2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52759

Bee Learning Mentoring Model in a Computer Science Research Project

Dr. Rahman Tashakkori, Appalachian State University

Rahman Tashakkori received his PhD in Computer Science from Louisiana State University in 2001. He is currently serving as the Lowe's Distinguished Professor of Computer Science at Appalachian State University. He is the program director on Beemon project and has led several NSF projects.

Dr. Abdelbaset Safwat Hamza
Christopher Leigh Campell, Appalachian State University

Bee Learning Mentoring Model in a Computer Science Research Project

Rahman Tashakkori, Abdelbaset Hamza, Christopher Campell
Appalachian State University
{*tashakkorir, hamzaas, campellcl*}@appstate.edu

Abstract

Sustaining effective mentoring and knowledge transfer in computer science research labs is challenging when students cycle through every three to six semesters, creating recurring gaps in expertise and project continuity. This paper presents the Bee Learning Mentoring Model, a framework for structured mentoring in CS research labs inspired by the social learning mechanisms of honeybees (Apis mellifera). The model was developed and refined over eight years within the Beemon project at Appalachian State University (App State), where undergraduate and graduate students collaboratively design, build, and maintain a remote beehive monitoring system encompassing hardware assembly, software development, networking, and data analysis. Drawing on parallels between honeybee colony organization and educational mentoring theory, the model is organized around four principles: (1) Guided Participation, (2) Peer Learning through Demonstration, (3) Adaptive Role Transitions, and (4) Distributed Knowledge Networks. We derived these principles through longitudinal reflective practice, informed by both honeybee behavioral research and socio-cultural learning theory. Over the eight-year span of the project, 44 students (68% undergraduate, 32% graduate) have participated, with 100% degree completion. Students co-authored 18 peer-reviewed papers and delivered 29 conference presentations with their mentors. Approximately 57% entered the workforce and 43% pursued or are pursuing graduate studies. Unlike conventional mentoring frameworks, this model explicitly addresses knowledge continuity across student generations by embedding adaptive role transitions and distributed knowledge networks as structural features. The paper contributes a biologically grounded, empirically refined mentoring framework with implications for CS education, interdisciplinary research teams, and labs facing similar challenges of student turnover.

Keywords: Honeybees, social learning, mentoring models, collaborative learning, CS education

1. Introduction

Learning in social systems is often distributed across individuals rather than isolated within single learners. Honeybees provide one of the most studied biological examples of socially mediated learning, where communication, observation, and interaction shape individual and collective behavior [1]. In educational research, mentoring similarly emphasizes guided participation, peer learning, and role modeling. Although bees and students differ fundamentally, we have observed that the principles underlying learning and mentorship show notable parallels. This paper examines honeybee social learning mechanisms and discusses how these findings have informed a student mentoring model in our research group. Honeybee learning paradigms, such as proboscis extension reflex conditioning, have been used in STEM education to mentor students through hands-on scientific inquiry. These experiences promote critical thinking, hypothesis testing, and guided learning, mirroring mentor-mentee relationships in research training [1].

Our research project deals with implementing monitoring systems for honeybee hives. The motivation for this project was the collapse of a large percentage of hives (as high as 50%) in recent years due to various causes that were previously referred to as the Colony Collapse Disorder (CCD) [2]. We built a system called Beemon that has evolved through several different microcontrollers over the life of the project. The current system accommodates several sensors for collecting data and uploading them to our server for analysis. Beemon obtains humidity, temperature, and audio recordings from inside the hives, video recordings at their entrance, as well as their weight. This system involves building various hardware components, writing software to interact with them, writing drivers specific to the sensors, and a website that provides generic information and tiered data access. These tasks require students to gain a range of skills and successfully transfer them to their peers, as generations of students go through our lab. These students often start as sophomores and leave as seniors or graduate students at the end of their studies. In addition to building and writing code for the Beemon system as a team, we have also produced research products such as journal publications, conference presentations, seminars, and invited talks in collaboration [3-15]. All these publications were in collaboration with graduate and undergraduate students. Students

work on the project for a limited time between three-to-six semesters after their initial training, making tangible contributions and continuity of their work challenging. Our bee-inspired Learning Mentoring Model consists of four principles: (1) Guided Participation, (2) Peer Learning through Demonstration, (3) Adaptive Role Transitions, and (4) Distributed Knowledge Networks, which we will elaborate further in the following sections.

