R6D9: Your Agentic Copilot, from Virtual to Real

1. Overview

1.1 Mission

Current computer operations still rely heavily on manual interaction. While automation tools such as RPA and macro recording exist, they often have **high learning costs, limited generalization, and low adaptability**.

This project aims to build a **universal Computer Control Agent**, allowing users to complete **complex computer operations** simply by **describing tasks in natural language**. This will lower the barrier to automation and improve work efficiency.

The project will be developed in two phases at this time point:

- **Browser Agent**: Automates web interactions, including form filling, data extraction, auto-login, and data analysis.
- **Computer Agent**: Expands to desktop and mobile applications, enabling file management, software operation, and system task automation.

Ultimately, anyone, human and AI agent, will be able to control their computer with natural language, achieving efficient task automation without requiring programming knowledge.

1.2 Challanges

Despite advancements in AI automation, we still face the following key challenges:

1. **Task Understanding & Decomposition**
    - How can AI fully comprehend user input in natural language and break it down into executable steps?
    - Does the task require coordination across different environments (browser/desktop/mobile)?
2. **Reliability of Execution**
    - How can we ensure automation accuracy and minimize errors?
    - How can AI maintain stability despite UI changes, website updates, or software upgrades?
3. **Security & Permission Management**
    - How can we prevent AI from triggering unintended or malicious actions?

- How can we protect user privacy and prevent sensitive data leaks?

4. **Environmental Adaptability** 🌍

   - How can AI adapt to different software, web architectures, and operating systems?

   - How can it reliably simulate human actions in the absence of APIs?

This project will integrate **AI task planning, multimodal perception, and intelligent execution optimization** to gradually overcome these challenges and build an **efficient, stable, and secure Computer Control Agent**.

2. Technical Architecture

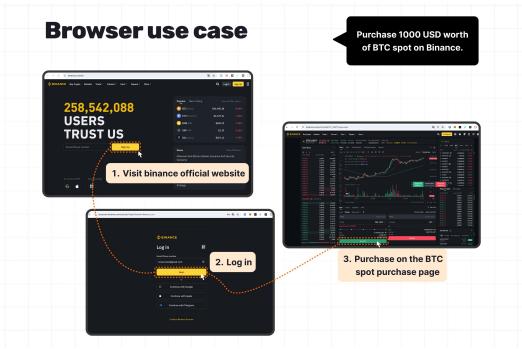

Figure 2.1 Browser Use (mobile app) use case

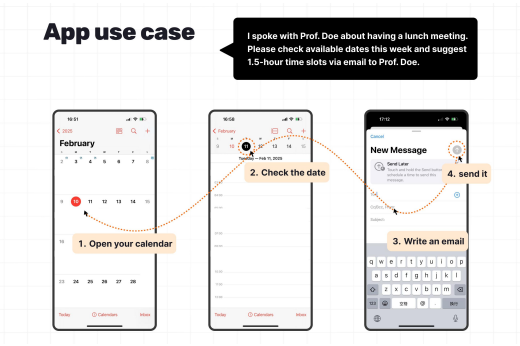

Figure 2.2 Computer Use (mobile app) use case

Figures 2.1 and 2.2 illustrate examples of agent execution in a browser and a mobile app, respectively. Although these application scenarios interact with entirely different environments, they can be abstracted into a unified Computer Control Agent model, as shown in Figure 2.3. Given a user input command i, the agent must interact with the environment to complete the corresponding task. This process can be broken down into the following stages:

- **Environmental Perception**: At any time t, the environment is in a state $s_t \in S$, but the agent cannot fully observe s_t. Instead, it receives partial observations $o_t \in O$((where the observation space $O \subset S$). For example, $o_t$could be a screenshot of the current browser window, while s_t includes the full state of the running browser.

- **Agent Input Preparation**: The observation o_t is often preprocessed before being fed into the agent, transforming it into $\mathrm{o\_t \backslash to\ o\_t\^{}*}$ This may involve cropping a screenshot or extracting a DOM tree from HTML.

- **Agent Action Planning**: Based on the processed input $o_t^*$and user command i, the agent computes the next action a_t^* using a policy π. The policy π is crucial for task accuracy and serves as the core component for algorithm optimization.

- **Action Grounding**: The abstract action $a_t^* \to a_t$ is converted into an executable command ata_t. For example, a high-level instruction like “click the submit button” is translated into a concrete command such as "click(x, y)", where (x,y)(x, y) represents the button’s coordinates.

This process repeats in a loop until a termination condition is met, such as completing the instruction or reaching the maximum step limit.

