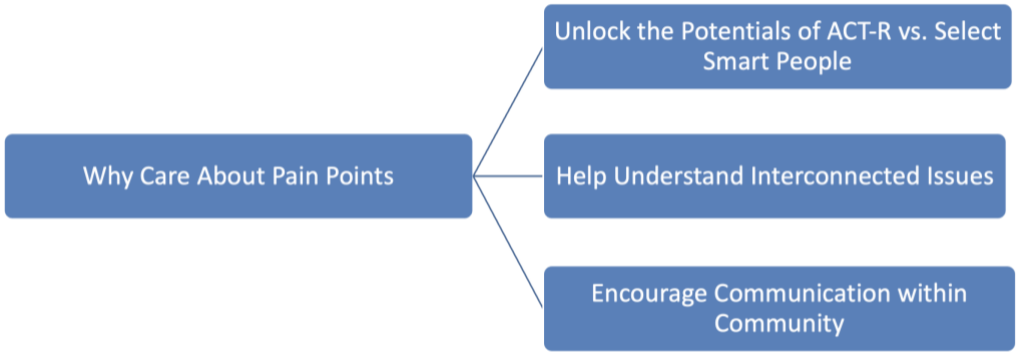



Figure **6-1**: *Why Care About Pain Points*.

As the project progressed, it became evident that not everyone prioritizes addressing pain points or reducing the learning curve. A question for researchers in the field, particularly senior researchers, is whether they aim to unlock the potential of ACT-R—much like how the potential of computers has been realized through diverse users and applications—or whether they prefer to focus on a select group of individuals. The former perspective is personally more appealing, which is why the interview study was conducted to identify pain points and explore opportunities for lowering the learning curve and increasing the accessibility of ACT-R research. Making the learning materials more accessible could emancipate human power. Both the professors and students can save time for in-depth discussions, instead of getting stuck at tool setup and debugging.

This dissertation also helps people have a relatively comprehensive and systematic understanding of the issues that ACT-R modelers and community face. The pain points are interconnected in some ways. Addressing a single problem may not yield a significant impact; therefore, a systematic presentation of these issues is provided here to facilitate a better understanding and to inspire more effective solutions. This collection of pain points also encourages communication with the community, by providing topics to discuss. With a comprehensive documentation of the pain points, even though people may have different opinions about them, those topics help build a common sense of the community which supports potential future collaborations.

### Not Easy to Change

This subsection addresses the challenges of effecting change, even when issues are clearly identified. The discussion is organized into the following subtopics: (a) the contrast between a unified theory in the education system and the trend toward specialization, (b) the differences

between apprenticeship models and online learning, (c) the disparity between identified needs and the assistance available, and (d) the obstacles without actionable suggestions. This set of issues is illustrated in Figure 6-2.

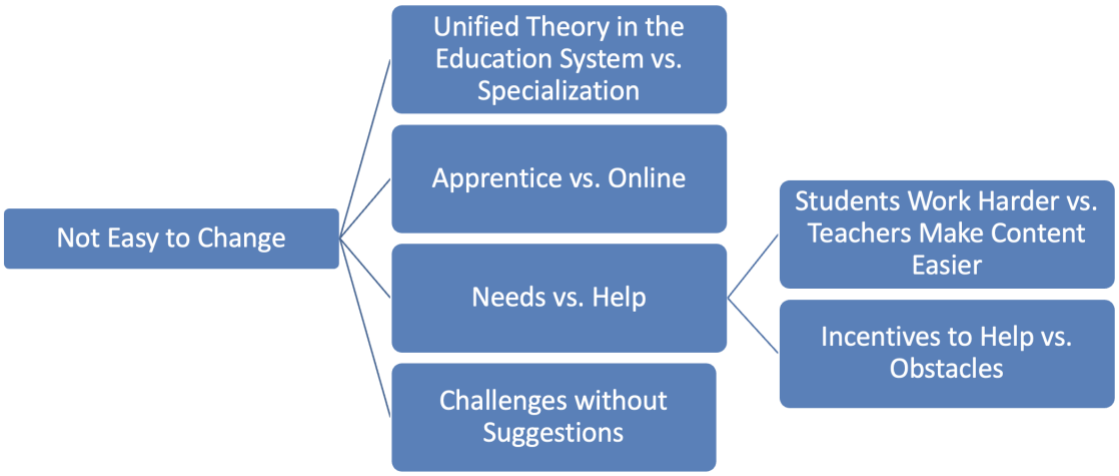



Figure **6-2**: *Not Easy to Change*.

The findings show that an explanation of the unified theory is not in the general education materials, unless the instructors happen to be ACT-R researchers and put ACT-R in their courses. Students tend to be asked to specialize in their majors. It is hard to train ACT-R researchers within the current education system, except by having apprentice-like training, which is slower and does not scale.