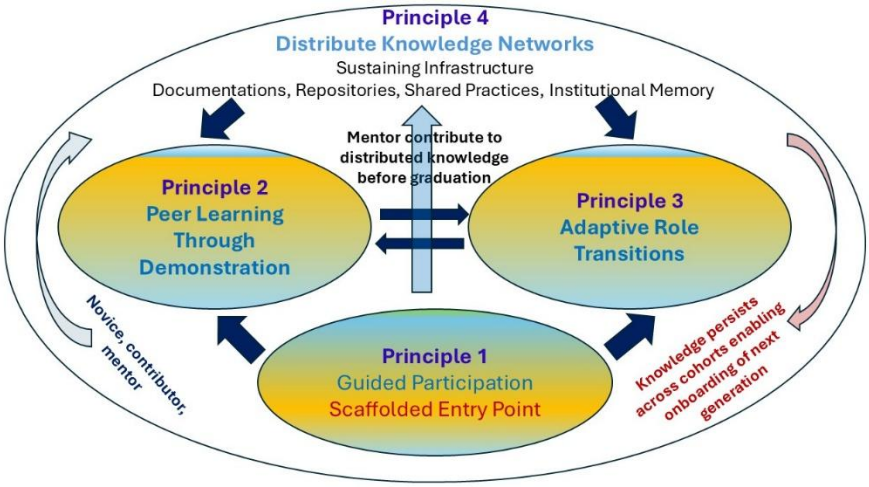

Fig 1. Conceptual diagram of the Bee Learning Mentoring Model. Guided Participation (Principle 1) provides the scaffolded entry point for new students. Peer Learning through Demonstration (Principle 2) and Adaptive Role Transitions (Principle 3) operate as interconnected mechanisms: as students learn from peers, they progress through roles from novice to contributor to mentor, with each process enabling the other. Distributed Knowledge Networks (Principle 4) serves as the sustaining infrastructure — documentation, repositories, shared practices, and institutional memory that persist across student cohorts. The feedback arrow indicates the model's cyclical nature: each graduating mentor's contributions to the distributed knowledge network provide the resources that support guided participation for the next cohort, ensuring continuity despite turnover.

1.1 Contributions

This paper makes the following contributions to CS education and mentoring research:

1. We introduce the Bee Learning Mentoring Model, a four-principle framework, Guided Participation, Peer Learning through Demonstration, Adaptive Role Transitions, and Distributed Knowledge Networks, grounded in honeybee social learning and socio-cultural educational theory.
2. We present an eight-year longitudinal case study demonstrating the model's application in a CS research lab with continuous student turnover, providing evidence of its sustainability and effectiveness.
3. We report measurable outcomes associated with the model, including 100% degree completion among 44 participating students, 18 co-authored publications, and 29 conference presentations.
4. We bridge biological distributed learning systems and CS education practice, offering a transferable framework for other research labs facing similar mentoring challenges.

Our model relates to but extends existing frameworks. Vygotskian scaffolding [16] emphasizes guided support within the zone of proximal development but typically assumes stable expert-novice pairs. Lave and Wenger's communities of practice [24] describe how newcomers move toward full participation, but do not prescribe how to sustain knowledge transfer when participants regularly leave the community. Peer instruction models [30, 31] focus on same-status collaboration, but do not address progressive role transitions from novice to mentor. The Bee Learning Mentoring Model addresses these gaps by explicitly incorporating adaptive role transitions and distributed knowledge networks as structural mechanisms to maintain continuity as students cycle through the lab every three to six semesters.

The model is both descriptive and prescriptive. It has emerged through longitudinal reflective practice, eight years of observing, adjusting, and refining our lab's mentoring processes, rather than through

formal grounded theory methodology. The honeybee analogy serves as a conceptual lens that helps us identify and articulate patterns already present in our practice. We present it as a prescriptive framework that other CS research labs can adapt to their own contexts.

The remainder of this paper is organized as follows. Section 2 discusses Guided Participation and Peer Learning, the first two principles of our model, grounding them in socio-cultural learning theory and describing their implementation in our lab. Section 3 examines Role Modeling, addressing how observation, imitation, and professional behavior shape student development. Section 4 presents Distributed and Networked Learning, drawing on honeybee communication research and connectivist learning theory to explain how knowledge flows across our team. Section 5 discusses the challenges, limitations, and transferability. Section 6 concludes with outcomes, future work, and broader implications.

2. Guided participation and learning

The first two principles of the Bee Learning Mentoring Model, i.e., Guided Participation and Peer Learning through Demonstration, are grounded in socio-cultural learning theory and form the foundation of daily mentoring practice in our lab. We first discuss the theoretical basis before describing our implementation.