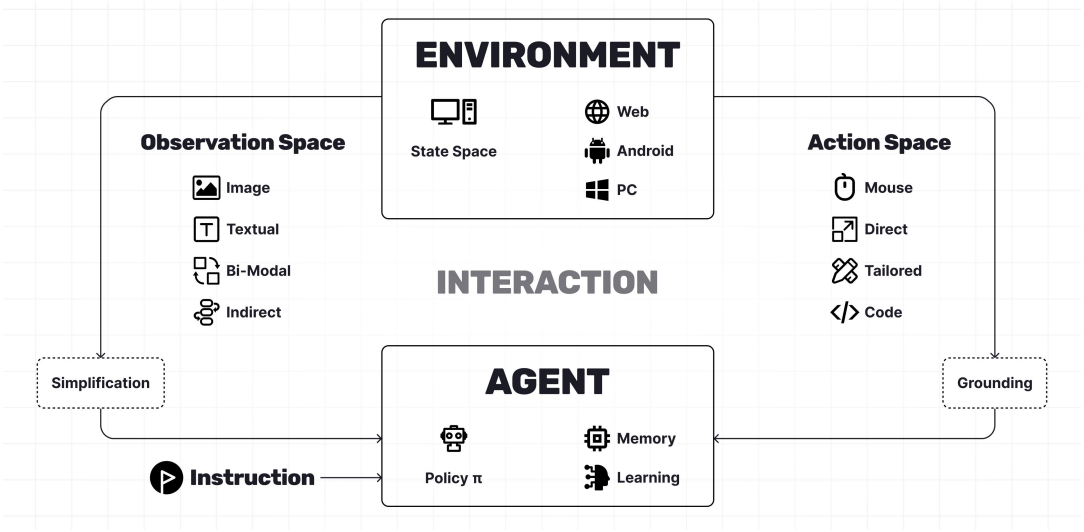

Figure 2.3 Univesal Computer Control Agent

To ensure the smooth operation of this unified Computer Control Agent model, a well-designed technical architecture is essential. Figure 2.4 illustrates the proposed architecture, which includes the following key components:

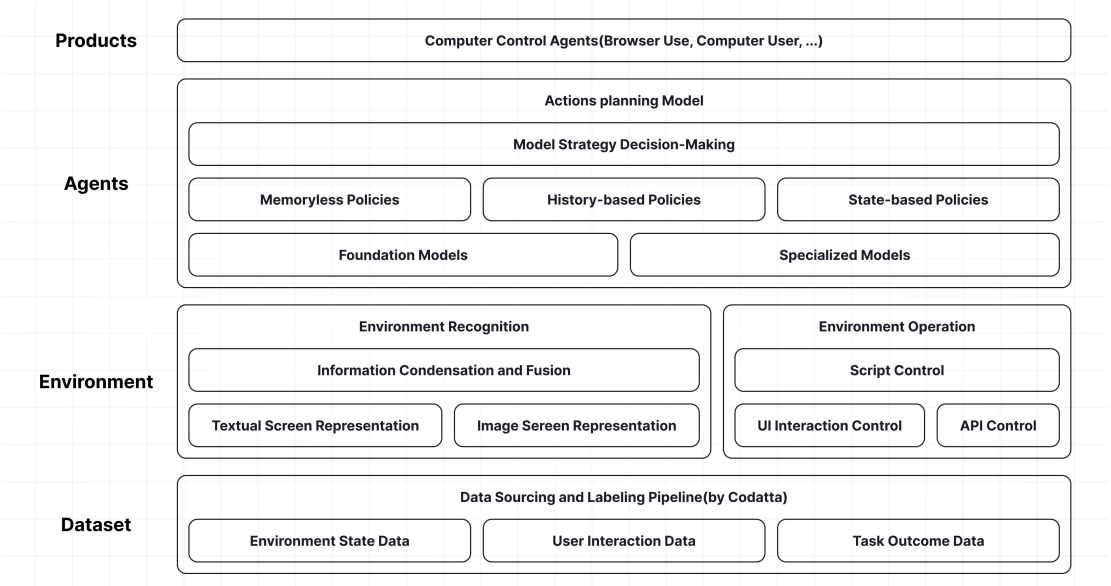

Figure 2.4: Technical Architecture of a Consistently Designed Computer Control Agent

- **Environment Recognition and Operation:** The primary task of the Agent is to

perceive and understand the operational environment to enable effective interaction. This process includes **environment observation** and **interaction execution**.

- **Environment Observation**: In complex computational environments, the Agent must analyze different operating contexts, such as web environments, desktop applications, or mobile applications. Since each environment has unique data structures and interaction patterns, the Agent employs a multi-layered perception strategy:
  - In web environments, the Agent directly parses the DOM structure to extract interactive elements such as buttons, input fields, and links.
  - In desktop/mobile environments, the Agent relies on UI component recognition, OCR analysis, and window management to extract key information from the active interface.

