On the Potential of Agentic Workflows for Animal Training Plan Generation

Jörg Schultz [1,*]

[1] *Tier Wohl Team GbR, Rödelsee, Germany*

Correspondence*:
Jörg Schultz, Tier Wohl Team GbR, Am Mühlschutz 2, 97348 Rödelsee, Germany
Joerg.Schultz@Tier-Wohl-Team.de

ABSTRACT

Effective animal training depends on well-structured training plans that ensure consistent progress and measurable outcomes. However, the creation of such plans is often time-intensive, repetitive, and detracts from hands-on training. Recent advancements in generative AI powered by large language models (LLMs) provide potential solutions but frequently fail to produce actionable, individualized plans tailored to specific contexts. This limitation is particularly significant given the diverse tasks performed by dogs—ranging from working roles in military and police operations to competitive sports—and the varying training philosophies among practitioners. To address these challenges, a modular agentic workflow framework is proposed, leveraging LLMs while mitigating their shortcomings. By decomposing the training plan generation process into specialized building blocks—autonomous agents that handle subtasks such as structuring progressions, ensuring welfare compliance, and adhering to team-specific standard operating procedures (SOPs)—this approach facilitates the creation of specific, actionable plans. The modular design further allows workflows to be tailored to the unique requirements of individual tasks and philosophies. As a proof of concept, a complete training plan generation workflow is presented, integrating these agents into a cohesive system. This framework prioritizes flexibility and adaptability, empowering trainers to create customized solutions while leveraging generative AI's capabilities. In summary, agentic workflows bridge the gap between cutting-edge technology and the practical, diverse needs of the animal training community. As such, they could form a crucial foundation for advancing computer-assisted animal training methodologies.

Keywords: Computer Assisted Animal Training, LangGraph, agent orchestration, modularity in AI systems,Welfare-aware AI systems, Training plan customization, Handler support tools, Task-specific training workflows

1 INTRODUCTION

As a professional animal trainer, creating structured training plans is not merely a helpful tool but a foundational practice that enhances the effectiveness and consistency of training. These plans serve two essential purposes: they provide a clear roadmap for progressive skill development and act as vital documentation for tracking progress and ensuring accountability. However, despite their undeniable value, the process of writing training plans can be both time-consuming and repetitive. Trainers often find themselves devoting significant effort to tasks that involve structuring similar steps for different contexts, leaving less time for the hands-on interaction and observation essential to effective training. Streamlining the creation of these plans could enable trainers to dedicate more time to working directly with their animals, improving outcomes while reducing the burden of administrative work.

Recent advancements in artificial intelligence, particularly in the field of generative AI, offer a solution by supporting processes such as writing training plans. Generative AI refers to a category of machine learning models capable of producing coherent and creative outputs, including text, code, and even images, by leveraging patterns learned from vast datasets (1, 2, 3). These models, powered by large language models (LLMs) like OpenAI's GPT, have demonstrated remarkable capabilities across a diverse range

of tasks. For instance, generative AI simplifies complex programming tasks by generating code snippets or even entire programs [1] . Additionally it has also excelled in intellectually demanding areas, such as high-stakes competitions like the International Biology Olympiad (5) , where it has provided accurate and contextually appropriate answers. These achievements highlight the adaptability and problem-solving capacity of generative AI, making it a compelling candidate for assisting in the creation of training plans (6).

Still, relying on standard chatbots like ChatGPT to produce training plans currently reveals significant limitations. Although the generated plans may read well, they often lack the specificity required for actionable progressions with clearly defined steps. Furthermore, such plans are typically generic and may fail to adhere to agency-specific training standards, including critical welfare considerations. Standard prompting approaches do not incorporate standard operating procedures (SOPs), leaving a gap in consistency and compliance with established guidelines. Additionally, these chatbots cannot effectively interact with trainers to iteratively refine goals or ensure that all relevant information—such as the animal's health conditions or unique circumstances—is taken into account. These shortcomings highlight the need for a more robust, structured approach to harness the capabilities of generative AI for creating truly actionable and individualized training plans.

Writing a detailed training plan typically involves several key steps. The process begins with a thorough understanding of the behavior to be trained, including its purpose and the context in which it will be performed. The behavior is then broken down into simpler, manageable components, each representing a foundational element of the final goal. For each of these components, detailed progression plans are generated, outlining how to build the necessary skills step by step. An AI system designed to reproduce these steps would need to mimic this decomposition and progression process, handling each subtask methodically. A promising framework that aligns with this approach is the *agentic workflow* , inspired by modular capabilities in generative AI. Here, autonomous working units—*agents* —specialize in specific tasks, such as structuring progressions, incorporating welfare guidelines, or ensuring compliance with training standards. When orchestrated in a well-defined manner, these agents collaborate, share information, and refine outputs, providing a systematic and adaptable framework for leveraging generative AI to its full potential (7).

The concept of agentic workflows is not just theoretical; it is a rapidly emerging field with promising applications across diverse domains. For example, the *ChatDev* system demonstrates the power of communicative agents in software development (8). This workflow orchestrates specialized agents to collaboratively perform tasks such as designing, coding, and testing software, mimicking a streamlined development team. Similarly, in the realm of human behavior simulation, *Generative Agents: Interactive Simulacra of Human Behavior* showcases how agentic workflows can create lifelike simulations, enabling detailed studies of complex interactions between autonomous agents in virtual environments (9). Another compelling example is the *Agent Hospital* , which employs medical agents to simulate hospital operations, allowing for the exploration of adaptive strategies in healthcare systems (10). In each case, the modularity, adaptability, and task specialization of agentic workflows have proven instrumental in addressing domain-specific challenges efficiently and effectively. These examples illustrate the significant potential of agentic workflows to tackle complex, multifaceted challenges by decomposing them into manageable tasks. If agentic workflows have proven so effective in domains such as software development, human behavior simulation, and healthcare operations, it is compelling to explore whether they could also address the challenge of generating training plans. Given the modularity and adaptability of these workflows, their application to training plan creation could open new possibilities for improving both efficiency and quality in animal training.

[1] https://openai.com/index/learning-to-reason-with-llms/

This manuscript aims to explore the potential of agentic workflows for training plan generation by presenting a team of AI agents capable of creating individualized, actionable training plans. However, the goal is not to deliver a rigid, monolithic, one-size-fits-all solution. The rapid pace of technological advancements, particularly in generative AI, means static workflows risk becoming quickly outdated. Additionally, the range of tasks performed by dogs is vast, training philosophies vary, and different training teams have unique SOPs. To address these challenges, this work introduces adaptable building blocks—autonomous agents and teams of agents—that tackle discrete aspects of training plan generation. The modularity of these building blocks enables seamless integration of new AI models or refined task descriptions, allowing the system to evolve with future advancements without requiring a complete overhaul. Furthermore, these building blocks are intentionally designed for flexibility, enabling users to modify, combine, and repurpose them to meet the specific requirements of their training units or organizations. By empowering trainers and organizations to customize workflows, this approach ensures it remains relevant and responsive to evolving standards, technologies, and needs.

2 METHOD

2.1 Agentic Frameworks

The implementation of agentic workflows is greatly facilitated by specialized frameworks designed to coordinate multi-agent systems. These frameworks streamline the development, deployment, and orchestration of agents, enabling effective collaboration on complex tasks. Table 1 summarizes several widely used frameworks, highlighting their key features and suitability for different applications.

For this work, *LangGraph* was chosen due to its emphasis on control, reproducibility, and adaptability—qualities essential for generating training plans for living organisms, such as dogs. LangGraph's graph-based architecture enables precise orchestration of agent interactions, allowing each step in the workflow to be explicitly defined, monitored, and adjusted as needed. This transparency ensures that the process remains consistent, traceable, and aligned with ethical standards. Unlike other frameworks that prioritize rapid iteration or emergent behaviors, LangGraph provides fine-grained control over agent operations, minimizing the risk of unintended actions. Its modular design supports the incorporation of domain-specific constraints, such as welfare principles and progression guidelines, ensuring that outputs meet the stringent requirements of animal training. Furthermore, LangGraph facilitates automated testing at each stage of the workflow, validating intermediate outputs and ensuring compliance with predefined standards, which enhances both safety and reproducibility.