Moreover, participants in the study would like more online accessible resources. However, some skills such as what is a good model can be hard to write down in words, as P11 said. Online resources can be helpful and can compensate limitations of apprentice training but may not replace the apprentice-like training.

This dissertation has collected many pain points and suggestions, but there may not be real actions to help. Discussions of whether students should work harder versus whether teachers

should make content easier to digest may arise. It is not just about simplifying content; it is about reducing unnecessary complexity. The goal is to streamline the material, making it more accessible without diluting its richness or value. This side focuses on removing barriers that may hinder understanding, ensuring that the content remains engaging and informative. However, even though some professors would like to help, they encounter obstacles. For example, P7 pointed out that at least for young professors, they are too busy to build their research portfolio to secure their tenure and may not have extra time to do extra volunteer. Other participants expressed a strong willingness to offer help.

Many pain points are left out for which no participant has provided any suggestion. Those issues are just hard to tackle. They include the difficulty of information search, the lack of guidelines and standards for a good model, code sustainability and compatibility, and the education path. This dissertation opens up doors for readers to think about the issues collected and to communicate within or outside the ACT-R community.

**No Need to Panic**

This subsection explores areas where concerns may be alleviated. The subtopics include: (a) the potential disappearance of the community contrasted with the presence of interested students, and (b) the decline of Lisp programming or its potential revival. These issues are illustrated in Figure 6-3.

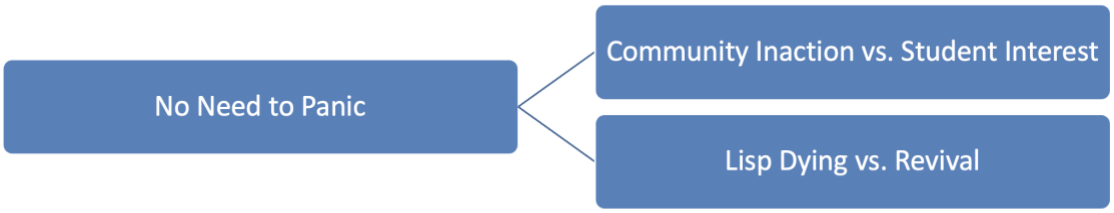



Figure **6-3**: *No Need to Panic.*

While some participants shared their pain points and raised their concerns, some other participants provided information that helped ease these concerns. For example, some people worry about the sustainability of the ACT-R community as they observe fewer students joining the community. P4 responded to this concern that there would always be a type of student who is interested in ACT-R or the unified theory. They might be born with the mindset of integration. They may not be the majority, but they will always exist. During the interviews with students, several expressed a strong interest in both psychology and computer science, indicating a desire to pursue academic careers focused on studying the mind.

Another concern is that Lisp might be dying with fewer users. There would not be enough incentives for Lisp developers to maintain the Lisp coding tools, and the ACT-R building that needs Lisp would be gone together. However, an encounter with a Lisp developer provided insights. They are a strong advocate for the Common Lisp language and believe in the revival of the language. They claimed ACT-R as the “most important Common Lisp project I had never heard of”. There could be more communication with those Lisp developers for potential collaboration.

## Summary of Limitations

This dissertation also has some limitations. For the interview study, additional ACT-R modelers from the Netherlands could have been invited to participate. The University of Groningen is also a place where a group of researchers gathered and are interested in cognitive modeling. Their learning and using experience of ACT-R might be slightly different. The current dissertation only represents the participants recruited and those who have no problem learning, building, and/or teaching ACT-R have less desire to say anything and may have just ignored my invitation message. This dissertation also has the limitation of the interview method, that the stories from the participants could not be verified. Participants' opinions may not align with the past facts and can also change as time goes by. More pain points could be found if broaden the scope to include connecting models to the simulation environment or using models in specific tasks, such as user model for testing interfaces (Tehranchi, Bagherzadeh, & Ritter, 2023)

For performance demonstrations, two cognitive modeling studies were presented to illustrate that the new approach works. With additional time and resources, experiments could be conducted to demonstrate that this new approach is faster than programming in Lisp and using the entire ACT-R architecture.

## Future Work

This dissertation has presented the pain points of the ACT-R modelers and the concerns of the whole research ecosystem. It has also pointed out the potential solutions and suggestions. The audience could have a better understanding of the difficulties faced by the research ecosystem. Providing specific actionable steps is challenging given the author's own limitations as a PhD student. The contribution of this study lies in offering more details than before, though not an

exhaustive account, as this is the first attempt to explore the topic amidst many obstacles, uncertainties, and hesitations. This dissertation hopefully can spark discussions on actionable steps, as also mentioned within the work.