- **Interaction Execution**: After understanding the environment, the Agent must execute appropriate interactions to complete tasks. The primary interaction methods include:
  - API-based precise control: When an environment provides standardized APIs (e.g., DevTools Protocol, Windows UI Automation), the Agent can send direct commands for precise execution.
  - UI-based intelligent control: In cases where direct API access is unavailable, the Agent leverages computer vision, event listeners, and automation techniques to simulate interactions such as mouse clicks and keyboard inputs.

The accuracy of environment recognition and operation directly determines whether the Agent can perform tasks stably and efficiently. Therefore, system architecture must ensure adaptive environment modeling, enabling the Agent to adjust its perception methods and interaction strategies based on different application scenarios.

- **Data Sourcing and Labeling Pipeline:** A highly efficient data sourcing and labeling pipeline is crucial for enhancing the Agent's ability to understand and execute tasks. The dataset not only consists of static task descriptions but also includes real-world interaction records, which support continuous model optimization. To ensure strong generalization capabilities, the Agent requires diverse data sources:
  - User Interaction Data: Logs of real user actions such as clicks, inputs, and scrolling in browsers or applications, providing executable task execution paths for learning.
  - Environment State Data: Stores UI structures, web DOM trees, and window layouts across different task scenarios, helping the Agent build task environment models.
  - Task Outcome Data: Captures key performance metrics, such as task completion accuracy and execution time, enabling continuous optimization of task planning strategies.

- **Actions Planning Model:** The Actions Planning Model is the core of task execution. It translates natural language instructions into executable action sequences while optimizing execution paths.
    - Task Parsing and Planning: Upon receiving a command, the Agent must decompose it into subtasks and determine the optimal execution strategy. This process includes:
        - Instruction Parsing: Extracting key tasks from user input and mapping them to executable actions. For instance, the command “Download a research paper and save it to Google Drive” can be broken down into: 1) Access the research paper website. 2) Identify and click the download button. 3) Open Google Drive and upload the file.
        - Hierarchical Task Planning: Implementing a layered planning strategy to break down complex tasks into smaller, executable subtasks, ensuring an optimized execution sequence.
    - Execution Strategy Optimization: During execution, the Agent must dynamically adjust its strategy based on the current environment to improve task success rates and efficiency. This includes:
        - Task Execution Path Optimization: Leveraging historical task data to predict the most efficient execution path, minimizing redundant operations.
        - Adaptive Error Recovery: When interactions fail or environments change, the Agent must automatically adjust execution methods, such as switching strategies or replanning the task sequence.

A highly efficient Actions Planning Model ensures that the Agent completes user tasks with minimal steps while maintaining robustness and adaptability in dynamic environments.

3. How to Improve AI Agentic System

3.1 Environment Recognition

In the execution of a Computer Control Agent (CCA), Environment Perception is one of its core capabilities, directly determining the agent’s understanding and adaptability to the computer operating environment. To achieve efficient and precise environment perception, we adopt a Multi-Modal Representation approach, gathering information from different Observation Spaces and leveraging Fusion Techniques to enhance the agent’s comprehension of its surroundings.

- **Image Screen Representation**: Captures the computing environment through screenshots, making it applicable to various platforms such as web, Android, and PC, closely resembling human visual perception. Its strength lies in scalability, but it comes with higher processing complexity. Optimization focuses on identifying key visual elements and simplifying image data.

- **Textual Screen Representation**: Extracts structured text data to analyze UI and environment information. It offers structured, high-precision parsing but has lower scalability. Text optimization includes key element extraction and summarization.
- **Multi-Modal Representation**: Combines image and text inputs for environmental perception, with potential future expansion to video and audio modalities. This approach integrates the strengths of both image and text while enhancing scalability to adapt to a broader range of operational environments.

However, **multi-modal perception** introduces additional technical challenges:

1. **Information Overload**: Multiple data sources may introduce irrelevant information that disrupts decision-making.
2. **Modality Alignment**: Integrating text and image data remains an open research problem.
3. **Computational Cost**: Multi-modal models have high inference costs, requiring optimization strategies.

These are key challenges we are actively researching and addressing.

3.2 Dataset

3.2.1 Data Collection Definition

To train and optimize the Computer Control Agent, the following data needs to be collected:

- **User Interaction Data**: Records user actions in browsers or applications, such as clicks, text inputs, and scrolling. For example, when a user fills out a form on a webpage, the sequence of text entries and the action of clicking the "Submit" button are recorded.
- **Environment State Data**: Includes screenshots, UI structures, webpage DOM trees, and window layouts in various task scenarios. For instance, in a browser environment, the structure and positions of elements (buttons, text fields, links, etc.) are recorded; in desktop and mobile environments, the layout of windows and controls is captured.
- **Task Outcome Data**: Measures key performance indicators such as task completion accuracy and execution time across different strategies. For example, in an automated login task, it tracks whether the login was successful, how long it took, and any encountered errors.