These features make LangGraph particularly suited for applications involving living beings, where safety, predictability, and ethical considerations are paramount. Additionally, its adaptability allows for seamless integration of new technologies or evolving training methodologies, ensuring that the system can grow and improve without requiring a complete redesign.

2.2 Agent Implementation

2.2.1 Core Agent

The agentic workflow is built upon a modular architecture, where each agent is designed to handle a specific subtask of the training plan generation process. To ensure consistency and adherence to defined coding standards, all agents inherit from a shared base class (Figure 1). This base class enforces a uniform structure and provides essential functionality, such as naming and action definition.

The *BaseAgent* class ensures that each agent:

- Defines a unique NAME attribute to identify its role within the workflow.
- Implements a static action method to perform its primary task.
- Includes optional utility methods to enable interaction in different environments.

Each agent follows a standardized workflow that consists of three main steps:

1. State Interaction: Retrieve relevant information from the shared memory or state.
2. Prompt Construction: Prepare input for the large language model (LLM). This often includes a background story (system message) and a specific task prompt (human message).
3. LLM Invocation and Output Handling: Call the LLM to generate output and update the shared memory with the result.

This design ensures modularity, allowing agents to be independently developed, tested, and integrated into the broader workflow. By leveraging the modular capabilities of LangGraph, these agents can collaborate seamlessly, with their interactions explicitly defined and traceable.

2.2.2 Enhancing Agent Functionality

Agents in this workflow are designed to handle specific tasks autonomously. Their functionality is significantly enhanced through two key features: the integration of external tools and the use of structured output formats. Together, these enhancements improve the adaptability, efficiency, and reliability of the system.

One critical aspect of agent functionality is their ability to autonomously determine when external tools are necessary to complete their tasks. **Tools** provide agents with access to additional resources or data, enabling them to refine outputs, address knowledge gaps, and adapt to dynamic requirements (11) . For example, data retrieval tools allow agents to query external resources, such as databases or online repositories, to gather up-to-date and task-specific information. Similarly, interaction-based tools enable agents to request clarification or additional input from human collaborators or other agents, facilitating problem-solving in scenarios where initial input may be incomplete or ambiguous. By analyzing the input data, agents can dynamically decide which tools to invoke, ensuring efficient operation while avoiding unnecessary or redundant usage.

Another essential feature of the workflow is the use of **structured output** formats. Unlike free-form text, which can vary across iterations or models, structured outputs adhere to predefined formats such as JSON or XML. This approach offers several benefits: structured outputs ensure consistency across multiple runs, even when generative models introduce randomness; they enable machine-readable results that can be directly parsed and utilized by other agents or systems; and they simplify validation processes, ensuring that outputs comply with task-specific requirements. These attributes make structured outputs vital for creating robust, modular workflows where multiple agents collaborate seamlessly.

By combining autonomous tool usage with structured output mechanisms, the system achieves a high degree of reliability and scalability. These features enable agents to perform their tasks effectively while ensuring that their outputs integrate seamlessly into the broader workflow. Together, these enhancements demonstrate the potential of agentic systems to address complex, real-world challenges with precision and adaptability.

2.2.3 Agent Collaboration

The true power of the agentic workflow emerges from interactions between agents, elevating the system beyond the capabilities of individual agents (12) . By collaborating in well-defined ways, agents can tackle complex tasks with efficiency and adaptability. Several interaction patterns can be employed, each suited to specific scenarios:

- **Sequential Collaboration** : A straightforward interaction where each agent's output serves as the input for the next in a predefined sequence. This approach ensures a logical progression from initial input to final result, making it ideal for tasks requiring step-by-step refinement.
- **Conditional Edges** : Agents can dynamically invoke other agents or teams based on the results of their tasks. This enables adaptive workflows that respond to varying input data or scenarios, ensuring flexibility in achieving desired outcomes.
- **Reviewer Pattern** : In this iterative approach, one agent reviews the output of another, providing feedback or suggestions for improvement (13). The original agent refines its output based on the review and resubmits it for evaluation. This pattern is particularly useful for tasks demanding high precision and specialization.
- **Sub Graphs as Teams** : Groups of agents form specialized sub-teams (sub-graphs), each focusing on a specific aspect of the overall task. These teams work independently, sharing selected information with the larger workflow. This modular structure simplifies complex tasks while maintaining controlled information flow.
- **Router (Manager)** : A central management agent directs tasks to the most appropriate team or agent based on predefined rules or contextual factors. This pattern is well-suited for workflows involving diverse subtasks or requiring task prioritization.
- **Map Reduce** : For tasks that can be divided into smaller, independent subtasks, this pattern enables parallel execution. A coordinator agent distributes the subtasks among agents, collects their outputs, and combines them into the final result. This approach is particularly effective for managing large or complex tasks efficiently (14).
- **Shared Memory**: Agents share information by accessing and modifying a centralized memory (15) . Separate memories can be used for different teams, with selected portions shared across the workflow. This maintains modularity while ensuring coherence in information exchange.

These interaction patterns, ranging from straightforward sequential workflows to dynamic and adaptive processes, enable agentic systems to efficiently address complex challenges. Selecting the appropriate pattern depends on task complexity, the need for adaptability, and the desired level of modularity and control.

2.3 Testing and Validation

Rigorous testing is a cornerstone of reliable software development, ensuring that individual components and systems function as intended (16, 17). However, when building agent frameworks that integrate large language models (LLMs), testing becomes even more critical. LLMs introduce a probabilistic element to the workflow, where outputs can vary based on subtle changes in input or model state. This variability makes testing essential, particularly when developing modular building blocks (agents) that need to function consistently across diverse scenarios.

In this workflow, testing must validate not only individual agents but also their interactions within teams. To achieve this, a multi-level testing strategy was employed:

1. **Unit Testing:** Unit tests validate the functionality of individual agents in isolation. These tests ensure that each agent interacts correctly with its dependencies, such as invoking a mocked LLM with the expected parameters. They also verify the agent's internal logic and state management, ensuring that intermediate outputs conform to the expected structure and logic. For example, a unit test might confirm that the correct input is passed to a mocked LLM and that the output adheres to predefined specifications.
2. **Integration Testing:** Integration tests assess the ability of agents to interact with real external systems, particularly LLMs. These tests verify that the LLM processes input data and produces responses with the expected structure or features. For example, an integration test might validate that the output includes key elements relevant to the task and adheres to the required format. Integration tests provide confidence in the agent's ability to function effectively in real-world scenarios.
3. **Probabilistic Output Validation:** Given the inherent variability in LLM outputs, exact matches with expected results are not always feasible. To address this, a validation step leverages another LLM call to evaluate outputs against predefined criteria. Instead of requiring verbatim equality, the validating LLM assesses whether the response includes essential elements and adheres to the desired format. This approach ensures robust testing while accounting for the probabilistic nature of LLM-generated outputs.

This three-tiered testing strategy ensures that agents and their interactions perform reliably. Unit tests provide a foundation for correctness by validating the internal logic and mocked interactions of individual agents. Integration tests confirm that agents operate effectively in real-world scenarios, interacting seamlessly with LLMs and producing structured outputs. Finally, probabilistic validation accommodates the inherent variability of LLMs by focusing on alignment with predefined criteria, ensuring robustness and flexibility. By adopting this comprehensive testing approach, the development of high-quality, reusable building blocks is supported. This strategy ensures that agents and teams function predictably, even as input conditions or details change, thereby reinforcing the modular and adaptable nature of the proposed workflow.

2.4 Code Availability

The software developed for this study, AI Agents for Animal Training Plan Generation, is available at https://github.com/Tier-Wohl-Team/AIAgents$_{TrainingPlan}$. The software is platform-independent and has been tested on Windows, and Linux operating systems. It is primarily written in Python and requires Python version 3.11 or higher due to dependencies associated with LangGraph. Users should ensure they have the appropriate Python version installed before running the software. The project requires several Python packages, which are listed in the requirements.txt file within the repository.

The repository includes a comprehensive suite of tests to validate both individual agents and their interactions. These tests can be executed using the ‘pytest‘ framework with specific markers for test selection: ‘pytest -m unit‘ runs unit tests, ‘pytest -m integration‘ runs integration tests, and ‘pytest -m llm‘ executes probabilistic output validation tests. The tests vary in runtime and cost depending on their complexity and external resource usage. Unit tests are fully local and execute quickly, as they mock dependencies and avoid external API calls. Integration tests, by contrast, involve calls to the LLM whenever required by the agent, leading to moderate runtimes and associated costs depending on the number of tests.