Actionable steps are difficult to define, but the following are some potential directions. Further and future studies could pick any direction from all those listed pain points and potential solutions to make their contribution. More specific experiments can be set up to provide quantitative evidence to see if the approach is easier and faster than other approaches. Researchers can also combine ACT-R with other AI techniques for innovative applications. They can change the learning constant for different subtasks (Kim &Ritter, 2016). Researchers can also work on the difficulties in connecting models to the task environment. Researchers can also collect insights from current Lisp developers who may not heard of ACT-R and design for debugging functions together.

This dissertation identifies several challenges faced by modelers. Future design-focused work could be beneficial. To better address user needs and ensure practical application, future research might use focus groups and participatory design approaches. The goal would be to foster communication within the community, not simply to create new tools. In this way, researchers could ensure that new solutions are both relevant and impactful. Students and professors could help solve different levels of challenges, with students working on lower-level improvement and professors working on community-level improvement. Collaboration between modelers and HCI researchers and designers is highly encouraged.

In addition to research, fostering community building and knowledge sharing online will be beneficial in the long term. Given the challenges identified, along with the information in Table 2-1 and the recommendations from this dissertation, further analysis of priorities and feasibility of solutions as well as deeper connections among challenges will benefit from

quantitative evaluation. This approach can help ensure that the community's needs are effectively addressed and supported.

## Summary

This chapter summarized the discussion and future work of the dissertation, including a discussion of the TAKLML approach, extra pain points identified, discussions of dilemmas, a summary of limitations from three studies, and future work. The chapter discussed dilemmas from three major themes: (a) why do we care about pain points, (b) how it is not easy to make changes, and (c) what we could be less concerned about. Under each of the major themes, the chapter discussed in further detail. Subtopics for “why we care about pain points and this dissertation” include unlocking potential of ACT-R vs. selecting smart people, helping understand interconnected issues, encouraging communication within the community by providing topics, as well as helping build common sense and supporting potential future collaborations. Subtopics for “how it is not easy to make changes” include providing unified theory in the education system for ACT-R or in Psychology in general vs. specialization, apprentice vs. online, needs vs help, and difficulties without suggestions. Subtopics for “what we can be less concerned about” include community disappearing vs. interested students and lisp dying vs. revival. The chapter ended by talking about how future work can emerge from this dissertation.

## References

Adams W. (2015). Conducting semi-structured interviews. In Newcomer K. E., Hatry H. P., Wholey J. S. (Eds.), *Handbook of practical program evaluation* (4th ed., pp. 492–505). John Wiley & Sons

Agre, P. E. (2005). How to be a leader in your field: A guide for students in professional schools https://pages.gseis.ucla.edu/faculty/agre/leader.html.

Amant, R. S., Freed, A. R., & Ritter, F. E. (2005). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research*, *6*(1), 71–88. https://doi.org/10.1016/j.cogsys.2004.09.008

Anderson, J. R. (2007). The Adaptive Control of Thought. In *How Can the Human Mind Occur in the Physical Universe?* (pp. 135–186). https://doi.org/10.1093/acprof:oso/9780195324259.003.0004

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. Psychological Science, 2(6), 396-408. https://doi.org/10.1111/j.1467-9280.1991.tb00174.x

Andren, L., Papp, D., Penno, E., & Vallas, A. (2008). A qualititative approach to HCI research. *Eesti Teadusliku Seltsi Rootsis: Aastaraamat. Annales Societatis Litterarum Estonicae in Svecia*, *11*, 216;ill. http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521870122&ss=toc

Anzai, Y., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review*, *86*(2), 124–140. https://doi.org/10.1037/0033-295X.86.2.124

Ashby, F. G., & Maddox, W. T. (2005). Human category learning. *Annual Review of Psychology*, *56*(February), 149–178. https://doi.org/10.1146/annurev.psych.56.091103.070217

Bagherzadehkhorasani, A., & Tehranchi, F. (2023, September). Automatic Error Model (AEM) for User Interface Design: A new approach to Include Errors and Error Corrections in a Cognitive User Model. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 67, No. 1, pp. 644-649). Sage CA: Los Angeles, CA: SAGE Publications.

Baxter, G. D., Churchill, E. F., & Ritter, F. E. (2014). Addressing the fundamental error of design using the ABCS. AIS SIGHCI Newsletter, 13(1), 9-10.

Best, B. J., & Lovett, M. (2006). Inducing a cognitive model from examples provided by an optimal algorithm. *Proceedings of the Seventh International Conference on Cognitive Modeling* (pp. 56-61).