Guided learning in computer science refers to instructional approaches in which learners receive structured support, scaffolding, and feedback from instructors, tutors (or tutoring systems), or peers as they engage with computational concepts and problem-solving tasks. Rooted in constructivist and socio-cultural learning theories, guided learning helps students bridge the gap between novice understanding and expert practice by gradually reducing support as competence increases [16, 17]. In computer science education, guided learning has been shown to improve conceptual understanding, programming skills, and retention by combining active learning with explicit guidance, such as worked examples, formative feedback, and collaborative problem solving [18, 19]. Well-designed guidance is particularly effective for novices, as it reduces cognitive load while fostering deeper learning and transferable skills essential for computing disciplines [20, 21]. Honeybees demonstrate analogous guided participation: colony members transition through roles based on social exposure and experience [36], and younger bees acquire communicative competence through observation of experienced dancers [38]. These biological findings have also been applied as pedagogical tools in STEM education [1]. This parallel between biological and educational guided participation has informed the first principle of our mentoring model.

Guided participation is a sociocultural learning process in which novices develop knowledge and skills through active engagement in culturally meaningful activities, supported by more knowledgeable peers. Originating in Vygotskian theory, guided participation emphasizes learning as a collaborative, situated process that unfolds within the zone of proximal development, where guidance, modeling, and shared problem-solving enable learners to gradually assume greater responsibility. In [16, 22, 23], authors further conceptualized guided participation as the coordination of cognitive, social, and communicative processes between experts and learners, highlighting how participation in everyday practices fosters the internalization of cultural tools and ways of thinking. Empirical research has shown that guided participation enhances conceptual understanding and transfer by embedding instruction within authentic activities and social interaction, making it especially effective in educational contexts that value apprenticeship, collaborative learning, and scaffolding [24, 25].

While these theoretical frameworks provide robust foundations for understanding guided and peer learning, they share a common limitation: they typically assume stable learning communities where experts remain available to mentor novices over extended periods. In CS research labs, however, student populations turnover every few semesters, and the experts of one year become alumni the next. Existing mentoring models in CS education [18, 19, 30, 31] address classroom settings with fixed cohorts but rarely account for the challenge of sustaining knowledge transfer across successive generations of student researchers. Our Bee Learning Mentoring Model addresses this gap by drawing on honeybee colony organization, where the colony sustains collective competence despite the continuous turnover of individual bees through distributed knowledge, adaptive role transitions, and socially embedded learning processes.

In our project, we take full advantage of this, as more experienced students, research staff, and faculty provide structural support and guidance to the new and less experienced students. We organize the

team into smaller working groups that meet daily, led by more experienced members, with weekly full-team meetings to maintain cohesion and cross-pollinate knowledge across projects. As students work collaboratively on projects, we provide formative feedback to the team and the individuals. Initially, several peers may serve as mentors to a new student, but as these students get assigned to a specific task, a domain-experienced mentor guides the mentee. We also have extensive documentation through GitHub and wiki pages that students continuously update. The GitHub repository also serves as a version control system for software development. The GitHub repository provides the historical evolution of the project.

In our project, students who have completed the CS2 course are recruited to participate in meaningful and rewarding research. They are excited about their work being in line with saving the bees' efforts [26]. The first step for those who are selected to participate is to complete a Responsible Conduct of Research (RCR) managed by our Office of Research and go through a crash course in Python. We use the Codecademy course for this training. During this training, several smaller projects are provided by the mentors to guide the students toward research-specific problem-solving and programming. For example, one of the early programming projects deals with reading image data and computing the entropy of that image. This requires mentors' engagement during the implementation of the mathematical model of entropy, the testing of it, and the understanding of its significance in our project. After completion of the course, students participate in our weekly meetings to learn about various projects we are working on and how they can contribute. We often have a range of open hardware, software, database, and/or web problems to be completed. Students can participate in a project that is more in line with their interests. However, they have opportunities to contribute to other projects if their interest shifts along the way. Even though some students come to us with various levels of experience, as far as our honeybee project is concerned, they are assumed to be novices, thus getting trained in deficit skills as they make progress.