Probabilistic output validation tests, which include an additional LLM call to analyze the results of the agent's LLM interaction, have the longest runtimes and highest costs. This flexibility allows users to select specific testing levels based on their requirements and resource constraints.

The software is licensed under the MIT License, which permits use, distribution, and modification for both academic and non-academic purposes. Detailed instructions for installation, setup, and usage are available in the README.md file within the repository. Users must obtain an OpenAI API Key to utilize the default language model gtp-4o-mini and a Tavily API Key if employing the Internet Research Agent.

The version of the software used in this manuscript is tagged as `v1.0.0` in the repository.

3 RESULTS

3.1 Agents: Core Components of the Workflow

3.1.1 A Minimalistic Agent Design - Tackling Distraction Training

The *Distraction Specialist Agent* (full code in Supplementary Material, Section 1.1.1) exemplifies the foundational principles of the agentic workflow. It demonstrates how structured inputs, targeted prompts, and the use of a large language model (LLM) can produce actionable and specialized outputs. This agent is designed to address a common and critical challenge in dog training: ensuring that a trained behavior can be reliably performed even under distractions. Its workflow is both straightforward and effective, following a sequence of key steps. First, the agent accesses relevant information from the shared memory, including the behavior being trained, the current status of the dog, the desired goal, and any specific details about the dog's circumstances. Using this input, the agent crafts a prompt that combines a detailed background story with a task-specific instruction **??** , clearly defining the LLM's role in generating a tailored training plan (Figure 2).

The task prompt is designed to guide the LLM in generating highly structured and actionable outputs. By explicitly defining sections such as "CURRENT STATUS", "GOAL", and "INFORMATION ABOUT THE DOG" the prompt ensures that the response aligns with the expectations of the training plan framework. Additionally, the instruction to progress from easy to hard distractions provides a logical structure to the training plan, ensuring clarity and usability for novice trainers. Specific guidance, such as reacting to distractions that are too strong, further personalizes the output and prepares trainers for real-world scenarios. An example of a training plan generated by the agent, with the initial status of "Dog can sit when the trainer moves their arms" and the goal of "Dog stays sitting when a ball is thrown," is provided in the Supplementary Material, Section 1.1.2.

The response generated by the LLM is stored in the shared memory as a draft_plan, making it accessible to other agents as needed. This interaction with the memory not only enables seamless communication between agents but also ensures that the output remains adaptable for review or refinement by other specialized agents, such as the *Welfare Agent*. This design guarantees consistency and adaptability, enabling the *Distraction Specialist Agent* to handle a variety of distraction-related scenarios while aligning seamlessly with the overall agentic workflow strategy.

However, the *Distraction Specialist Agents*'s simplicity also presents a limitation: it relies entirely on the LLM's output without leveraging any additional sources or predefined guidelines. This dependence on a single source may reduce consistency or adherence to specific standards when generating plans. In the next section, I explore how integrating internal information, such as SOPs, can enhance an agent's ability to produce outputs that align with established best practices.

3.1.2 Incorporating External Information and SOPs – Adding Duration to a Behavior

The Duration Specialist Agent builds on the foundational principles of the agentic workflow while introducing additional functionality by incorporating SOPs into the training plan generation process. This agent illustrates how accessing external information, such as predefined guidelines, can enhance the quality and consistency of the generated plans, addressing some limitations of purely LLM-driven agents. Unlike the *Distraction Specialist Agent*, which relies solely on input data and the LLM, the *Duration Specialist Agent* accesses predefined SOPs stored in a configuration file. These SOPs outline progression steps for gradually increasing the duration or distance of a behavior following established progressions (18) . The agent retrieves the relevant progression steps from the configuration based on the current status and goal specified in the shared memory. This additional information is then seamlessly integrated into the prompt provided to the LLM. An important advantage of using an external configuration file is that it enables the agent's behavior to be easily adapted without modifying its underlying code. By simply updating the text based configuration file, users can define new progression steps or adjust existing ones to align with specific training requirements or standards. This approach showcases the modularity of the agentic workflow, allowing non-programmers (such as trainers or handlers with domain expertise) to customize the agent's outputs through straightforward edits to a text file. This adaptability further highlights the potential of the workflow to integrate domain-specific knowledge while remaining accessible to a broader audience.

To ensure that the LLM output aligns with predefined guidelines, the agent uses single-shot prompting (19). That is, by including an example output format in the prompt, the agent informs the LLM about the expected structure of the response. This approach maintains flexibility while reducing variability in the generated outputs, helping to ensure that the progression steps are accurately reflected in the final training plan. As a result, a plan generated by this agent can look like this:

1. **33.0 seconds:**
- Start with 33 seconds.
- Repeat with 16 seconds.
- 5 seconds
- 24 seconds
- 45 seconds
- 1 second
- 16 seconds
- 8 seconds

...

2. **45.0 seconds:**
- Start with 45 seconds.
- Repeat with 22 seconds.

...

This design demonstrates how internal information, such as SOPs, can be incorporated into an agent's workflow to improve consistency and adherence to standards. However, this approach remains relatively simple, relying on a static text based configuration file for predefined progressions. Future enhancements could involve more dynamic methods, such as Retrieval-Augmented Generation (RAG) (20) , where the agent retrieves SOPs or other relevant guidelines from a large knowledge base in real time. Such advancements would allow agents to handle a wider range of tasks and adapt to evolving contexts with even greater precision.

3.2 Collaborative Agent Interactions: Building Modular Teams

3.2.1 Conditional Edges – Collecting Information Dynamically

An essential feature of agentic workflows is the ability of agents to make decisions dynamically, based on their current context or available information. This capability is achieved through the concept of conditional edges, which enable agents to decide the next step in their workflow rather than following a predefined sequence of actions. Conditional edges introduce flexibility and autonomy, elevating agents from simple task executors to intelligent decision-makers. The *Outline Writer Agent* (Figure 3) provides an excellent example of this functionality. Its primary task is to generate a plan outline based on the given behavior, current status, and target goal. However, before proceeding, the agent evaluates whether it has sufficient information to complete the task. If additional input is required, the *Outline Writer Agent* determines whether to request it from the *Internet Researcher Agent* or the *Handler Interaction Agent*. This decision depends on the nature of the missing information—whether it can be sourced from online references or requires input directly from the dog handler.

This example highlights the core idea of agency: the ability of agents to make decisions autonomously. By incorporating conditional edges, agents not only perform predefined tasks but also adapt to dynamic conditions, making the workflow more robust and intelligent.

3.2.2 Map Reduce and structured output - Breaking Down Training Steps

While conditional edges enable agents to make decisions dynamically, they are limited when workflows require managing multiple tasks simultaneously or activating an unknown number of additional agents. This limitation is particularly evident in the context of training plans, which often consist of multiple steps that are relatively independent and require their own detailed progression plans. Since the number and nature of these steps are not always known in advance, it becomes necessary to identify and manage them dynamically. This challenge is effectively managed using a Map-Reduce approach—a strategy that divides tasks into smaller, manageable units, processes them in parallel, and efficiently recombines the results (14).

The identification and distribution of tasks are implemented in the *Team Manager Agent* (Figure 4) . Starting with a training plan outline, this agent extracts individual training steps and determines which team of agents is best suited to handle each step. To achieve this, the agent leverages the capabilities of a large language model (LLM) in a novel way: instead of generating plain text outputs, it prompts the LLM to produce structured outputs in JSON format. This structured format explicitly defines each training step, the corresponding team, and additional parameters required to complete the task. Structured outputs are particularly advantageous in complex workflows, as they ensure that accurate and complete information is transmitted to subsequent agents, avoiding ambiguities inherent in natural language responses. This guarantees consistency and reliability when activating multiple teams. Once the training steps and their details are extracted, the *Team Manager Agent* dynamically activates the necessary Specialist-Welfare Teams. Each team processes its assigned task independently and writes its output directly to the shared memory. Importantly, the teams work in parallel, significantly reducing the time required to generate the final training plan while preserving the modularity and scalability of the system. After all teams complete their tasks, their outputs are collected from the shared memory for further integration.