Best, B. J., Lebiere, C., & Scarpinatto, K. C. (2002). Modeling synthetic opponents in MOUT training simulations using the ACT-R cognitive architecture. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavior Representation* (Vol. 7, p. 33).

Brasoveanu, A., Dotlačil, J., Brasoveanu, A., & Dotlačil, J. (2020). The ACT-R cognitive architecture and its pyactr implementation. *Computational Cognitive Modeling and Linguistic Theory*, 7-37.

Booher, H. R., & Minninger, J. (2005). Human systems integration in army systems acquisition. In *Handbook of Human Systems Integration* (pp. 663–698). John Wiley & Sons, Inc. https://doi.org/10.1002/0471721174.ch18

Büttner, P. (2010). "Hello Java!" Linking ACT-R 6 with a Java simulation. In *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 289-290). Philadelphia: Drexel University.

Cairns, P., & Cox, A. L. (Eds.). (2008). *Research methods for human-computer interaction*. Cambridge, UK: Cambridge University Press.

Card, S. K., Newell, A., & Moran, T. P. (1983). *The psychology of human-computer interaction*. L. Erlbaum Associates Inc.

Choi, D., & Langley, P. (2018). Evolution of the icarus cognitive architecture. *Cognitive Systems Research,48,* 25–38.

Cockburn, A., Gutwin, C., Scarr, J., & Malacria, S. (2015). Supporting novice to expert transitions in user interfaces. *ACM Computing Surveys (CSUR) Surveys Homepage archive, 47*(2), Article No. 31.

Colusso, L., Munson, S. A., Jones, R., & Hsieh, G. (2019). A translational science model for HCI. *Conference on Human Factors in Computing Systems - Proceedings*, 1–13. https://doi.org/10.1145/3290605.3300231

Daily, L. Z., Lovett, M. C., & Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science*, *25*(3), 315–353. https://doi.org/10.1016/S0364-0213(01)00039-8

Delaney, P. F., Reder, L. M., Staszewski, J. J., & Ritter, F. E. (1998). The strategy-specific nature of improvement: The power law applies by strategy within task. *Psychological Science*, *9*(1), 1-7.

Elio, R., & Scharf, P. B. (1990). Modeling novice-to-expert shifts in problem-solving strategy and knowledge organization. *Cognitive Science*, *14*(4), 579–639. https://doi.org/10.1016/0364-0213(90)90010-T

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, *5*(1), 1-4.

Friedrich, M. B., & Ritter, F. E. (2020). Understanding strategy differences in a fault-finding task. *Cognitive Systems Research*, *59*, 133–150. https://doi.org/10.1016/j.cogsys.2019.09.017

Glaser, B., & Strauss, A. (2017). *Discovery of grounded theory: Strategies for qualitative*

*research*. Routledge.

Hsieh, H. F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research*, *15*(9), 1277-1288.

Jones, G., & Ritter, F. E. (2000). Over-estimating cognition time: The benefits of using a task simulation. In Simulating Human Agents, American Association for Artificial Intelligence Fall 2000 Symposium Series, 67-74. Menlo Park, CA: AAAI Press.

Kase, S. E., Ritter, F. E., Bennett, J. M., Klein, L. C., & Schoelles, M. (2017). Fitting a model to behavior reveals what changes cognitively when under stress and with caffeine. *Biologically Inspired Cognitive Architectures, 22*, 1–9. https://doi.org/10.1016/j.bica.2017.09.008

Kieras, D. E., & Meyer, D. E. (2020). The role of cognitive task analysis in the application of predictive models of human performance. In *Cognitive Task Analysis* (pp. 251–274). https://doi.org/10.4324/9781410605795-25

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *Transactions on Computer-Human Interaction, 4*(3), 230-275.

Kim, J. W., & Ritter, F. E. (2016). Microgenetic analysis of learning a task: Its implications to cognitive modeling. In *Proceedings of the 14th International Conference on Cognitive Modeling (ICCM 2016)*, 21-26. University Park, PA: ACS Lab, Penn State.

Koripi, M. (2020). A review on architectures and needs in advanced wireless-communication technologies. *A Journal of Composition Theory, 13*, 208–214.

Kotseruba, I., & Tsotsos, J. K. (2020). 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, *53*(1), 17-94.

Kukreja, U., Stevenson, W. E., & Ritter, F. E. (2006). RUI: Recording user input from interfaces under Windows and Mac OS X. *Behavior Research Methods*, *38*(4), 656–659. https://doi.org/10.3758/BF03193898

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Landauer, T. K. (1995). Let's get real: A position paper on the role of cognitive psychology in the design of humanly useful and usable systems. In *Readings in Human–Computer Interaction* (pp. 659–665). https://doi.org/10.1016/b978-0-08-051574-8.50067-4

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Models of competence in solving physics problems. *Cognitive Science*, *4*(4), 317–345. https://doi.org/10.1016/S0364-0213(80)80008-5

Lazar, J., Feng, J. H., & Hochheiser, H. (2017). *Research methods in human-computer interaction* (2nd ed.). Cambridge, MA: Morgan Kaufman.