Dataset Comparison & Structure Design

We analyzed compute-use datasets, including Mind2Web, AITW, WebShop, and Rico. Based on this comparative study, we designed a unified dataset structure that is both adaptable to different contexts and provides detailed actionable data. This structure is well-suited for PC, Web, and mobile environments.

1. **Task and Subtask Representation**
    - The definition of tasks and subtasks (episodes) is platform-agnostic.
    - Whether interacting with websites, desktop applications, or mobile apps, the decomposition of tasks into actionable steps remains applicable.
2. **Element Identifiers**
    - The object, object_struct_path, and object_struct_type fields store platform-specific identifiers.
    - For example, in mobile apps, an element's ID could be a resource identifier, whereas, in desktop applications, it could be a window handle or GUI control name.
    - Similarly, object_struct_type adapts to different structures like DOM for Web, WinAppDriver for PC, or UIAutomator for mobile.
3. **Actions and Behaviors**
    - The action and behavior fields describe operations at a high-level, task-oriented abstraction.
    - This abstraction enables mapping the same task (e.g., opening the homepage, searching for an item) to platform-specific implementations.
    - For example, clicking a button on a mobile app may involve a touch event, while on a desktop app, it could require a mouse click or keyboard shortcut.
4. **Environment Understanding**
    - The environment_understanding field captures context-aware data across different platforms.
    - On mobile devices, VLMs (Vision-Language Models) may detect screen elements differently than on desktops, but the overall structure for capturing and integrating this information (parsers, VLMs, LLMs, fusion mechanisms) remains consistent.
5. **State & Completion**
    - The State field represents the overall task completion status and reasons, serving as a platform-independent progress indicator.

Based on these considerations, the labeled dataset fields are as follows:

```JSON
{
  "task": "Purchase the cheapest 50-inch TV from Amazon",
  "episodes": [
    {
```

```
      "episode": 1,
      "subtask": "Open the Amazon website",
      "action": "Open the Amazon homepage",
      "object": "Amazon homepage",
      "object_struct_path": "#homepage",
      "object_struct_type": "DOM",
      "object_struct_content":
"local_path/struct_content/episode1_amazon_home_struct.json",
      "behavior": "Enter the Amazon website URL in the browser and
visit",
      "status": "completed",
      "execution_time": "2024-12-01T10:00:00",
      "current_time": "2024-12-01T10:00:00",
      "screenshot":
"local_path/screenshots/episode1_amazon_home.png",
      "environment_understanding": {
        "parser": {
          "object_hierarchy": [...]
        },
        "vlm": {
          "object_detection": [...]
        },
        "llm": {
          "semantic_analysis": {...}
        },
        "merged": {
          "comprehensive_structure": {
            "dom_elements": {
              "navigation_bar": "Top section",
              "search_bar": "Center top",
              "logo": "Top-left"
            },
            "text_semantics": {
              "key_text": "Sign in, Cart, Search"
            },
            "visual_objects": {
              "logo_detected": true,
              "menu_items": ["Electronics", "Computers", "TVs"]
            }
          }
        }
      },
      "state": {
        "overall_task_completion": false,
```

```
        "reason": "..."
      }
    },
    ...
}
```

3.2.2 Relying on Codatta for High-Quality Data

R6D9's Computer Control Framework Relies on High-Quality Data

The **R6D9** computer control framework requires high-quality data, but such data is not readily available. Instead, it must go through precise collection, professional annotation, and rigorous validation to form a truly valuable knowledge layer. Traditional data collection methods fail to meet these requirements due to several challenges:

- **Lack of suitable data contributors** – High-quality data requires knowledgeable contributors, but few individuals in the market are willing to participate.
- **Unclear data ownership and revenue distribution** – Contributors struggle to benefit from the long-term value of their data, leading to a lack of motivation.
- **Poor data liquidity** – Even when high-value data is generated, efficient circulation mechanisms are lacking, making it difficult to maximize commercial value.

To continuously optimize R6D9, a stable knowledge production and assetification mechanism is needed. To achieve this, we have partnered with Codatta, a leading Web3 data infrastructure provider, to build R6D9's data workflow.

Codatta: A Knowledge Data Production & Assetification Platform

Codatta provides R6D9 with a complete solution for data acquisition, annotation, validation, revenue distribution, and assetification, solving the core challenges of knowledge data production. It offers three core capabilities to support R6D9:

- **FaaS (Frontier as a Service) Data Contribution Mechanism**

  Codatta's FaaS model enables R6D9 to rapidly launch and manage vertical data collection tasks while attracting data contributors. Through this mechanism, R6D9 can:

  - **Define task rules** – Clearly specify data collection and annotation standards, such as environment recognition labeling and robot behavior feedback.
  - **Incentivize contributors** – Contributors earn rewards based on data quality, increasing participation.