This example demonstrates a higher level of complexity in agentic workflows, showcasing how structured outputs, specialized teams, and parallel processing enhance the system's modularity, flexibility, and scalability. By leveraging Map-Reduce principles, this approach addresses the limitations of conditional

edges, enabling the system to dynamically adapt to diverse and complex scenarios while ensuring high-quality outputs.

3.2.3 Reviewer Agents – Refining Training Plans for Welfare Compliance

The integration of a Reviewer Agent introduces a new dimension to agent interactions by enabling iterative refinement of outputs (13). This is exemplified by the *Welfare Agent* , which collaborates with Specialist Plan Writers, such as the *Duration Specialist Agent* , to ensure that training plans align with predefined welfare principles (Figure 5). This architecture highlights two critical features of agent orchestration: the ability to incorporate expert feedback loops and the potential for iterative cycles within the workflow. The *Welfare Agent* does not modify the training plan directly. Instead, it evaluates the plan against encoded welfare principles and provides a detailed review, including specific suggestions for improvement. For instance, the agent may flag ethical or practical concerns and recommend adjustments to ensure the plan adheres to humane training practices. This review is then handed back to the Specialist Plan Writer, who updates the plan based on the feedback. The revised plan is resubmitted to the *Welfare Agent* for further evaluation, allowing iterative refinement. This process is illustrated through two integration tests. In the first case, the draft plan includes the following step:

> **Part of Draft Plan**
> Use a leash correction to enforce the sit position.
> **Review**
> Feedback: The use of a leash correction is considered an aversive method and is not in line with positive reinforcement training principles. Instead, I recommend using treats or praise to encourage the dog to sit. This approach will promote a more positive learning environment and ensure the dog's well-being. Please revise this step to align with humane training practices.

In the second test case, the draft plan addressed a French Bulldog with known breathing problems:

> **Part of Draft Plan**
> Get the dog motivated by throwing a ball and let him run a lot.
> **Review**
> Feedback: While getting the dog motivated is important, throwing a ball for a French Bulldog, especially one with potential breathing issues, may not be the best approach. Instead, consider using treats or toys that require less physical exertion to avoid putting strain on the dog's respiratory system. Ensure that the motivation method aligns with the dog's health status to prevent any discomfort or health risks.

These cases illustrate the *Welfare Agent*'s ability to identify potential issues in training plans and provide constructive feedback aligned with welfare principles. In both examples, the agent flagged concerns—whether ethical, practical, or health-related—and suggested humane and effective alternatives. This feedback is then handed back to the Specialist Plan Writer, ensuring that the revised plans address the identified issues while adhering to best practices. To prevent infinite loops or excessively long cycles, the workflow includes a safeguard: a maximum iteration limit. This ensures that the process terminates after a defined number of cycles, even if the *Welfare Agent* is not fully satisfied with the plan. The final iteration results are stored in the shared memory for use by other agents or for presentation to the user.

This design demonstrates the flexibility and robustness of agentic workflows by introducing dynamic cycles as a core feature of agent interactions. By separating review and revision tasks into distinct agents,

this architecture promotes modularity, scalability, and clarity—particularly in workflows that involve complex and multidisciplinary requirements.

3.3 Integrating Tools: Expanding Agent Capabilities

3.3.1 Internet Research Agent – Gathering Web-Based Background Knowledge

In the previous sections, agents relied primarily on large language models (LLMs) to generate outputs. However, integrating external tools can expand their functionality and adaptability to meet more diverse requirements. The *Internet Research Agent* demonstrates this integration by leveraging the internet to gather additional, up-to-date, and context-specific information (Figure 3). Rather than simply executing a basic search query, this agent employs a systematic approach to ensure that the retrieved information is both relevant and well-structured. This ensures that the collected data can be efficiently utilized by other agents within the workflow.

The *Internet Research Agent* receives the behavior to train from the *Outline Writer Agent*. Instead of directly searching for it, the agent first leverages an LLM to refine the query according to predefined training styles and generate variations to ensure comprehensive results. Specifically, the agent adapts the query to align with positive reinforcement principles and exclude aversive methods. It also generates multiple variations of the query to explore diverse perspectives and approaches, increasing the likelihood of retrieving relevant and informative results. For each derived question, the agent performs an internet search and retrieves a predefined number of results. It then scrapes these webpages, and adds the content to the shared memory thereby making it accessible to other agents. By using this multi-step process, the *Internet Research Agent* ensures that the retrieved data is both comprehensive and tailored to the needs of the workflow. Importantly, this tool-based approach enables the system to address scenarios where the LLM alone cannot provide sufficient or up-to-date information.

This example demonstrates the versatility of agentic workflows when augmented with external tools. The *Internet Research Agent* effectively bridges the gap between real-time data access and the LLM's generative capabilities, providing other agents with a richer context to enhance their outputs.

3.3.2 Reasoning and Acting – Iterative Goal Refinement with the Handler

The *Handler Interaction Agent* leverages the ReAct (Reasoning and Acting) architecture (21), representing a significant advancement in tool usage. Unlike agents with predefined workflows, this agent dynamically determines the number of interactions required with the handler to achieve its task (Figure 6).

To refine a training plan goal, the *Handler Interaction Agent* begins by analyzing the input memory to evaluate whether the behavior, current status, and goal are sufficiently defined. If the input is incomplete or unclear, the agent formulates targeted questions to request clarification or additional details from the handler. These questions bridge the gap between ambiguous input and the detailed requirements of subsequent agents. Using the handler as a 'tool', the agent iteratively refines its understanding by continuously assessing the completeness of the input and updating the conversation. This reasoning-action loop continues until the necessary details are provided or predefined interaction limits are reached. By dynamically determining the required number of interactions, the agent ensures precise goal definitions, even in scenarios where the handler may initially provide vague or incomplete objectives. A sample interaction is shown in Figure 6.

This iterative approach highlights the importance of dynamic, handler-driven refinement in agentic workflows. By ensuring that ambiguous input is clarified and detailed objectives are established, the *Handler Interaction Agent* plays a critical role in bridging the gap between vague initial input and the

precise specifications required by subsequent agents. This capability not only enhances the accuracy of downstream tasks but also exemplifies the adaptability and autonomy introduced by integrating the ReAct architecture into agentic systems.

3.4 An Integrated Workflow: Generating Comprehensive Training Plans

Building on the individual agents and teams introduced in the previous sections, this section presents how they can be orchestrated into a cohesive workflow capable of generating detailed training plans tailored to specific behaviors and individual dogs (Figure 7). The workflow begins with the *Handler Interaction Agent*, which gathers essential information directly from the handler. This includes defining the target behavior, its current status, and a measurable training goal. Once this information is clarified, the *Behavior Research Team* takes over.

The *Outline Writer Agent* evaluates whether the behavior can be addressed using existing knowledge or if additional information is required. If necessary, it collaborates with the *Internet Research Agent* to gather relevant background knowledge or consults the *Behavior Requests Agent* to obtain further clarification from the handler. This process ensures that the draft outline is both well-informed and contextually relevant. After drafting the outline, the *Dog Details Agent* collects specific information about the individual dog that could influence the training plan. For instance, when creating a plan to extend the duration of a ”sit” behavior, the agent may inquire about potential health concerns, such as hip issues, to ensure the plan is safe and achievable (Figure 8). Next, the *Evaluator* reviews whether the additional information gathered by the *Dog Details Agent* necessitates modifications to the draft outline. If revisions are needed, the *Outline Writer* updates the plan, and this cycle repeats until the draft accurately accounts for the dog’s unique needs and circumstances

Once the outline is finalized, the *Plan Writing Team* begins its work. The *Team Manager Agent* extracts individual training steps from the outline and assigns them to the appropriate *Specialist-Welfare Teams*, such as those focused on distractions, cues, duration, or distance. Each team generates a detailed training plan for its assigned step, incorporating welfare standards and individual dog characteristics. The completed plans are collaboratively stored in shared memory, ensuring accessibility for further refinement or review. The *Final Plan Writer* creates an overarching overview of the training steps, providing clear references to each detailed plan generated by the Specialist-Welfare Teams. Due to the inherent token length limitations of LLM responses, even in models with higher limits, the detailed plans for individual training steps are not merged into a single document. Instead, the overview links to the individual plans, enabling trainers to navigate the information efficiently while maintaining modularity. Although this design may result in some duplication across plans, it prioritizes clarity and accessibility. Future advancements, such as fine-tuning LLMs to handle longer outputs, could address this limitation (22). The *Plan Filer Agent* consolidates the finalized plans into the system’s storage, ensuring they are stored and organized for immediate use. This step guarantees that trainers can easily access the complete set of plans when needed. Examples of the generated plans are provided in the Supplementary Material Sections 1.2 and 1.3.