Mavor, A. S., & Pew, R. W. (Eds.). (1998). *Modeling human and organizational behavior: Application to military simulations*. National Academies Press.

Mavor, A. S., & Pew, R. W. (Eds.). (2007). *Human-system integration in the system development process: A new look*. National Academies Press.

Morgan, J. H., Cheng, C. Y., Pike, C., & Ritter, F. E. (2013). A design, tests and considerations for improving keystroke and mouse loggers. *Interacting with Computers, 25*(3), 242-258.

Naderifar, M., Goli, H., & Ghaljaie, F. (2017). Snowball sampling: A purposeful method of sampling in qualitative research. *Strides in Development of Medical Education, 14*(3), 1-6.

National Research Council. (1998). *Modeling human and organizational behavior: application to military simulations*. National Academies Press.

Newell, A. (1994). *Unified theories of cognition*. Harvard University Press.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review 88*, 1-15.

Oyewole, S. A., Farde, A. M., Haight, J. M., & Okareh, O. T. (2011). Evaluation of complex and dynamic safety tasks in human learning using the ACT-R and SOAR skill acquisition theories. *Computers in Human Behavior, 27*(5), 1984–1995. https://doi.org/10.1016/j.chb.2011.05.005

Paik, J., Kim, J. W., Ritter, F. E., & Reitter, D. (2015). Predicting user performance and learning in human-computer interaction with the Herbal compiler. *ACM Transactions on Computer-Human Interaction (TOCHI), 22*(5), 1-26.

Pew, R. W., & Mavor, A. S. (Eds.). (1998). Modeling human and organizational behavior: Application to military simulations. Washington, DC: National Academy Press. books.nap.edu/catalog/6173, checked Feb 2020.

Pew, R. W., & Mavor, A. S. (Eds.). (2007). Human-system integration in the system development process: A new look. Washington, DC: National Academy Press. books.nap.edu/catalog/11893, checked May 2019.

Goodman, L. A. (1961). Snowball sampling. *The Annals of Mathematical Statistics, 32*(1), 148-170.

Reason, J. (1990). *Human error*. Cambridge, UK: Cambridge University Press.

Ricupero, S. (2024). *Modeling working memory: Varying noise based on neural differences.* Unpublished PhD thesis, Huck Institute, The Pennsylvania State University, University Park, PA.

Ritter, F. E. (accepted July 2022, accessed 2024). Design patterns for cognitive simulations and HCI. Oxford, UK: Oxford University Press.

Ritter, F. E., Baxter, G. D., Jones, G., & Young, R. M. (2000). Supporting cognitive models as users. ACM Transactions on Computer-Human Interaction, 7(2), 141-173.

Ritter, F. E., Bittner, J. L., Kase, S. E., Evertsz, R., Pedrotti, M., & Busetta, P. (2012). CoJACK: A high-level cognitive architecture with demonstrations of moderators, variability, and implications for situation awareness. *Biologically Inspired Cognitive Architectures, 1*(1), 2-13.

Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. *Human-Computer Interaction*, *9*(3&4), 345–383. https://doi.org/10.1207/s15327051hci0903&4_4

Ritter, F. E., Tehranchi, F., Brener, M., & Wang, S. (2020). Testing a complex training task. *Proceedings of ICCM 2019 - 17th International Conference on Cognitive Modeling, January 2019*, 184–185. https://www.researchgate.net/publication/339051161

Ritter, F. E., Morgan, G. P., Stevenson, W. E., & Cohen, M. A. (2005). A tutorial on Herbal: A highlevel language and development environment based on Protégé for developing cognitive models in Soar. *Proceedings of the 14th Conference on Behavior Representation in Modeling and Simulation*. University of Central Florida Orlando, FL.

Ritter, F. E., Tehranchi, F., & Oury, J. D. (2019). ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, *10*(3), 1–19. https://doi.org/10.1002/wcs.1488

Ritter, F. E., Workman, D., & Wang, S. (2022). Predicting learning in a troubleshooting task using a cognitive architecture-based task analysis. *International Conference on Cognitive Modeling*. 222-224

Ritter, F. E., Ricupero, S., Yeh, M. K., Workman, D., Oury, J. D., Stager, S. J., & McDermott, A. F. (2022). Testing a learning and retention theory using a troubleshooting task. (Tech. Report No. ACS 2022-1). Applied Cognitive Science Lab, College of Information Sciences and Technology, Penn State.

