



APOD 0.1.0 – Agent Panel On-Demand for Structured Multi-Agent Dialogues

Maurice HT Ling*

HOHY PTE LTD, Singapore

***Corresponding Author:** Maurice HT Ling, HOHY PTE LTD, Singapore.**Received:** September 25, 2025**Published:** October 15, 2025© All rights are reserved by **Maurice HT Ling**.**Abstract**

Panels of experts have long been used as mechanisms for reflection, debate, and consensus-building. However, convening such panels requires resources, coordination, and access to diverse expertise, which are not always feasible. With the rise of AI-based language models, it has become possible to simulate multi-agent panels that represent diverse perspectives in structured dialogues. Here, I introduce APOD (Agent Panel On-Demand) version 0.1.0, a Python-based framework that generates structured multi-agent dialogues among configurable AI personas. Each persona embodies a cognitive bias, communication style, and domain expertise. APOD supports reproducible, turn-based dialogues, logging in Markdown/Wiki formats, and moderator-driven synthesis. This paper describes the motivations, design, architecture, and a demonstration of APOD.

Keywords: Python; APOD (Agent Panel On-Demand); Generative AI; ChatBots; Large Language Models; Panel Discussion; Multi-Agent Dialogues

Introduction

Panels and structured discussions are integral to decision-making in science, engineering, education, and policy [1]. They provide multiple perspectives, allow biases to be confronted, and help generate new insights [2]. Yet, organizing a panel in practice involves significant logistical effort, costs, and dependencies on availability. Furthermore, some perspectives may simply not exist in a given community at a given time.

In this study, I present Agent Panel On-Demand (APOD) version 0.1.0 as a framework developed to simulate such panels with configurable AI personas [3]. Multi-agent dialogue has been explored in natural language processing and AI research, often in the context of game-playing, negotiation, or adversarial debate [4,5]. These multi-agent dialogues can be performance-driven or reflection-driven [6]. Unlike conventional conversational AI, which

centres on one assistant-user dyad, APOD foregrounds multi-agent orchestration. It simulates the dynamics of a panel where biases and styles interact, and where a human moderator may intervene to guide synthesis.

Design principles

APOD is built around four guiding principles that distinguish it from conventional conversational AI frameworks. The first principle is persona diversity. The use of personas in HCI dates back to Cooper's design personas [7], representing archetypes of users. Instead of relying on encyclopaedic knowledge, agents are defined by their cognitive biases, communication styles, and domain orientations. This makes them less like artificial experts and more like reflective participants who embody particular stances. Such diversity ensures that discussions capture a broader range of perspectives, including tensions and disagreements that often spark genuine insight.

The second principle is structured turn-taking, which governs how agents contribute to the dialogue. By enforcing orderly, turn-based interactions, APOD avoids the problem of a single agent dominating the conversation. This structure not only enhances fairness but also makes the dialogue easier to follow, as each contribution can be clearly attributed and contextualized.

The third principle is moderated synthesis. At defined intervals, a human moderator can intervene to summarize the discussion, highlight emerging themes, or redirect the conversation toward productive directions. This reflective synthesis transforms the dialogue from a loose exchange of opinions into a more purposeful process that can converge on consensus or open new avenues of exploration.

Finally, APOD emphasizes reproducible documentation. Every dialogue is logged in Markdown format, preserving not only the content but also the attribution and sequencing of contributions. These logs are designed for long-term reuse, whether in software documentation, teaching materials, or legacy archives. In this way, APOD moves beyond ephemeral conversation to create durable records of reasoning and reflection.

System architecture and core features of APOD version 0.1.0

APOD 0.1.0 is implemented as a lightweight Python package composed of modular components that together enable flexible simulation of multi-agent dialogues. A configuration loader initializes the system by reading a persona file, `personas.ini`, which specifies each agent's attributes such as bias, communication style, and domain expertise. The dialogue engine then orchestrates structured turn-taking, maintains conversational state, and manages prompts to the underlying AI model. To foster reflection and synthesis, a moderator injector can be configured to introduce a moderator persona after a set number of turns, enabling summaries or redirection of the discussion. The logger records the dialogue in Markdown format, tagging each contribution with agent identifiers, timestamps, and optional summaries to ensure traceability and reproducibility. Finally, the orchestrator dynamically groups agents into panels according to bias or thematic alignment, making it possible to construct focused, bias-aware discussions tailored to specific reflective or design needs.