In summary, this integrated workflow demonstrates the potential of agentic systems to produce detailed, actionable training plans. By leveraging modular agents and collaborative teams, the system ensures that the plans are progression-based and tailored to the specific needs of each dog. Furthermore, the workflow showcases the adaptability and scalability of agentic systems in addressing complex, real-world challenges, emphasizing their relevance in fields requiring precision and customization.

4 DISCUSSION

This study demonstrates the feasibility of employing an agentic workflow to generate individualized and actionable training plans for dogs. It showed how modular agents can collaborate to address the complex task of creating detailed training steps while incorporating specific considerations, such as welfare compliance and health concerns. However, while this workflow offers a proof-of-concept, it is not intended as a one-size-fits-all solution. The diversity of tasks performed by dogs and the wide range of training methodologies used in the community make such a generalized solution impractical. Instead, this work highlights how the building blocks of an agentic workflow can be adapted to meet the specific needs of different tasks and training philosophies.

A major advantage of the presented approach lies in its modularity. By assigning distinct tasks to specialized agents, the workflow is both adaptable and efficient. Furthermore, each agent can operate independently using smaller LLMs, such as GPT-4o-mini[2], which reduces operational costs compared to larger models. Importantly, this opens the door for workflows to utilize open-source LLMs, enabling deployment on local devices. Such an approach not only enhances data privacy but also allows users to operate the workflow without reliance on third-party providers, a significant consideration in sensitive applications. This modular design is further supported by a robust testing framework, which allows each agent and team to be tested individually and at different levels. These automated tests ensure consistent functionality even after modifications, such as replacing an LLM implementation or adapting an agent for a new task. By combining modularity with rigorous testing, the workflow maintains its reliability while remaining flexible and adaptable to future advancements in AI and machine learning.

One of the key strengths of the presented agentic workflow is its ability to generate actionable training plans with detailed progressions, outlining clearly defined steps for each stage of training. For instance, the *Duration Specialist Agent* does more than set the overarching goal of extending a behavior's duration—it provides a step-by-step progression plan, specifying each increment and the precise actions the handler should take to achieve success. This level of detail ensures that the plans are immediately usable by trainers, reducing ambiguity and eliminating the need for extensive interpretation. This capability addresses a notable limitation of general-purpose large language models (LLMs), which often produce generic, unstructured outputs that require significant adaptation to be practically useful. By leveraging modular agents designed for specific tasks, the workflow overcomes this gap, delivering training plans that are both precise and practical. This precision is especially critical in dog training, where clear, actionable steps are essential to achieving consistency and success.

Despite its strengths, the workflow also has limitations. One notable challenge arises from the tendency of LLMs to limit the length of their generated outputs, even when their theoretical token capacity is far greater. This affects the workflow when an agent is tasked with merging training plans from multiple specialist agents into a single, unified document. In such cases, the agent may inadvertently shorten or edit the detailed plans, potentially omitting important details or introducing inconsistencies. While fine-tuning an LLM to generate longer outputs could mitigate this issue (22), doing so would reduce the adaptability of the workflow by tying it to a specific model. As a result, the current implementation generates multiple separate files for the training plans, which, while ensuring completeness, can sometimes lead to overlapping or redundant training steps. Another limitation lies in the accessibility of the current implementation. The workflow is primarily Python-based and designed for developers, as this approach allows for rapid prototyping, flexibility, and a focus on validating the core concept. However, for broader

[2] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

practical application by dog handlers and trainers, a more user-friendly interface, such as a dedicated app or web-based platform, would enhance usability. While this was beyond the scope of the current work, future development efforts could focus on creating such interfaces to bridge the gap between technical implementation and real-world application.

A broader and more fundamental challenge of the presented approach lies in evaluating the quality of the generated plans themselves. At present, a formal evaluation is not feasible due to the absence of a standardized 'ground truth,' such as a database of universally accepted training plans. One potential evaluation approach could involve trainers reviewing and rating the generated plans; however, designing such a study as a double-blind experiment would be complex and resource-intensive. Additionally, the diversity of training philosophies within the community means that evaluation outcomes could be influenced by the individual perspectives of selected trainers. The most rigorous evaluation would involve training dogs using the AI-generated plans and comparing their outcomes to those trained with plans created by experienced humans. However, such an approach would require significant time and resources, which fall outside the scope of this manuscript. This gap highlights the need for broader discussions within the training community. While this manuscript does not propose a specific solution, fostering collaboration among trainers, researchers, and organizations could help establish benchmarks or shared evaluation criteria that respect the diversity of training philosophies and intellectual property. Thus, the presented workflow should be viewed as an initial prototype, demonstrating the potential of agentic workflows in this domain. Future studies should prioritize formal evaluations, comparing AI-generated plans with those of experienced trainers in controlled experiments. Such efforts would not only validate the quality of the plans but also provide valuable insights for further refinement and development.

Looking ahead, several promising directions could enhance the presented approach and its applicability. One key avenue is the integration of prior training data, which could significantly refine the workflow's ability to generate tailored plans. For example, incorporating a long-term memory component would allow the system to store and utilize handler-specific information, such as a dog's health history, behavioral tendencies, or previous training progress. By leveraging this contextual data, agents could produce increasingly personalized and precise training plans, adapting dynamically to the unique needs of individual dogs and their handlers. Additionally, advanced techniques such as RAG present an exciting opportunity to enrich the workflow further

A fundamental aspect of this workflow is its role as a collaborative tool designed to assist, not replace, dog handlers. Just as AI-assisted tools in other fields, such as code generation or medical diagnostics, enhance human expertise rather than substitute for it, the outputs generated by this workflow are intended to support trainers. These plans should not be applied blindly; instead, they serve as a foundation that handlers can critically assess, adapt, and refine to suit their specific goals, training philosophies, and the unique needs of each dog. Beyond this, the workflow aims to free trainers from time-consuming administrative tasks, such as drafting detailed training plans from scratch. By accelerating the plan-writing process, the approach allows trainers to devote more time to what truly matters—interacting with and training their dogs. By placing the handler's expertise at the center of the process, the workflow reinforces the importance of professional judgment and experience. Its purpose is not to dictate how training should be conducted but to provide a flexible and innovative resource that empowers trainers to work more efficiently and effectively. In this way, the workflow complements the invaluable skills and insights of trainers, fostering a partnership between human expertise and AI-driven innovation.

To conclude, the ultimate vision for this work is the development of an integrated, AI-enhanced training ecosystem. In this envisioned system, training plans would be dynamically generated prior to each session,

tailored to the dog's current behavioral status. During training, real-time data on the dog's progress and challenges could be used to iteratively update these plans, ensuring they remain responsive and effective. After each session, updated status information and identified challenges would be automatically incorporated into a centralized system, enabling continuous improvement over time. By combining data-driven refinements with real-time adjustments, this ecosystem has the potential to bridge the gap between cutting-edge technology and practical, humane training methodologies. While ambitious, this vision highlights the transformative possibilities of integrating AI with the art and science of dog training.

AUTHOR CONTRIBUTIONS

JS conceived the study, developed the methodology, implemented the software, conducted the analyses, and wrote the manuscript. JS is solely responsible for the content of this work and agrees to be accountable for all aspects of the manuscript.

CONFLICT OF INTEREST STATEMENT

JS is the founder of Tier Wohl Team GbR, which developed the software described in this manuscript. The software is made publicly available under an MIT license, and the research was conducted in the absence of any external funding or financial relationships that could be construed as a potential conflict of interest.

REFERENCES

1 .Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. *Advances in Neural Information Processing Systems* (Curran Associates, Inc.) (2017), vol. 30.

2 .Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* (Curran Associates, Inc.) (2020), vol. 33, 1877–1901.

3 .Ramesh A, Pavlov M, Goh G, Gray S, Voss C, Radford A, et al. Zero-Shot Text-to-Image Generation. *Proceedings of the 38th International Conference on Machine Learning* (PMLR) (2021), 8821–8831.

4 .[Dataset] Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/ (????).