Ritter, F. E., & Ricupero, S. (2023). *Running Behavioral Studies With Human Participants Online*. SAGE Publications Ltd.

Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R. M., Gobet, F., & Baxter, G. D. (2003). *Techniques for modeling human performance in synthetic environments: A supplementary review*. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center (HSIAC).

Siegler, R. S. (1988a). Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General*, *117*(3), 258–275. https://doi.org/10.1037/0096-3445.117.3.258

Siegler, R. S. (1988b). Individual differences in strategy choices: Good students, not-so-good students, and perfectionists. *Child Development*, 833-851.

Tehranchi, F., & Ritter, F. E. (2016). Connecting cognitive models to interact with human-computer interfaces. *Proceedings of ICCM-2016-14th International Conference on Cognitive Modeling*. 266–267.

Tehranchi, F., & Ritter, F. E. (2017). An eyes and hands model for cognitive architectures to interact with user interfaces. *28th Modern Artificial Intelligence and Cognitive Science Conference, MAICS 2017*, 15–20.

Tehranchi, F., & Ritter, F. E. (2018a). Using Java to provide cognitive models with a more universal way to interact with graphical user interfaces. *Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction and Behavior Representation in Modeling and Simulation, BRiMS*.

Tehranchi, F., & Ritter, F. E. (2018b). Modeling visual search in interactive graphic interfaces: Adding visual pattern matching algorithms to ACT-R. *Proceedings of ICCM 2018 - 16th International Conference on Cognitive Modeling*, 162–167.

Tehranchi, F., Bagherzadehkhorasani, A., & Ritter, F. E. (2023). A user model to directly compare two unmodified interfaces: A study of including errors and error corrections in a cognitive user model. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 37*(e27), 1-11.

Tsoi, A. C., & Back, A. D. (1994). Locally recurrent globally feedforward networks: a critical review of architectures. *IEEE Transactions on Neural Networks, 5*(2), 229-239.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science, 15*(1), 1–47. https://doi.org/10.1207/s15516709cog1501_1

Wang, S. N., & Ritter, F. E. (2023a). Modeling a strategy with learning in a complex task. *International Conference on Cognitive Modeling*. 275-277.

Wang, S. N., & Ritter, F. E. (2023b). Exploring errors towards a more realistic strategy model. *International Conference on Cognitive Modeling*. 277-279.

## Appendix A

# Interview Guide

The following is the semi-structured interview guide used in the study detailed in Chapter 3. The questions listed below were selected to ask participants, based on their current jobs. Following this predetermined set of questions, the interviewer asked follow-up or clarification questions based on their responses. Also, all participants were given the opportunity to share any additional information, insights, concerns, or questions that they may have developed over the course of the interview.

1. Learning.
    a. How did you learn ACT-R (materials used, mentor, advisor, …)?
    b. How long did it take you to learn ACT-R?
    c. What are the difficult parts in journey of learning ACT-R?
    d. Any suggestion for potential improvement?
2. Using.
    a. How did you use ACT-R (tools used, etc.)?
    b. How long did it take you to use ACT-R?
    c. What are the difficult parts in journey of using ACT-R?
    d. Any suggestion of improvement?
3. Tasks related.
    a. What tasks have you used in your previous studies?
    b. How long did you take to code for the task, the model, etc.?
4. Insights for ACT-R community.
    a. Why do you think other people do not use ACT-R/cognitive modeling?
    b. Why do you think researchers leaving the field?

c. Do you know who else are using ACT-R and are a potential participant as well?

The following are the script used to communicate with participants.

"Hi __, thank you for your time participating in the study. Today I'm going to interviewing you about your learning and using experiences of ACT-R. I'm about to record this meeting. This interview includes two parts. The first part is questions. The second part requires you to share your screen and show me how you usually build an ACT-R model.

Let's get it started.

(Here the interviewer asked questions, as well as follow-up questions)

Okay. Now we finished the first part. Let's move to the second part. Please share your screen. Thank you.

Assuming you will need to build an ACT-R model for a task. Please show me the process of how you will act, including what tools, interfaces you might use during the whole process. The whole process includes analyzing the declarative knowledge and procedural knowledge, designing chunks and rules, acting building the model, testing and debugging the model.

Thank you for your sharing. Now we ended the second part of the interview.
Any questions that you have for me or for the project?
Any suggestions that you have for me or for the project?

Noted. Thank you.

I will stop the recording. Thank you again for your time. my interview session has ended. You are free to leave the zoom meeting. Hope you have a great day!”