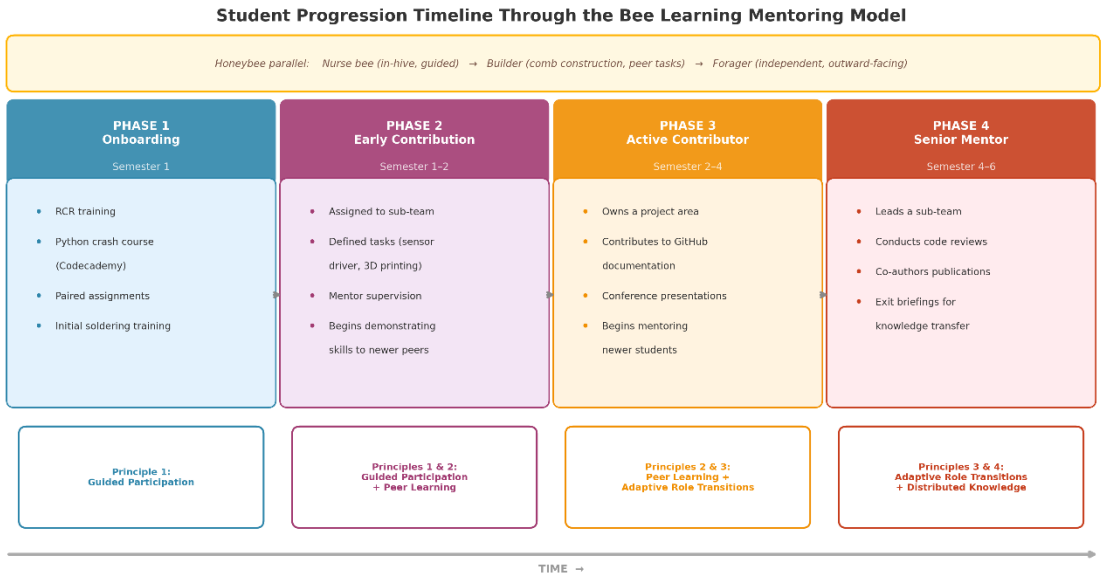

Fig 2. Typical student progression timeline through the Bee Learning Mentoring Model, mapped against the four mentoring principles. Students enter as novices and progressively assume greater responsibility, culminating in mentoring roles. This trajectory mirrors the honeybee developmental progression from nurse bee (in-hive tasks under social guidance) to forager (independent, outward-facing contributions) described by Johnson [36]

In our project, peer learning is a great instructional approach in which students engage collaboratively with others of similar status to construct knowledge, develop skills, and reflect on understanding through mutual interaction. Grounded in social constructivist theory, peer learning emphasizes dialogue, shared problem solving, and reciprocal teaching, as mechanisms through which learners co-construct meaning and extend their cognitive development [16, 27]. Research indicates that peer learning can enhance conceptual understanding, motivation, and metacognitive awareness by encouraging

learners to articulate reasoning, confront alternative perspectives, and provide feedback to one another [28, 29]. Well-structured peer learning environments—such as peer instruction, cooperative learning, and peer tutoring— support deeper learning and improved academic outcomes when roles, tasks, and expectations are clearly defined [30, 31]. Honeybees illustrate peer-mediated learning through observational exposure: bees that observe experienced dancers produce more accurate waggle dances than those without such social exposure [38]. Similarly, in our lab, students learn from peers at comparable experience levels, challenging each other and sharing discoveries as they develop shared competence.

Table 1: A short list of the major tasks in our project

Hardware	Tasks
Soldering	● Scale load cells, controller, quality control, testing ● Humidity/Temperature connectors, quality control, testing ● Microcontroller's header, quality control, testing ● Assembly of microcontroller's sensors (hum/temp, camera, microphone, camera, & scale
3D Printing	● Designing in Fusion 3 and printing the Beemon box ● Designing in Fusion 3 and printing the camera holder ● Designing in Fusion 3 and printing the hum/temp cavity ● Designing in Fusion 3 and printing the load cell brackets ● Designing in Fusion 3 and printing the entrance and landing plate
Quality Control/ Testing	● Quality of soldering ● Ensuring accuracy ● Ensuring connectivity ● All 3D printed pieces
Software	
Coding	● Beemon (the main controller, testing and debugging, updates, …) ● Driver for hum/temp sensor ● Driver for microphone ● Driver for video camera ● Driver for the load balance controller
Network	● Hives' fleet management and maintenance ● Hives' data upload to the server ● Hives' firmware update ● Hives' monitoring and diagnostic system
Web	● Project's website ● Live and archived video system ● Database design, implementation, and expansion ● APIs for data retrieval

These are also core to the success of our project. We usually have five or six undergraduate and one or two graduate students working in the lab to perform the tasks shown in Table 1. There are some tasks that all our students should know, but there are also tasks which are specific to each of them. For example, all students need to learn how to solder electronic components. So, during or after their Python training, they do some soldering under the supervision of a mentor. A mentor works with them to help them gain some level of expertise and checks the quality of their soldering work. The reason students need to gain this skill is because almost all the software development work can be done remotely, if needed, but if a soldering task is needed, we need to have someone in the lab to take care of it. Since all students learn to solder, they challenge each other to produce the best and cleanest work.