5 .[Dataset] OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, et al. GPT-4 Technical Report (2024). doi:10.48550/arXiv.2303.08774.

6 .Shin D, Hsieh G, Kim YH. PlanFitting: Tailoring Personalized Exercise Plans with Large Language Models (2023). doi:10.48550/arXiv.2309.12555.

7 .Qian C, Xie Z, Wang Y, Liu W, Dang Y, Du Z, et al. Scaling Large-Language-Model-based Multi-Agent Collaboration (2024). doi:10.48550/arXiv.2406.07155.

8 .Qian C, Liu W, Liu H, Chen N, Dang Y, Li J, et al. ChatDev: Communicative Agents for Software Development (2024). doi:10.48550/arXiv.2307.07924.

9 .Park JS, O'Brien J, Cai CJ, Morris MR, Liang P, Bernstein MS. Generative Agents: Interactive Simulacra of Human Behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA: Association for Computing Machinery) (2023), UIST '23, 1–22. doi:10.1145/3586183.3606763.

10 .Li J, Wang S, Zhang M, Li W, Lai Y, Kang X, et al. Agent Hospital: A Simulacrum of Hospital with Evolvable Medical Agents (2024). doi:10.48550/arXiv.2405.02957.

11 .Schick T, Dwivedi-Yu J, Dessi R, Raileanu R, Lomeli M, Hambro E, et al. Toolformer: Language Models Can Teach Themselves to Use Tools. *Thirty-Seventh Conference on Neural Information Processing Systems* (2023).

12 .Chen W, Su Y, Zuo J, Yang C, Yuan C, Chan CM, et al. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. *The Twelfth International Conference on Learning Representations* (2023).

13 .Welleck S, Lu X, West P, Brahman F, Shen T, Khashabi D, et al. Generating Sequences by Learning to Self-Correct. *The Eleventh International Conference on Learning Representations* (2022).

14 .Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. *Commun. ACM* **51** (2008) 107–113. doi:10.1145/1327452.1327492.

15 .Sumers T, Yao S, Narasimhan K, Griffiths T. Cognitive architectures for language agents. *Transactions on Machine Learning Research* (2024).

16 .Beck K. *Test Driven Development: By Example* (Boston: Addison-Wesley Professional), 1 edn. (2002).

17 .Nagappan N, Maximilien EM, Bhat T, Williams L. Realizing quality improvement through test driven development: Results and experiences of four industrial teams. *Empirical Softw. Engg.* **13** (2008) 289–302. doi:10.1007/s10664-008-9062-z.

18 .Spector M. *Clicker Training for Obedience: Shaping Top Performance-Positively* (Sunshine Books, Inc.), illustrated edition edn. (2006).

19 .Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* **55** (2023) 195:1–195:35. doi:10.1145/3560815.

20 .Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA: Curran Associates Inc.) (2020), NIPS '20, 9459–9474.

21 .Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, et al. ReAct: Synergizing Reasoning and Acting in Language Models (2023). doi:10.48550/arXiv.2210.03629.

22 .Wang L, Yang N, Zhang X, Huang X, Wei F. Bootstrap Your Own Context Length (2024). doi:10.48550/arXiv.2412.18860.

5 TABLES

Framework	Summary	Special Features
AutoGen[1]	A Microsoft framework for building collaborative multi-agent systems with shared memory and task execution.	Offers strong integration with memory systems and facilitates natural agent collaboration.
CrewAI[2]	A framework specializing in team-based agent workflows, where agents take on defined roles and responsibilities.	Focuses on role-based collaboration, making it suitable for task-specific teamwork and simulations.
LangGraph[3]	A graph-based agent orchestration framework designed for controlled, iterative workflows and safe AI interactions.	Modular design allows flexible and programmable workflows, ideal for research and safety-critical domains.
OpenAI Swarm[4]	A framework enabling the deployment of multiple cooperative AI agents using OpenAI's API. Agents interact to solve complex tasks collaboratively.	Emphasizes scalability and efficient task distribution in multi-agent setups.

Table 1. Comparison of AI Frameworks

[1] https://github.com/microsoft/autogen

[2] https://github.com/crewAIInc/crewAI

[3] https://github.com/langchainai/langchain

[4] https://github.com/openai/swarm

FIGURE CAPTIONS

```python
class SpecificAgent(BaseAgent):
    NAME = "ExampleAgent"
    LLM = "gpt-4o-mini"  # Choose best fitting llm

    @staticmethod
    def action(state):
        # Step 1: Retrieve information from the memory (state)
        input_data = state.get("key_in_memory")

        # Step 2: Construct the LLM prompt
        background = "You are a helpful agent tasked with..."
        task_prompt = f"Given {input}, generate a plan..."

        # Step 3: prepare prompt messages
        messages = [
            SystemMessage(content=background),
            HumanMessage(content=task_prompt.format(
                input=input_data))
                ]

        # Step 4: Call the LLM
        response = LLM.invoke(messages)

        # Step 5: Update the state with the result
        return {"key_to_store_result": response.content}
```

Figure 1. Template for a typical agent implementation - The example demonstrates the workflow for retrieving input from memory, constructing an LLM prompt, invoking the LLM, and updating the agent's state with the result.

Background Story
You are an experienced dog trainer having trained dogs to perform behaviours even under incredible distractions. In addition, you have a knack for helping novice trainers to teach their own dogs to cope with extreme distractions. The novice trainers love to work with you, as you give them extremely detailed instructions on how to start with mild distractions and build up from there in minuscule steps which enable the dog to master each training step successfully. You give the novice trainers a clear progression plan detailing out which distractions they should train, what the difficulty in this distraction is, when exactly they should reinforce the dog. In addition, each of your steps holds the information how the novice trainer should react in case the distraction is too strong and the dog breaks the behaviour.

Task Prompt
Please write a training plan to assure that the behaviour {behavior} can be performed under distractions. The dog already knows the behaviour and can perform it. Only look at the distractions and leave other elements to the other members of your team.

CURRENT STATUS
{status}

GOAL
{goal}

INFORMATION ABOUT THE DOG:
{dog_details}

In your training plan, start by stating the current status and then the goal. Next, develop a detailed progression plan for the trainer. In your progression plan, start from easy to hard. Clearly state the distraction and explain what the challenge is in this setup. When choosing distractions, take into account the information about the dog (if they are given). Also, add the information on how the trainer should react in case the distraction is too strong and the dog breaks the behaviour.

Figure 2. Example of a prompt for an agent – This specific prompt is designed for the *Distraction Specialist Agent* and guides it in generating a progression plan to ensure reliable performance of a trained behavior under distractions. It includes the behavior, current status, goal, and relevant dog details, ensuring the generated plan is detailed, step-by-step, and tailored to the dog's needs and the trainer's actions.

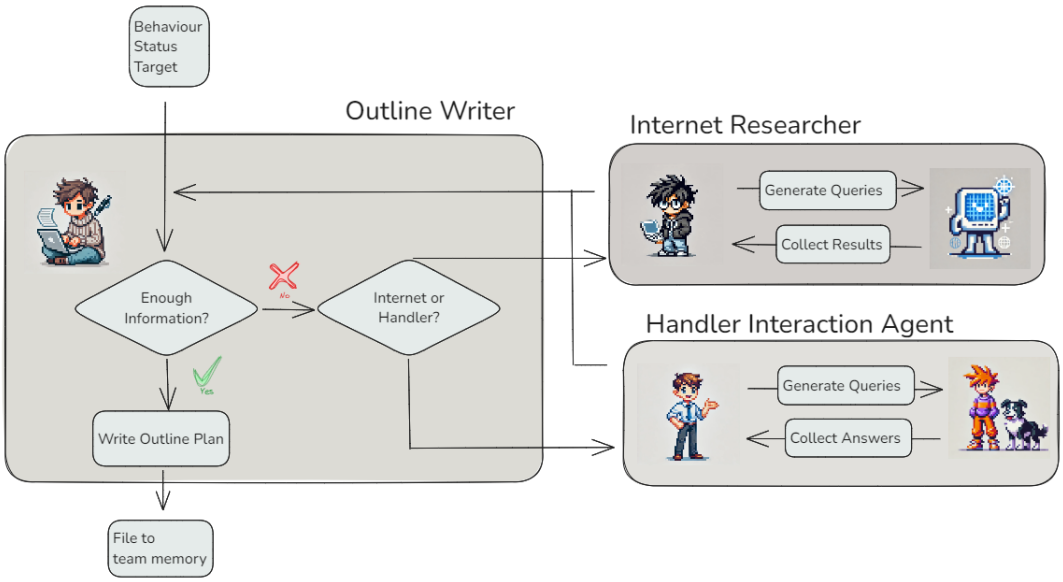

Figure 3. Conditional Edges in Agent Collaboration – When insufficient information is available, the *Outline Writer Agent* dynamically determines whether to consult the *Internet Researcher Agent* or the *Handler Interaction Agent* , based on the nature of the missing data. Each supporting agent performs its specialized task, returning results to the *Outline Writer Agent* for generating a complete outline plan.