- **Automate data validation** – Leverage Codatta's reputation system and AI quality control to ensure data reliability.

- **Data Contributor Revenue Model (Royalty Model)**

  Codatta employs a data royalty model, ensuring that contributors not only receive one-time payments but also benefit from their data's long-term value. This model includes:

  - **Contributor data ownership** – When contributors upload data, the system automatically assigns ownership based on contribution levels and records it on-chain, securing their rights.
  - **Ongoing revenue distribution** – Contributors can continuously earn from their data through multiple revenue channels.
  - **Asset liquidity** – If contributors prefer an instant payout, they can transfer or sell their data ownership for immediate earnings.

  This model ensures **short-term profits** while creating opportunities for **long-term data asset appreciation**, attracting more high-quality contributors to participate in data production.

- **Knowledge Data Assetification**

  The value of knowledge data lies not only in production but also in circulation and monetization. Codatta leverages Web3 technology to assetify R6D9's high-quality data, enhancing market liquidity through:

  - **On-chain data ownership certification** – Blockchain technology guarantees transparent and immutable data ownership.
  - **Privacy and security protection** – Privacy-preserving and compliance mechanisms ensure secure data usage.
  - **Enhanced liquidity** – Using Web3 mechanisms such as DeFi, data assets can be traded, leased, or collateralized, expanding monetization opportunities.

  By assetifying data, Codatta lowers the cost of data circulation and makes it easier for contributors to generate income. This enhances data market activity, fostering the sustainable growth of the R6D9 data ecosystem.

3.3 Action plan model

3.3.1 Hybrid Agent Strategy

In the process of building an efficient Computer Control Agent, a single type of agent has its limitations. To strike a balance between generalization ability and task execution efficiency, this architecture adopts a **Hybrid Agent Strategy**, combining the advantages of **Foundation Agents** and **Specialized Agents** to enhance system adaptability and execution efficiency.

Foundation Agents

These agents rely on **pretrained foundation models (LLMs, VLMs)** as their policy π\pi, offering several advantages:

- **Broad task adaptability**: By leveraging the contextual learning ability of foundation models, they can handle a variety of tasks across different environments.
- **No need for extensive task-specific training**: They can generate interaction actions (e.g., `click(id=search)`) directly based on natural language instructions, making them suitable for dynamically changing environments.
- **Support for multi-turn interaction optimization**: Through contextual adjustments and adaptive learning, they can improve task execution success rates.

However, they also have drawbacks:

- **Potential inefficiency**: Since Foundation Agents rely on large model inference, they may lack task-specific optimizations, leading to slower response times.
- **Require additional task-specific adjustments** to ensure execution accuracy.

Specialized Agents

These agents depend on **specially designed deep learning architectures** as their policy π\pi, offering the following benefits:

- **Optimized for specific tasks**: For instance, directly predicting UI operations (e.g., clicking at coordinates (x,y)(x, y)) to execute fixed tasks more efficiently.
- **Reinforcement learning enhancement**: Continuously optimizing strategies through interaction with the environment, improving task completion accuracy and efficiency.
- **Enables efficient execution**: Bypassing the inference process of foundation models, these agents generate specific execution actions directly based on observation information, reducing computational resource consumption.

However, they also have limitations:

- **Weaker generalization ability**, making it difficult to adapt to unknown tasks, requiring separate training for different task models.
- **Higher demand for task annotation data** and reinforcement learning optimization processes.

Hybrid Agent Strategy

To fully leverage the strengths of both approaches, this architecture employs a **hybrid strategy**, dynamically selecting the appropriate agent for task execution in different scenarios:

- **Basic task parsing**: Foundation Agents handle user natural language instructions and generate an initial task plan.

- **Task execution optimization**: If the task involves high-efficiency interactions (e.g., clicking, dragging, form filling), Specialized Agents are invoked for precise execution.
- **Adaptive environmental adjustments**: In new environments, Foundation Agents first explore and determine the optimal execution strategy, after which Specialized Agents refine the operation path for better efficiency.

This **hybrid approach** ensures both **adaptability and efficiency**, enabling strong generalization in open environments while delivering high-performance execution for specific tasks.

3.3.2 Mixed Policies

In the decision-making process of a Computer Control Agent, the policy is the core mechanism that determines the agent's next action. Different policies are suited to varying levels of task complexity and execution requirements. This architecture adopts a **Mixed Policies** approach, combining **Memoryless Policies, History-based Policies, and State-based Policies** to strike a balance between **generalization capability and execution efficiency**.