The tasks in Table 1, each serve a specific function within the Bee Learning Mentoring Model. Soldering and quality control tasks serve as entry points for Guided Participation (Principle 1): they are concrete, observable, and require direct mentor oversight, allowing new students to experience scaffolded learning in a low-risk context. 3D printing tasks exemplify Peer Learning through Demonstration (Principle

2): students who master Fusion 3 design and printer operation teach others through hands-on demonstration, and the iterative nature of prototyping encourages collaborative troubleshooting. Software coding tasks, from driver development to the main Beemon controller, span the full project lifecycle and illustrate Adaptive Role Transitions (Principle 3): a student may begin by writing a simple sensor driver under close supervision, progress to maintaining and debugging the main controller, and eventually leads the software team while mentoring newer students. Network and web tasks embody Distributed Knowledge Networks (Principle 4): they require coordination across multiple students and rely on shared documentation, version control histories, and API specifications that persist independently of any individual team member.

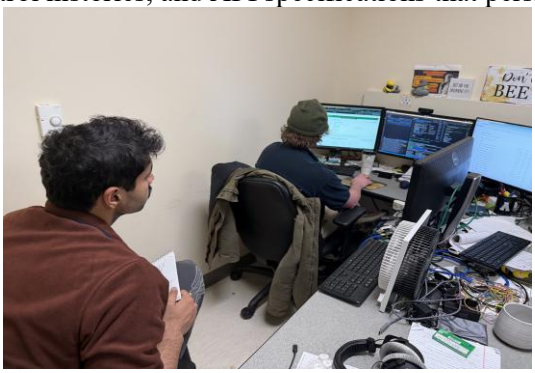

Fig 3a. A graduate mentee receiving scaffolded instruction from research staff, illustrating the Guided Participation principle (Principle 1) in which experienced members provide structured support during hands-on tasks.

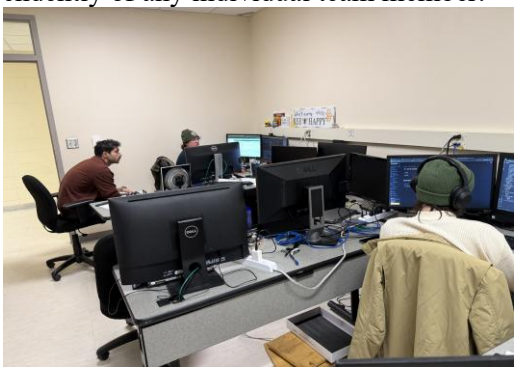

Fig 3b. One of the project's incubation spaces, where hardware assembly and testing tasks provide early guided learning opportunities for new students.

3. Role modeling

Beyond structured guidance and peer collaboration, learning in our lab also occurs through observation and imitation, the basis of the third principle of our mentoring model: Adaptive Role Transitions through Role Modeling. Role modeling emphasizes that learning occurs not only through direct instruction but also through observation, imitation, and the interpretation of others' actions within a social context [32]. In educational and professional settings, effective role models demonstrate both technical competence and social or ethical behaviors, thereby shaping learners' motivation, self-efficacy, and identity development [33]. Research suggests that exposure to credible and relatable role models can enhance performance and persistence, particularly when models are perceived as attainable and when reflective guidance accompanies observation [34, 35].

Understanding honeybee social learning, therefore, contributes not only to human learning theory but also to the design of our system that scaffolds learning through collective intelligence. In our project, similar principles guide the design of collaborative learning platforms (and adaptive systems) that support peer mentoring and distributed expertise. Professional behavior and work ethics are something we value in our lab, and students don't have to go far to see role models to follow. One of the most interesting tasks in our project is the 3D printing of the parts. We have endeavored to have at least two students comfortable with Fusion 3 for designing and creating *gcodes,* and working with 3D printers, although almost everyone in the lab knows how to operate and maintain the printers. The way students approach 3D printing, which is not a technical or programming skill, has been very interesting. Students who figure something out both in the design and operation show others how it is done. We have a master of 3D printing who has been amazing in mentoring others regarding how to design and print parts as they constantly get updated.

This pattern of role modeling extends beyond individual skills to professional practices. Students observe senior lab members: presenting at conferences, leading code review sessions, or collaborating with faculty on manuscript drafts. This allows them to internalize both the technical and professional standards of a research community. For example, one student who joined the lab as a sophomore with no prior hardware experience observed a senior student's approach to systematic debugging during scale calibration. Within two semesters, this student adopted the same methodical approach and was teaching it to newer

members, demonstrating the adaptive role transition from observer to practitioner to mentor that characterizes Principle 3 of our model. This progression mirrors honeybee role transitions from nurse bee to forager, where experience and social context (rather than fixed instruction) determine new roles [36].