**Appendix B**

## A Sample of Coded Transcripts

| Information units/Abstracts | Theme | Sub-themes | Codes |
|---|---|---|---|
| But the learning. Hmmm. Well, I think one part of the learning process that is difficult is that, especially because I did it on my own, I had a very, very weak sense of what is good modeling and bad modeling practices. | Challenges | Material related, Lack of guidelines and standards | Learning process |
| It's actually pretty hard to download if you haven't done it before. Once you do it, it's fine, but if you're doing it alone and no one is with you, it can actually be kind of difficult to download LISP. | Challenges | Coding related, Setup | download |
| Some interactive tutorials would be very beneficial. | Suggestions | Material related, Tutorial | Interactive tutorial |
| You know, because it's not something that's as common now. I mean, if you go through a computer science, like, you know, I learned some of that in my classes as an undergraduate student, but for someone who doesn't have that kind of exposure, I mean, they're just going to be very confused. And that's not their fault. It's just it's not part of the curriculum, really. You need the programing, and you need to understand the concepts related to cognitive psychology. So, I think it's a problem. | Challenges | Community related, Education path | curriculum |

| Information units/Abstracts | Higher-level codes | Lower-level codes | Topics and sub-topics |
|---|---|---|---|
| The difficult part is that you spend so much time on debugging. It's crazy. There are these stupid things that, the production rules they pick up on, on certain chunks that they're not supposed to pick up on, and they fire, and you don't want them to fire or you want them to fire, but then on a different chunk, then the chunk it is firing on, that is actually the most, annoying part because you don't get, error messages, then you just get, yeah, a model run and it seems to be all fine. But it is not fine. So, what you should do then is to go through all the whole trace, basically. And that takes just a lot of time | Challenges | Coding related, Debugging | Debugging |
| I've seen the manual. And it's it hasn't been very useful to me. Like maybe in some instances if you want to look up one parameter and what the default value is or some things, some simple things. But it's like hundreds of pages. It hasn't been proven useful to me so much | Challenges | Material related, Information search | Look up |
| I just kind of wonder like what that looks like in the future. Does, you know, does ACT-R kind of stop with when that core group at CMU retires or. I don't think ACT-R ends there. But how it continues is key. | Challenges | Community related, Centralization, sustainability | ACT-R ends |

**Appendix C**

## Partial Python Code

The following presents the code for makeFault(), clearFault(), and propagate() functions in task simulation. The coding languages is Python. Comments are after "#".

```
#### 1 insert a fault
def makeFault(x = None):
    if x == None :
        #automake fault
        #make x any possible component.not power nor switches.
        randomnum = random.randint(1, 35)
        x = benFranklin['Name'][randomnum]
    elif x == 'Power':
        print("Try again.Please choose to break a component that's not power.")
        return
    elif x in listofswitches:
        print("Try again. Please choose to break a component that's not a switch.")
        return
    elif x not in component_names:
        print(x, "Try again. Type correct component name.")

    #first. before breaking, make sure all function well with x.
    clearFault(x) #finished
    #break x. => status=0. light=0
    benFranklin.loc[benFranklin[benFranklin['Name'] == x].index[0], 'Status'] = 0
    benFranklin.loc[benFranklin[benFranklin['Name'] == x].index[0], 'Light'] = 0
    #results of breaking x.turn off light of x's followers
    xrownum = benFranklin[benFranklin['Name'] == x].index[0]
    propagate(xrownum)
```

```
def clearFault(x):#used in makeFault()
    #x is the componnet you want to break after clearFault()
    #first turn all components on, then turn off 2 pairs modulators that are supposed to be off.
    benFranklin.loc[benFranklin.Status != 1, 'Status'] = 1 # swtich on
    benFranklin.loc[benFranklin.Light != 1, 'Light'] = 1
    benFranklin.loc[0:35,'Switch'] = 0
    benFranklin.loc[36:50,'Switch'] = 1

    #set clean rule based on learning type
    if learningornot == 3: #no learning at all
        global cleanaftereach
        cleanaftereach = True
        #if nolearnig, clear history
    elif learningornot == 2: #learning within one faultfinding
        cleanaftereach = True
    elif learningornot ==1: #with learning. checked and trails across whole session
        cleanaftereach = False
        #if ploting 35, add global cleanaftereach=True
        #when call s1() only. set global ceanaftereach =True
        # if ploting session1 and 5, for y1. add global clearnaftereach=False
        #if ploting session1 and 5, for y2. cleanaftereach= True.
        #if ploting session1 and 5, for y3. cleanaftereach=True
    if cleanaftereach == True:
        benFranklin.loc[0:50, 'Checked'] = 0
    #else: clean in plot functions # if ploting session1or5, clean history after whole session1 or 5.
    #make sure clean codes"benFranklin.loc[0:50, 'Checked'] = 0" are at end of plot functions for
session1and5
    #how to clear checked after print one session. so far done in end of plot functions
    #make sure x is in the active path.
    if x in ['S1M', 'RFA1', 'S2M', 'RFA2', 'S3M', 'RFA3']:
        # if the broken part is among modulators
        pass
    else:#if x is not one of modulator pairs
```

```
        randomnum = random.randint(1, 3)
        x = randomnum
    #turn off pairs of modulaters based on switch conditions.
    if x in ['S1M', 'RFA1', 1]:
        benFranklin.loc[[38,41,49,39,42,50],'Switch'] = 2 # swtich=2, means off
        benFranklin.loc[[10,9,12,11], 'Light'] = 0
    if x in ['S2M', 'RFA2', 2]:
        benFranklin.loc[[37,40,48,39,42,50],'Switch'] = 2
        benFranklin.loc[[8,7,12,11], 'Light'] = 0
    if x in ['S3M', 'RFA3', 3]:
        benFranklin.loc[[37,40,48,38,41,49],'Switch'] = 2
        benFranklin.loc[[8,7,10,9], 'Light'] = 0
    #now all is working well. with only one pair modulator.
    #next. break component!and turn off followers!(back to makefault. progagate)
    #end here.

#### 2 porpogage a fault/propagae a swtich/ propogate a change/ run the
#x.status=0, light=0 already! now propagate!
    #x is 1 to 35, with any pair of 7 8 or 9 10 or 11 12
    #output numbers may be 0,1,2,3,4,or 6(Limiter to modulator) #power and switches are ok.
    #Limiter can be a bit tricky to propagate
    #breadth or depth first propagate
    #now depth first propagage

def propagate(xrownum):#used in makeFault()
    for i in benFranklin.loc[xrownum, 'Outputs']:
        if len(benFranklin.loc[xrownum, 'Outputs']) == 0:break
        #i should not be swiches.
        if i in listofswitches:continue
        #i.light=0 #if already off.next iteration
        #depth first propagate continue.
        elif benFranklin['Light'].values[benFranklin[benFranklin['Name'] == i].index][0] ==
0:continue
```

```
else:
    benFranklin.loc[benFranklin[benFranklin['Name'] == i].index[0], 'Light'] = 0
    irownum = benFranklin[benFranklin['Name'] == i].index[0]
    propagate(irownum)
```