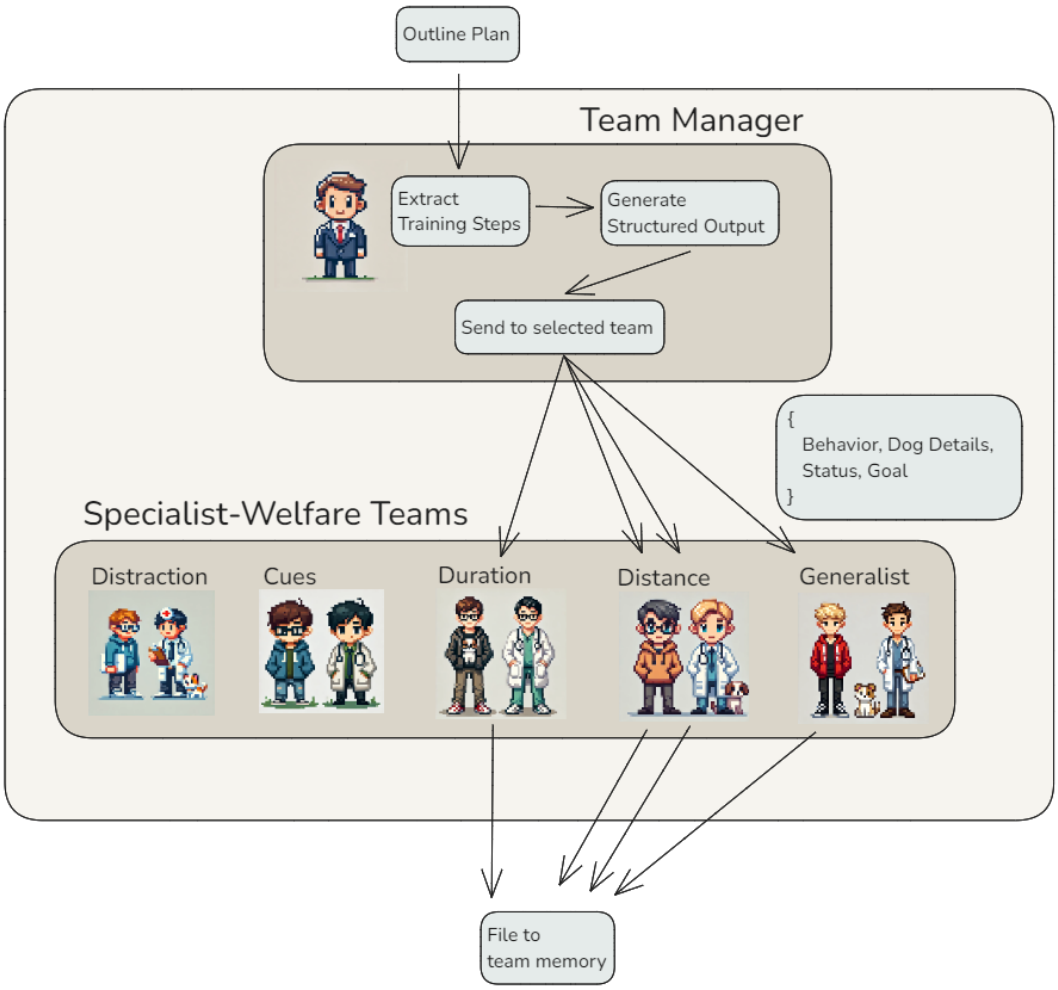

Figure 4. Map-Reduce Workflow with Structured Output – The *Team Manager Agent* orchestrates a Map-Reduce pattern for training plan generation. Individual training steps are extracted from the outline plan, structured as JSON outputs, and distributed to the Specialist-Welfare Teams based on the specific requirements of each step. Each team processes its task independently and stores the results in shared memory, ensuring modularity, scalability, and consistency in the workflow.

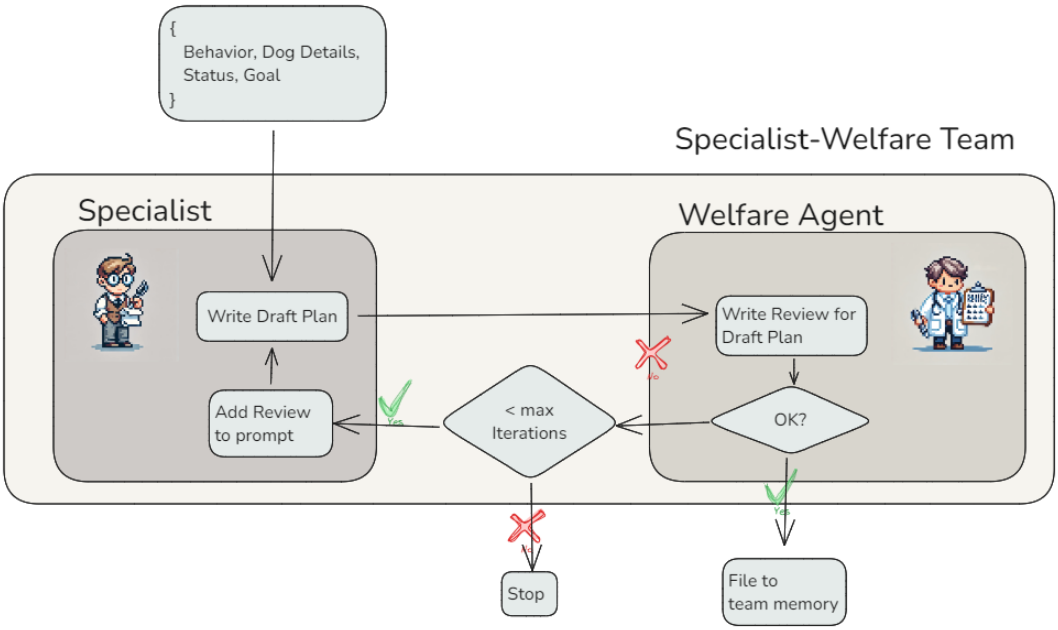

Figure 5. Refining LLM Results with Review Agents – Review agents, such as the *Welfare Agent*, improve the quality of outputs through iterative evaluation and feedback. In this workflow, the Specialist Agent drafts a training plan, which the *Welfare Agent* reviews for compliance with welfare standards. Feedback is incorporated iteratively, with a maximum iteration limit to prevent infinite loops.

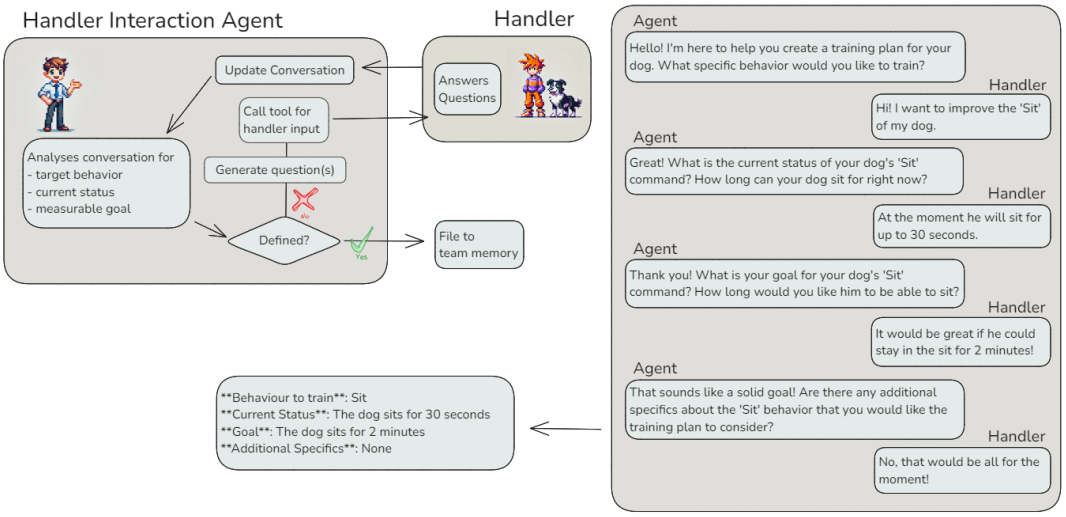

Figure 6. Reasoning and Acting (ReAct) with Tool Usage – The *Handler Interaction Agent* exemplifies the ReAct architecture by iteratively refining its understanding of a training plan goal through interaction with a human handler, treated as a ”tool”. The left panel illustrates the agent’s workflow, dynamically generating questions and updating the conversation based on handler input. The right panel shows an example interaction between the agent and the handler, culminating in a well-defined training plan goal.