Memoryless Policies

These policies rely solely on the current observation data o_t to predict the next action a_t, without considering past observations or historical behaviors.

- **Advantages**:
  - **High computational efficiency**: No need to store or process historical information, making it suitable for high-concurrency tasks.
- **Disadvantages**:
  - **Limited ability to handle complex tasks**: Not well-suited for multi-step tasks or applications involving cross-page interactions.

History-based Policies

These policies accumulate observation and action history $h_t = (o_0, \ldots, o_{t-1}, a_0, \ldots, a_{t-1})$, using past observations and decision behaviors to guide the next action.

- **Advantages**:
  - **Enhanced task continuity**: Maintains context in multi-step tasks, improving operational accuracy.
- **Disadvantages**:
  - **Higher computational cost**: As historical data accumulates, computational complexity increases, requiring optimized storage and processing strategies.

State-based Policies

These policies maintain an internal state m_t using a state update function $m_{t+1} = f(o_t, m_t)$, tracking environmental changes to inform decision-making.

- **Advantages**:
  - **Strong environmental adaptability**: Retains long-term task state information, improving performance on complex tasks.
- **Disadvantages**:
  - **Higher computational resource demand**: State updates often rely on recurrent neural networks (RNNs) or Transformers, leading to significant computational costs.

Mixed Policies

This approach **combines History-based and State-based policies**, leveraging past actions, current observations, and external textual states to enhance task continuity and generalization ability. The system dynamically selects **Memoryless, History-based, or State-based policies** depending on task complexity.

For **high-complexity tasks** (such as e-commerce recommendations, automated trading, and intelligent data entry), **Mixed Policies** provide a more **intelligent and stable** operational capability.

4. Roadmap

The project will be developed in multiple phases to ensure the gradual maturation of the technology and the stability of the ecosystem. Below is our key roadmap:

Phase 1: Research & Prototype Development (Q1 2025)

- Develop the Browser Agent prototype, building the Environment Recognition Module to support web DOM parsing and basic UI interaction.
- Open-source the LLM-based Control Agent Framework, enabling community participation and making it the first open-source framework for browser control.
- Collect and curate the initial dataset, combining open-source data with limited human annotations.
- Publish technical whitepaper and establish the developer community.

Phase 2: Decentralized AI & Web3 Integration (Q2 2025)

- Introduce decentralized data training and validation, leveraging blockchain and Web3 technologies to enhance incentives for data contributors.
- Utilize the Codatta ecosystem to promote data assetization, ensuring transparent access and fair distribution of AI training data.

Phase 3: System Expansion & Performance Optimization (Q3 2025)

- Implement multimodal environment perception, integrating text, screenshots, OCR, and other data sources to enhance task comprehension.
- Strengthen the Actions Planning Model, optimizing execution pathways to improve task automation success rates and efficiency.
- Expand support for multiple browsers and operating systems, enhancing compatibility.
- Collaborate with the Codatta platform to establish a high-quality data sourcing and labeling pipeline, refining AI task planning capabilities.
- Release the developer SDK and open APIs for third-party integrations.

Phase 4: Computer Agent Development (Q4 2025)

- Expand from Browser Agent to Computer Agent, enabling support for desktop and mobile applications, including file management, software operation, and system task automation.
- Develop security policies and permission management systems to ensure safe and controlled AI task execution.
- Upgrade the LLM-based Control Agent Framework to facilitate broader control over desktop applications beyond the browser environment.
- Launch an experimental commercial application, inviting enterprises and developers to test AI-powered task automation.

Phase 5: Fully Autonomous AI Agent (2026 and Beyond)

- Integrate a high-autonomy AI task execution model, enabling complex cross-software and cross-device operations.
- Develop self-optimization and learning mechanisms, allowing the Agent to refine its task execution strategies based on user scenarios.
- Expand the AI application ecosystem, supporting industries such as finance, healthcare, and customer support.
- Introduce a decentralized Agent network, enabling global users to contribute and share AI task execution capabilities, achieving true AI-powered computing.

This roadmap ensures a steady technological evolution, while also focusing on real-world applications and commercialization, ultimately creating a highly efficient, secure, and scalable Computer Control Agent ecosystem.

5. Tokenomics and Governance Structure

The R6D9 ecosystem employs a sophisticated tokenomic model designed to incentivize contribution and ensure sustainable development. Token distribution follows a carefully balanced approach, allocating resources across development, community engagement, and long-term sustainability needs.

5.1 Token Utility

Payment

Users, whether human or other Agents, can use tokens to hire R6D9 to work for them.