# VITA

**Shan Wang**

## EDUCATION

Ph.D., Informatics, Penn State University — Aug 2020 – Dec 2024

M.S., IST, Penn State University — Aug 2018 – Jun 2020

B.A., Mathematics & Psychology, UC Davis — Sep 2014 – Jun 2018

## EXPERIENCE

**Research Assistant**, The Applied Cognitive Science (ACS) Lab — Jan 2019 – May 2024

**Teaching Assistant**, College of IST — Aug 2018 – May 2024

**Research Assistant**, The Human in Computing and Cognition (THICC) Lab — Jun 2023 – Dec 2023

**Lab member**, The Wellbeing & Health Innovation (WHI) Lab — Aug 2018 – Jun 2020

## PUBLICATIONS

Wang, S. N., & Ritter, F. E. (2023). Modeling a strategy with learning in a complex task. *International Conference on Cognitive Modeling*. 275-277.

Wang, S. N., & Ritter, F. E. (2023). Exploring errors towards a more realistic strategy model. *International Conference on Cognitive Modeling*. 277-279.

Ritter, F. E., Workman, D. & Wang, S. (2022). Predicting learning in the troubleshooting task using a cognitive, architecture-based task analysis. In *Proceedings of the 20th International Conference on Cognitive Modeling (ICCM 2022).* 222-224.

Kunchay, S., Wang, S., & Abdullah, S. (2019, November). Investigating users' perceptions of light behaviors in smart-speakers. In *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing* (pp. 262-266).

Ritter, F. E., Tehranchi, F., Brener, M., & Wang, S. (2019). Testing a complex training task. In *Proceedings of the 17th International Conference on Cognitive Modeling (ICCM 2019). pp.* 184-185.

## PROFESSIONAL SERVICE

**Student Affiliate**, Society Center for Socially Responsible Artificial Intelligence — Sep 2023 – Present

**Student Member**, Human Factors & Ergonomics — Aug 2023 – May 2024

**Talk Presenter**, The Soar Workshop — Jun 2023

**Poster Presenter**, Graduate Exhibition — Mar 2023

**Reviewer,** Computers in Human Behavior, CHI LBW (Conference on Human Factors in Computing Systems Late Breaking Work) 2022, C & C (Conference on Creativity and Cognition) 2022, DIS (Conference on Designing Interactive Systems) 2022