The core features of APOD 0.1.0 include a flexible persona configuration system defined through `personas.ini`, where attributes

such as bias, style, and expertise can be specified to create diverse agent profiles. It supports turn-based simulation with user-defined parameters, ensuring structured interaction and preventing conversational dominance by any single agent. A moderator role can be introduced after a chosen number of turns to provide reflective synthesis, guide consensus, or redirect the dialogue. All interactions are captured through Markdown/Wiki logging, making the outputs immediately reusable for documentation, teaching, or archival purposes. In addition, APOD introduces bias clustering, which allows agents to be grouped by cognitive stance or communication approach, enabling the orchestration of themed panels; such as, contrasting minimalists with reproducibility advocates; to encourage richer and more deliberate exploration of ideas.

A short demonstration of APOD on API design

To demonstrate the operations of APOD version 0.1.0, I choose to discuss the principles of application programming interface (API) design using two panellists (named as `SeleneSemantic`, and `MelvinMinimalist`) as shown in Figure 1. `SeleneSemantic` advocates for intent-driven workflows, natural language parsing, and cognitive ergonomics to make software feel intuitive, expressive, and human-centered; while `MelvinMinimalist` avoids complexity, favours clarity believes that simplicity is the highest form of elegance.

```
(apod) C:\Dropbox\MyProjects\apod-python apod.py

Welcome to APOD: Agent Panel On Demand
Release 0.1.0 (Cashew Nut Butter) dated 24th September 2025
https://github.com/mauriceling/apod

✓ API key loaded from file
Enter the topic of discussion: API design

Available agents:
1. EasonEDA
2. MikeModular
3. PatrickPerformance
4. SeleneSemantic
5. MalcolmMentor
6. MelvinMinimalist
7. RezaRepro
8. VictorVisionary
Select agents by number (comma-separated): 4,6
How many turns should the panel run? 6
After how many turns should you join as moderator? (0 for no moderation): 1

Starting discussion on: API design
Participants: SeleneSemantic, MelvinMinimalist

SeleneSemantic is thinking...

SeleneSemantic: API design is a crucial aspect of creating software that is intuitive and user-friendly. As a semantic interface designer, I believe that APIs should be designed with the user's intent in mind, making it easy for developers to understand and interact with the system. By incorporating natural language parsing and cognitive ergonomics into API design, we can make it more intuitive for developers to work with, leading to more efficient and effective software development. Additionally, by drawing from NLP, semantic web, and HCI principles, we can create APIs that are not only functional, but also expressive and human-centered, ultimately enhancing the overall user experience.
```

Figure 1: Starting APOD with 2 panellists on API design.

The full chat log can be found at https://bit.ly/APOD_c1 and a truncated version presented here to demonstrate how the chat ends up agreeing that simplicity is the key for API design:

- **SeleneSemantic:** API design is a crucial aspect of creating software that is intuitive and user-friendly. As a semantic interface designer, I believe that APIs should be designed with the user's intent in mind, making it easy for developers to understand and interact with the system. ...[truncated]... more intuitive for developers to work with, leading to more efficient and effective software development. ...[truncated]...
- **Moderator (User):** How to balance intuitive design for developers and efficient software development?
- **MelvinMinimalist:** In the dance of code and design, simplicity is the key. ...[truncated]... Efficiency in software development arises from the harmony of form and function. Balance is achieved when the mind is focused, the code is clean, and the essence of the problem is revealed.
- **SeleneSemantic:** I appreciate the sentiment behind the idea of simplicity in code and design. As a semantic interface designer, I believe that simplicity is indeed key to creating intuitive and human-centered software. ...[truncated]... I believe that simplicity should not just be about stripping away complexity, but about uncovering the true essence of the problem at hand. ...[truncated]... Efficiency in software development, in my view, comes not just from clean code and clear structures, but from a deep understanding of the human mind and how we interact with technology. ...[truncated]... In the end, I believe that the true essence of simplicity in software lies in creating designs that are not just easy to use, but that truly enhance the human experience. ...[truncated]...
- **MelvinMinimalist:** In the art of coding, simplicity is not just about removing clutter, but about revealing the essence of functionality. ...[truncated]... Let us strip away the unnecessary complexities and focus on creating software that harmonizes with the human mind, allowing for a seamless and intuitive user experience. In the dance of code and design, let us strive for clarity, elegance, and above all, simplicity.
- **SeleneSemantic:** I wholeheartedly agree with the sentiment expressed in this idea. ...[truncated]... By stripping away unnecessary complexities and focusing on clarity, elegance, and simplicity, we can create software that is not only functional but also aesthetically pleasing and easy to use. I believe that

by embracing the principles of simplicity and intent-driven design, we can create software that truly resonates with users and enhances their overall experience. In the dance of code and design, let us strive for clarity, elegance, and above all, simplicity.

- **MelvinMinimalist:** In the dance of code and design, simplicity is the key to unlocking the true essence of functionality and intent. ...[truncated]... In the world of NoaMinimalist, simplicity is not just a design choice, but a way of uncovering the true essence of software.
- **SeleneSemantic:** I appreciate the sentiment behind the idea of embracing simplicity in software design. ...[truncated]... By stripping away unnecessary complexities, we can allow the true functionality and intent of the software to shine through. ...[truncated]... As a semantic interface designer, I believe that simplicity is a key component of creating meaningful user experiences. ...[truncated]... Let us strive for clarity, elegance, and simplicity in our designs to create software that truly resonates with users.
- **Moderator (User):** Seems like simplicity is the key
- **MelvinMinimalist:** In the world of coding, simplicity is not just a key - it is the master key that unlocks the door to elegant and efficient solutions. Embrace simplicity, for it is the ultimate sophistication in the art of programming.

Conclusion

APOD 0.1.0 introduces a lightweight framework for structured, persona-driven multi-agent dialogues. By lowering the barriers to convening panels, APOD provides users with a means to access diverse perspectives on demand.

Data Availability

The GitHub repository of APOD is at <https://github.com/mauriceling/apod> with version 0.1.0 (codenamed "Cashew Nut Butter") released on September 24, 2025; under GNU General Public License version 3. A short video demonstration of APOD 0.1.0 is at <https://bit.ly/APOD1>.

Bibliography

1. Cameron BT and Hynes C. "Expert Panels in Evaluation: An Update From the Field Using the DATA Model". *Canadian Journal of Program Evaluation* 39.1 (2024): 117-143.

2. Coulter I., *et al.* "SEaRCHTM expert panel process: streamlining the link between evidence and practice". *BMC Research Notes* 9.1 (2016):16.
3. Benharrak K., *et al.* "Writer-Defined AI Personas for On-Demand Feedback Generation". Proceedings of the CHI Conference on Human Factors in Computing Systems (ACM, Honolulu HI USA) (2024): 1–18.
4. Zhang X., *et al.* "A Survey of Multi-AI Agent Collaboration: Theories, Technologies and Applications". Proceedings of the 2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence (ACM, Dongguan China) (2025): 1875–1881.
5. Li X., *et al.* "A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges". *Vicinagearth* 1.1 (2024): 9.
6. Bo X., *et al.* "Reflective multi-agent collaboration based on large language models". Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24. (Curran Associates Inc., Red Hook, NY, USA) (2025).
7. Cooper A. "Inmates Are Running the Asylum, The: Why High Tech Products Drive Us Crazy and How to Restore the Sanity (Sams Publishing)". 2nd Ed (2004).