Ownership

Tokens represent partial ownership of the dataset used for R6D9 training, allowing holders to receive a share of future data-generated revenue.

Priority

Token holders will have certain privileges, such as eligibility to participate in specific community activities or early access to new R6D9 product features.

Governance

Token holders will have the opportunity to participate in platform governance by voting on key decisions, such as protocol upgrades, resource allocation, and partnership approvals through a structured proposal and voting system.

5.2 Role and Activity

Continual Improvement

Engineers

R6D9 is a practical-oriented Robotic Agent, evolving from an intangible assistant to embodied intelligence, from software operations to physical execution. It bridges virtual and real-world applications to serve humanity. This places high demands on R6D9's architecture and capabilities, requiring collaboration among algorithm engineers, software engineers, and hardware engineers to ensure continuous delivery and iterative upgrades. These engineers will also receive corresponding incentives for their contributions.

From software operations to physical tasks.

Community

The sustainable development of the ecosystem requires not only the participation of professionals but also the support of a large number of community users. It is necessary to engage them actively in project development, promotion, and governance. This includes but is not limited to participating in R6D9's daily activities, sharing related concepts and visions, proposing initiatives or suggestions, engaging in discussions, and voting.

Knowledge Contribution

As a Robotic Agent designed to serve humans, R6D9 requires a vast amount of

knowledge data for training and fine-tuning to improve its usability and accuracy. This knowledge guides R6D9 in effectively and reliably planning task execution paths and completing tasks accurately and systematically.

Knowledge data refers to curated or processed information that possesses accuracy and directive value. This well-prepared data is crucial for R6D9's learning and improvement, typically including demonstration data and annotation data, which involve two major categories of data contributors.

Demonstrators

Robots can learn to execute tasks by observing human or expert demonstration data instead of relying solely on trial and error. In many application scenarios, such as path planning, hand-eye coordination, and grasping tasks, this method is direct and effective. Even in complex environments, demonstration data helps reduce the training search space and accelerate learning. A large number of demonstrators are needed to provide reliable demonstration data.

Annotators

In addition to demonstration data, annotation data is also essential. For instance, annotators label targets in the operational environment, such as icons on a computer desktop corresponding to software or file details, or details of physical objects in a real-world setting, including category, specifications, distribution, functionality, and interfaces.

Moreover, when the robot completes specific tasks, data on task quality—such as accuracy, precision, and unnecessary steps—needs annotation.

A substantial number of annotators are required to provide labels and feedback, which are critical for the training and continuous improvement of the robot.

Marketplace

Consumers of Data

The knowledge data that drives R6D9's evolution is not a one-time consumable. In fact, there are many possibilities for value extraction and monetization.

Firstly, this data provides value for R6D9's capability-building, allowing it to continuously offer services in the form of product functionalities and charge corresponding service fees. Every external service is essentially a re-extraction of data value, and data owners have the right to receive dividends based on their contributions.

Secondly, other Agent applications can also use this data for training to enhance their capabilities. These Agents could be similar robots or different types, such as those used for virtual reality, content creation, etc. Once produced, knowledge data can be consumed and utilized multiple times without degradation.

Finally, new Agents can leverage their capabilities to provide services, and data owners may still earn a share of the revenue from these services, depending on contractual agreements at the time of data consumption.

Overall, data owners can choose various ways to appreciate and monetize their knowledge data, such as outright purchases or continuous dividends, making data ownership a highly valuable asset.

Therefore, data assetification and trading will be crucial aspects of R6D9's development. Our ecosystem partner, Codatta, will provide vital support in this area, working with us to build a comprehensive data ecosystem.

Consumers of Robotic Functions

R6D9's ultimate goal is to become a versatile assistant for humans. Utilizing R6D9's services not only contributes to the prosperity of the Agentic economy but also helps refine R6D9 through real-world iteration.

Thus, we encourage more users to actively participate in experiencing R6D9's products, enjoying the convenience of robotic services while providing ongoing interaction and feedback.

Ecosystem Partners

Virtuals Protocol

Virtuals Protocol is building the co-ownership layer for AI Agents in gaming and entertainment. VIRTUAL offers a plug-and-play, Shopify-like solution for Agent deployment and ensures that the co-ownership of AI Agents and the provenance of contributors' work are stored on-chain, enabling broader connectivity for AI Agents across developers and consumers, thereby fostering expansive growth opportunities.

Codatta

Codatta is a platform for the production and assetification of knowledge data, providing Frontier-as-a-Service (FaaS) solutions and a vast network of data supply-side users. It helps R6D9 quickly customize and launch data-related tasks, efficiently completing data collection, annotation, and validation to maximize data value. Furthermore, as mentioned earlier, R6D9 and Codatta will collaborate deeply in data assetification and liquidity, co-creating a reliable and highly fluid data consumption market to unlock and monetize data value.