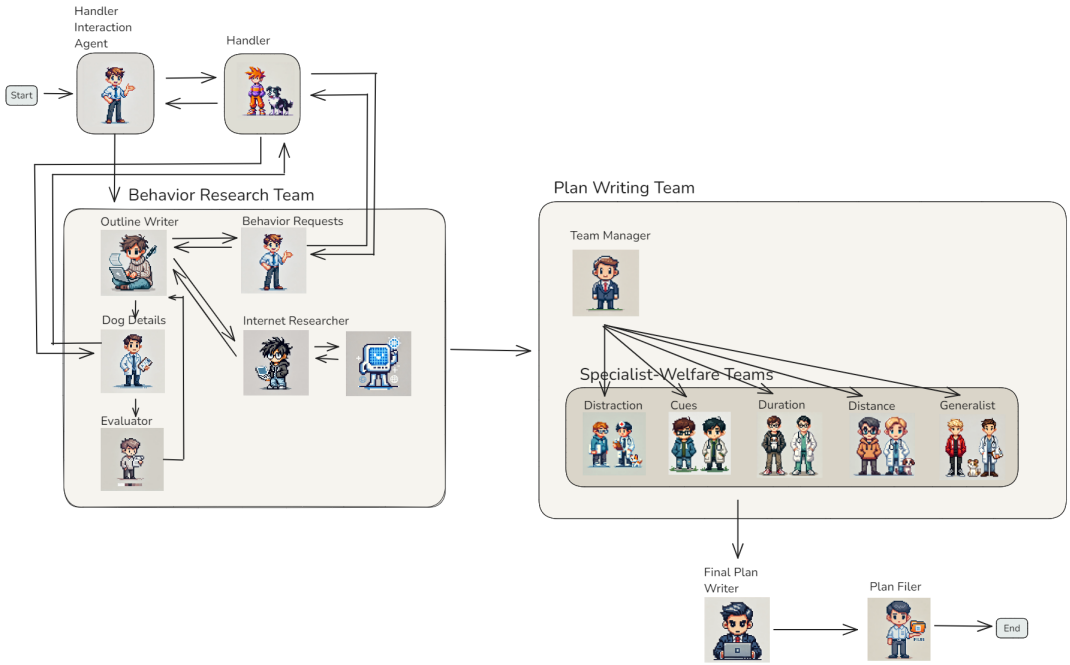

Figure 7. Complete Workflow of the Training Plan Team – This sample workflow integrates all agents and teams into a comprehensive system for generating detailed training plans. The process begins with the *Handler Interaction Agent*, which collects input from the handler to define the behavior, current status , and goal. This information flows into the *Behavior Research Team*, where agents such as the *Outline Writer*, *Internet Researcher*, and *Evaluator* collaborate to refine the outline plan. The refined outline is passed to the *Plan Writing Team*, led by the *Team Manager*, who distributes tasks to Specialist-Welfare Teams based on the specific requirements of each training step. The workflow concludes with the *Final Plan Writer* and *Plan Filer*, who consolidate and store the completed training plans, ensuring modularity, scalability, and precision in the system.

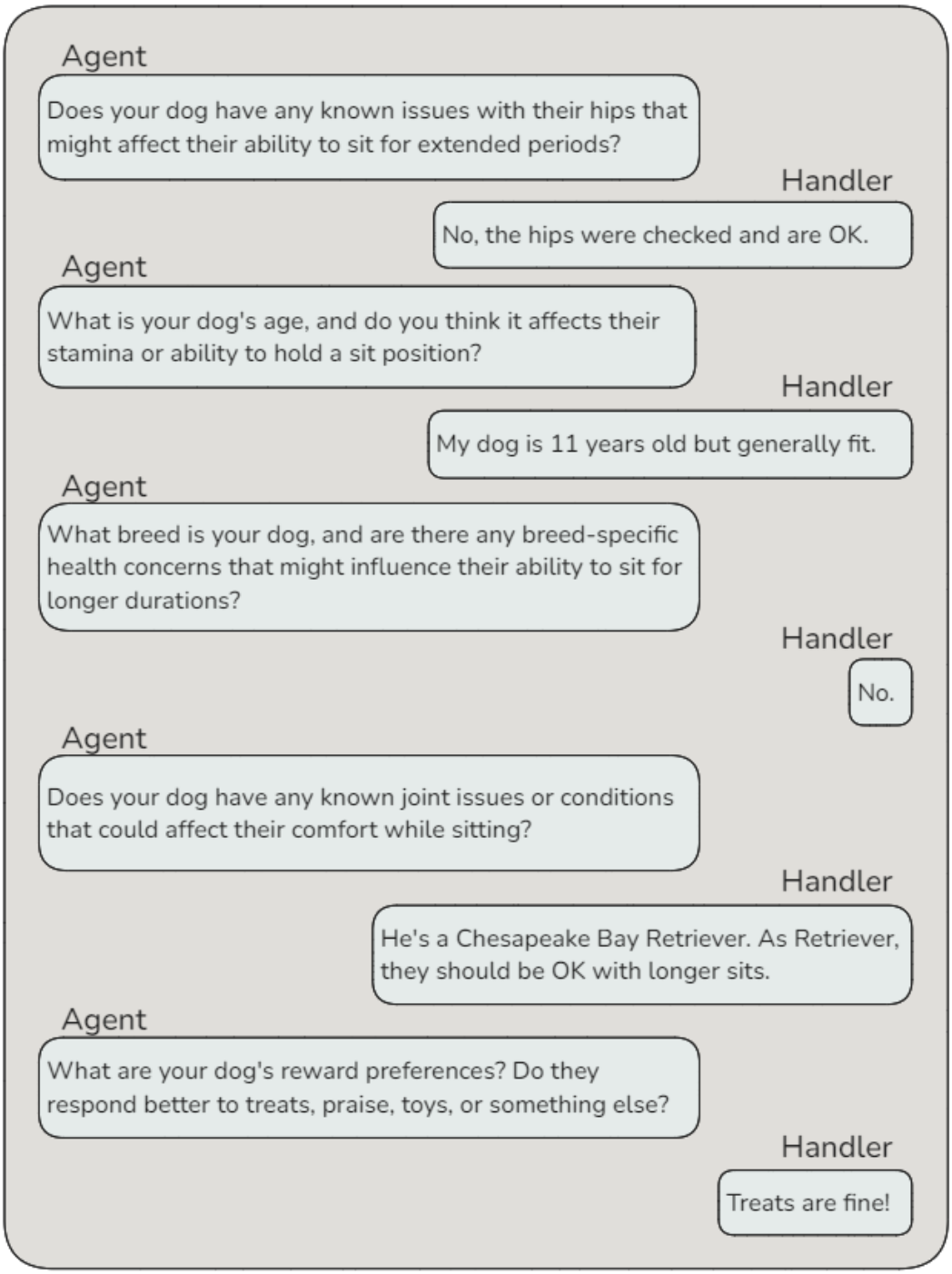

Figure 8. Agent-Handler Interaction Example – This dialogue demonstrates the interaction between the *Dog Details Agent* and the handler. The agent dynamically generates specific, goal behavior relevant questions to gather information about the dog's health, breed-specific concerns, and reward preferences. These details are essential for tailoring the training plan to the individual dog's needs while ensuring safety and effectiveness.