4. Models for Distributed and Networked Learning

While the preceding sections focused on mentoring interactions between individuals and small groups, much of the knowledge in our lab resides not in any single person but in the network of documentation, repositories, and shared practices that persist across student generations. This section discusses the fourth principle of our model, Distributed Knowledge Networks, and its parallels in honeybee colony intelligence. Honeybee colonies function as distributed learning systems, where no single individual possesses complete knowledge, yet collective intelligence emerges through local interactions and social signals. Each bee operates with partial information, and colony-level learning arises from communication mechanisms such as the waggle dance and pheromonal feedback [36]. This model aligns closely with connectivist learning theory, which conceptualizes learning as the formation and navigation of networks rather than the internalization of static knowledge [37]. Within the hive, bees transition between roles—such as nursing, comb construction, and foraging—based on experience, social cues, and colony needs rather than fixed instruction [36]. These role changes are gradual and reversible, emphasizing adaptability.

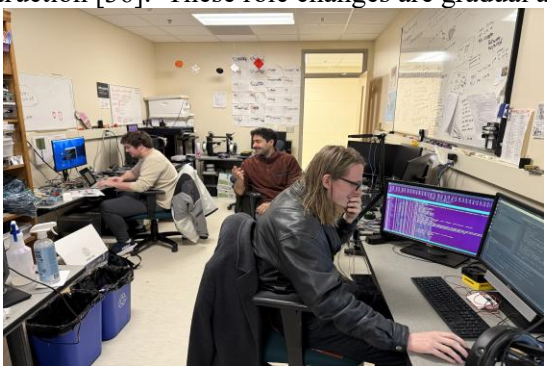

Fig 4a. An additional lab workspace used for prototyping and collaborative troubleshooting, supporting both Peer Learning through Demonstration (Principle 2) and Adaptive Role Transitions (Principle 3) as students work alongside more experienced peers.

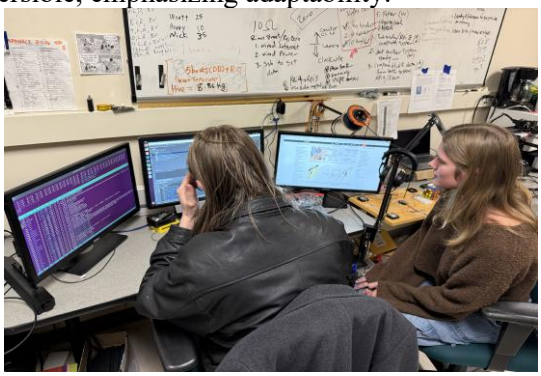

Fig 4b. Students collaboratively diagnosing a challenging hardware issue. This type of shared problem-solving mirrors honeybee collaborative foraging and illustrates how distributed expertise within the team compensates for individual knowledge gaps.

In our project, students routinely learn through distributed systems, including collaborative repositories, discussion meetings, version-control platforms, and peer networks. As in a bee colony, effectiveness depends not on centralized instruction alone but on network participation and information flow. Our project mirrors this pattern, as students progress from novices to contributors and eventually mentors through participation in projects and collaborative problem-solving environments. We allow students to assume increasing responsibility in ways that reflect adaptive role progression, a principle strongly supported by the honeybee model.

The honeybee waggle dance illustrates how distributed knowledge transfer operates in practice. This symbolic communication system conveys spatial information about food sources, and recent research demonstrates that dance accuracy improves through social learning; bees that observe experienced dancers perform more precisely than those without such exposure [38]. Nieh and colleagues [41] showed that this communicative competence can be culturally transmitted across individuals, and that bees exhibit a critical period for acquiring it — analogous to how early mentoring experiences shape a student's ability to communicate technical knowledge effectively. These findings, along with broader evidence of cross-generational knowledge transfer in social species [39, 40], reinforce a central insight for our mentoring model: knowledge systems that depend on social transmission must actively involve early, structured

exposure to experienced practitioners. In our lab, this translates to pairing new students with mentors from their first week and embedding them in collaborative activities before expecting independent contributions.

Undergraduate and graduate students usually join our lab at different stages of their studies and with varying backgrounds in CS. Often, more than one student, usually a pair, joins our team and goes through some initial training together. We ask them to complete tutorials and work together to complete assignments. The more advanced students serve as mentors and give them challenges to figure out during this period, and tutees return to them as reliable resources. We also ask the trainees to share their findings with each other immediately by demonstrating their work to each other. Student trainees demonstrate their mastery of problem-solving and code writing to their mentors and research staff.