Agentic Fellows

Diverse Agent collaboration is one of the key forms of AI in the future. R6D9 itself is a multi-Agent coordination system. Moving forward, R6D9 will interact with more external Agents, which may contribute to its ecosystem in various ways. Some Agents may serve as key consumers, hiring R6D9 for services, while others may collaborate with R6D9 to accomplish more complex functionalities—these interactions are highly encouraged.

Some Agents may contribute to improving the quality of R6D9's data. We welcome and encourage these partnerships to enhance R6D9's capabilities and expand its ecosystem.

5.3 Token Allocation

Team: 5%

Ecosystem: 25%

- Knowledge Contribution: 10%
- User Incentives: 3%
- Partners: 2%
- Developer Grants: 5%
- CEX Listing: 5%

Community: 70%

- Airdrop : 5%
- Fair launch : 65%

6. Community & Ecosystem

A thriving community and ecosystem are essential for the long-term success of the Computer Control Agent. By fostering an open, collaborative, and innovation-driven environment, we aim to accelerate the development, adoption, and continuous improvement of the agent. Our ecosystem is built upon the following key pillars:

- **Open-Source Collaboration**

We believe in the power of open-source development. By making core components of the Computer Control Agent accessible to developers, researchers, and contributors worldwide, we encourage:

  - Community-driven enhancements: Developers can contribute improvements, optimize models, and expand functionality.
  - Transparency and trust: Open-source code ensures security, auditability, and reliability.
  - Interoperability: Encouraging integrations with various platforms, APIs, and applications to extend the agent's capabilities.

- **Developer and Research Support**

To foster innovation, we provide extensive support for developers and researchers interested in advancing the agent's capabilities:

  - Comprehensive SDKs and APIs: Allowing developers to build and customize applications using the agent.
  - Technical documentation and tutorials: Lowering the barrier to entry for new contributors.
  - Research grants and hackathons: Encouraging AI and automation research in academia and industry.

- **User Adoption & Real-World Applications**

Expanding real-world use cases is key to the agent's success. We plan to engage users across various domains, including:

  - Automation enthusiasts and professionals: Streamlining repetitive workflows in businesses and personal computing.
  - Enterprise integrations: Enhancing productivity in industries like finance, customer service, and software development.
  - Web3 and decentralized applications: Empowering users with AI-driven automation in blockchain-based ecosystems.

- **Web3-Powered Knowledge Economy**

We collaborate with Web3 data infrastructure providers, such as Codatta, to establish a decentralized knowledge economy where users can:

  - Contribute data and insights to enhance the agent's capabilities.
  - Earn incentives for valuable contributions to training datasets and model improvements.
  - Access AI-powered automation tools tailored for decentralized applications (dApps) and smart contract interactions.

- **Community Engagement & Governance**

To ensure that the Computer Control Agent evolves in alignment with community needs, we are committed to:

  - Community forums and governance: Enabling contributors to propose and vote on key improvements.
  - Decentralized decision-making: Exploring DAO-based governance models for sustainable ecosystem growth.
  - Educational programs and workshops: Empowering individuals and organizations to leverage AI-driven automation effectively.

By fostering an inclusive, open, and decentralized ecosystem, we envision the Computer Control Agent becoming a truly collaborative AI, evolving through community-driven innovation and real-world applications.

7. Conclusion

The Computer Control Agent is designed to overcome the limitations of traditional computer operations, enabling anyone to efficiently complete complex tasks using natural language commands. We have developed a unified agent framework that seamlessly adapts to browsers, desktop applications, and mobile environments. By integrating environment perception, data acquisition, task planning, and intelligent

execution optimization, we ensure reliability, stability, and scalability in task execution.

This whitepaper outlines the technical architecture that enables this vision, including:

- Environment Recognition and Operation: Utilizing multimodal perception (text, images, UI structures, etc.) to accurately understand the operating environment.
- Data Sourcing and Labeling Pipeline: Building high-quality datasets to enhance the agent's ability to comprehend and execute tasks.
- Actions Planning Model: Leveraging task decomposition, execution path optimization, and adaptive error recovery to improve efficiency and success rates.

Additionally, we collaborate with Web3 data infrastructure providers such as Codatta to facilitate the production and tokenization of knowledge-based data, ensuring continuous optimization for the agent. Through this framework, we aim to develop a truly universal Computer Control Agent, one that is not limited to single-task execution but can autonomously learn, adapt, and evolve, ultimately becoming an ideal bridge for human-computer collaboration.

Looking ahead, we will continue to refine the agent's task generalization capabilities, explore advanced multimodal fusion techniques, and expand its applications—transforming it into an intelligent computing assistant accessible to everyone.