Honeybees flexibly choose between personal experience and socially acquired information. When environmental uncertainty or foraging costs are high, bees rely more heavily on social signals from nestmates, demonstrating adaptive social learning strategies [41]. We have experienced this firsthand in our project throughout its lifetime, as we have enhanced our hardware and software development process in a similar manner. Our initial codes were written in C++, and our limited sensors were controlled by Arduino. Our transition to Python was motivated by a desire to run programs more efficiently, as well as to utilize existing libraries for analysis, instead of writing them from scratch. This transition was inspired by the experience students have shared with us (and their peers in the lab). Our hardware has hence evolved from the Arduino, with a limited sensor capability, to the current Raspberry Pi4B system.

Research in self-regulated learning suggests that expert learners develop sensitivity to context and informational reliability, echoing the adaptive information strategies seen in bees [42]. Honeybees dynamically balance individual exploration with socially acquired information depending on environmental uncertainty and cost. When personal information is unreliable or costly, bees rely more heavily on social cues from nestmates [41]. This adaptive strategy reflects a form of metacognitive regulation. In our project, students similarly decide when to rely on personal experimentation versus communal knowledge sources such as documentation or mentors. The honeybee model thus provides a biologically grounded analogy for metacognitive decision-making in technical education.

Studies demonstrate that honeybees require social exposure to experienced individuals to accurately perform the waggle dance, a complex symbolic communication behavior [38]. Bees deprived of this exposure exhibit persistent errors, indicating that observation and social feedback are essential for mastery. Division of labor, feedback loops, and role transitions in hives parallel mentoring structures where novices progress toward expertise through observation, practice, and social feedback [38]. This phenomenon parallels apprenticeship and mentoring models in our project, wherein novices develop expertise through observation, guided participation, and feedback from more experienced peers. Lave and Wenger's theory of *legitimate peripheral participation* describes how learners move from observation to full participation within a community of practice—precisely the pattern observed in honeybee colonies [24]. In our project, practices such as pair programming, code review, and design walkthroughs function as social learning signals that scaffold novice development. As students grow in our project, they take more ownership and move toward an advanced level of understanding. They know they are valued and take apprenticeships, mentoring, and role modeling to a new level.

5. Discussion

5.1 Challenges of Continuity

The most persistent challenge in our mentoring model is the cyclical loss of institutional knowledge. Students typically contribute for three to six semesters before graduating, and despite our structured onboarding and documentation practices, some knowledge inevitably leaves with them. Tacit knowledge — the intuition a student develops for diagnosing a sensor malfunction or the unwritten conventions in our codebase — is particularly difficult to transfer. We have mitigated this through several strategies: overlapping cohorts so that at least one experienced student remains during each transition period, maintaining detailed GitHub documentation and wiki pages that capture not just procedures but rationale and troubleshooting histories, and conducting exit briefings where departing students present their work and unresolved challenges to the continuing team. Nevertheless, we acknowledge that some loss of

momentum occurs with each transition, and certain specialized skills (such as advanced 3D printer maintenance or custom driver optimization) occasionally need to be relearned rather than transferred.

5.3 Limitations of the Biological Metaphor

While the honeybee analogy has been a productive conceptual tool for our mentoring model, we recognize its boundaries. Honeybees operate through instinctive behaviors shaped by pheromonal signals and evolutionary pressures; students exercise agency, make deliberate choices, and respond to individual motivations, anxieties, and career goals. The metaphor is inspirational and structural, not isomorphic. We do not claim that student mentoring should replicate bee colony dynamics. Rather, we use the honeybee model as a lens to identify patterns — distributed knowledge, adaptive role transitions, socially mediated learning — that are independently supported by educational theory [16, 24, 37] and that we have observed in our practice. Readers should interpret the biological parallels as a framework for thinking about mentoring design, not as a literal mapping between insect and human behavior.

5.4 Transferability and Lessons for Other Labs

Based on our eight years of experience, we suggest the following conditions for successfully adopting the Bee Learning Mentoring Model in other CS research labs:

1. A project with sufficient complexity and breadth to support multiple concurrent tasks at varying difficulty levels, enabling students to progress from simple to complex contributions.
2. A team size of at least four to six students at any time, to ensure overlapping expertise and the possibility of peer mentoring relationships.
3. A commitment to documentation infrastructure (version control, wikis, shared repositories) that persists independently of any individual student.
4. Faculty or staff who serve as long-term anchors for institutional knowledge, providing continuity that complements the student-to-student mentoring.
5. Structured onboarding that standardizes baseline skills while allowing flexibility in subsequent task assignments based on student interest and aptitude.

Labs with smaller teams or shorter-term projects may need to adapt the model, for example, by relying more heavily on documentation and less on overlapping cohorts for knowledge transfer.

5.5 Evidence and Limitations of Outcome Data

The outcomes reported in this paper are drawn from departmental enrollment and graduation records, lab participation logs maintained since the project's inception, GitHub contribution histories, and publication records. While we did not conduct a controlled experiment comparing our mentoring model to alternatives, the consistent pattern of degree completion, scholarly productivity, and career placement across eight years and 44 students — spanning multiple faculty, staff, and student cohorts — provides sustained observational evidence of the model's effectiveness. We acknowledge that factors external to the mentoring model (departmental support, project funding, student self-selection) also contributed to these outcomes, and we do not claim strict causation. Future work should incorporate formal assessment instruments such as self-efficacy surveys, skill transfer rubrics, and comparative studies with labs using different mentoring approaches.

We have observed 100% degree completion among both graduate and undergraduate students who have worked on our project. In total, 44 undergraduate (68%) and graduate students (32%) have been engaged in our project since the beginning. About 57% have joined the job market, and 43% have pursued or are pursuing graduate studies. These students have expressed an interest in the fields of IoT, Embedded Systems, Software Development, Data Science, Artificial Intelligence, and Machine Learning, to which they have had exposure while working on our project. These students have participated in meaningful, rewarding research that helps save the bees. They worked in a successful mentoring environment that produced tangible products and scholarly work. Our students have co-authored 18 honeybee-related papers and 29 conference presentations with their mentors.

5.6 Representative Student Trajectories

To illustrate how the mentoring model shapes individual trajectories, we briefly describe three representative student experiences (details anonymized):

Student A joined the lab as a sophomore with no hardware experience. During the first semester, they completed the Python crash course and soldering training under mentor supervision (Guided Participation). By the second semester, they had taken ownership of the humidity/temperature sensor driver and were teaching soldering techniques to a new cohort (Peer Learning, Adaptive Role Transition). By their senior year, they were leading the software team, co-authored two conference papers, and mentored three incoming students. They are now a software engineer at a technology company working on IoT systems.

Student B entered as a graduate student with strong programming skills but no embedded systems experience. Their initial onboarding focused on the hardware-software interface of the Beemon controller. Within one semester, they had improved the fleet management system and contributed a novel data upload protocol. They served as the primary mentor for undergraduate hardware teams during their second year (Distributed Knowledge Networks) and co-authored a journal paper. They continued on to enroll in a doctoral program in computer engineering.

Student C joined with limited confidence in programming and initially struggled with the Python course. Paired with a peer at a similar level and supported by a graduate student mentor, they gradually built competence through small, scaffolded tasks. By their third semester, they had developed the landing plate 3D printing design and were regularly demonstrating it to newer students. They graduated on time and entered a data science position.

These trajectories illustrate the model's capacity to accommodate students with diverse backgrounds and skill levels while supporting consistent progression from novice to contributor to mentor.

6. Conclusion

This paper presented the Bee Learning Mentoring Model, a four-principle framework for sustaining effective mentoring in CS research labs where students cycle through every three to six semesters. Drawing on parallels between honeybee colony organization and socio-cultural learning theory, the model is organized around Guided Participation, Peer Learning through Demonstration, Adaptive Role Transitions, and Distributed Knowledge Networks.

Over eight years of application within the Beemon project at Appalachian State University, the model has supported 44 undergraduate and graduate students, all of whom completed their degrees. Students co-authored 18 papers and delivered 29 conference presentations with their mentors, and the project has sustained continuous operation despite the complete turnover of the student team multiple times. These outcomes suggest that embedding adaptive role transitions and distributed knowledge networks as structural features of a mentoring program can mitigate the knowledge loss that typically accompanies student turnover in research labs.

Several directions for future work emerge from this experience. We plan to formalize assessment through self-efficacy surveys and skill transfer rubrics, explore scaling the model to multi-institution and remote collaborations, and investigate AI-assisted onboarding tools that complement human mentoring. More broadly, the model's principles are applicable to any context characterized by high participant turnover and complex collaborative work — from interdisciplinary research teams to industry engineering groups onboarding new members into large systems. By aligning mentoring practice with the principles of distributed knowledge systems, we offer a path toward resilient and sustainable learning environments.

7. Acknowledgments

The authors wish to thank the NC General Assembly, the NC Innovation, and Lowe’s Hardware Inc. for their financial support of the honeybee project at App State. We also wish to thank the Department of Computer Science, Office of Sustainability, and Office of Campus Planning for their support. Our special thanks go to many graduate and undergraduate students who have been part of the project and made the implementation of our mentoring model successful by producing exemplary work.