Solus: An end-to-end AI software developer.

Adam Blumenfeld

President, CSX Labs

TODO: Fill in abstract.

1 Introduction

1.1 Market Background

The seamless transfer of knowledge and data across nations powered the information age. Global access to electronic devices created high-growth platforms and services, springing millions of jobs in content creation, software development, and more. An estimated 26.9 million professional developers practiced worldwide as of 2022, growing 14.5% from 2018[1]. The inception of large language models gave rise to the generative AI space, with platforms growing at a record pace, such as OpenAI's Chat-GPT, which rose to 100 million users in two months[2]. One implication of generative AI is that it automates large portions of the jobs created in the information revolution in the tech, legal, financial, media, and support fields, to name a few[3]. S&P Global Market Intelligence predicts generative AI market revenue to hit $36 billion by 2028, forecasting code generators to lead the expansion at a compound annual growth rate of 72.9%[4]. The ability of language models to generate code has sparked new products such as GitHub's Copilot[5], Replit's Ghostwriter[6], and Amazon's Codewhisperer[7], which provide inline code completions saving developers time from repetitive tasks, routine lookups, and forgotten implementations.

Inline completions are helpful but far from autonomous, requiring manual developer intervention and revision.

Projects like Smol[8], AutoGPT[9], and BabyAGI[10] aim to perform tasks autonomously by making language models reason and evaluate themselves. However, projects like AutoGPT and BabyAGI are general in project scope, leading to a jack-of-all-trades issue.

In contrast, Smol is simple though it lacks capability and requires much human intervention.

1.2 Problems

1.2.1 Scope Limits Capability

Tools like AutoGPT and BabyAGI aim to achieve artificial general intelligence (AGI), a system that can learn to carry out any task a human or animal can perform[11]. Although a novel goal, refraining from tailoring the generative process to a specific task limits the value, the systems can bring to such a complex task as software development. A capable AI software developer project must tailor agents to the software development lifecycle.

1.2.2 Multi-Agent Collaboration

There are no effective frameworks for multi-agent communication and collaboration on a resource in an organized manner. Software development is an inherently complex process, with many scoped concerns spanning a project with many dependencies and business-related tradeoffs to weigh in. As agents act on a project's resources, a standard protocol must exist for compartmentalizing working areas, communicating issues, and resolving conflicts.

1.2.3 Manual Feedback Loop

Most tools require an amount of human feedback and approval when operating to stay on track and approve resource usage. Language model costs incurred during critical reasoning and self-regulation limit more capable projects, such as AutoGPT. These costs are a barrier to making multi-agent systems possible, as the quality of these tools' performance as software developers render the cost unjustified. Therefore, an AI software developer system must have transparent resource monitoring and self-guidance, ensuring the quality of iteration.

Adam Blumenfeld: adamb@csxlabs.org

1.3 Solution

Solus *will be*[1] an end-to-end AI software development solution utilizing self-critical, multi-role AI agents.

1.3.1 System Requirements

Solus must be robust, secure, and scalable to be a suitable system for enterprises to trust with their projects.

We achieve this by compartmentalizing agents and adding layers of abstraction to inter-agent communication, allowing us to optimize, shard, and secure resources under the hood.

1.3.2 Product Requirements

The system must be capable of doing most software development tasks, such as managing dependencies, refactoring, debugging, and generating business logic and documentation related to the project and business goals. It must execute these tasks self-regulating while ensuring all operations are transparent, traceable, and intervenable. We accomplish this by orchestrating the agents in a service mesh where sidecars intercept communications between agents and resources and beam them to a centralized control plane that controls downstream processes and provides visibility over the operation.

1.3.3 Business Requirements

Solus must be versatile, profitable, and sustainable. By making agents customizable components with a clear separation of concerns, we can give Solus a clear path of expansion to other applications in the generative AI space. By operating on a cloud model, we can stay profitable on a low operating margin minimizing costs for us and our customers. To make Solus sustainable, we plan on creating a plugin ecosystem for optimized agent types for specific tasks and technologies. These developer ecosystems give Solus an economic moat over other initiatives. We also plan on building Solus to be open-source to ensure the most significant impact on the developer community. Open source allows us to create an

---

[1]NOTE: We are raising funding to build Solus due to the steep upstart costs. Contact the corresponding author if you want to contribute financially.

ecosystem around our technology while maintaining profit and advantage due to enterprise trust in our platform.

2 System Architecture

2.1 Components

As shown in Figure 1, the Solus system comprises three primary components:

1. **Control Plane**: The control plane orchestrates agents and manages resources. It also provides an entry point for users to interact with the system and monitor operations. Agents and Resources beam all actions to the control plane, which analyzes them and decides how to proceed.

2. **Agents**: Agents act on resources and interact with each other to complete the task given by the user. Agents are customizable components with roles and a set of allowed operations. They communicate with each other to regulate each other. For example, a Project Management agent may help a Software Engineer agent to stay working on items relevant to the business concerns of the system. Agents are also responsible for self-regulation and monitoring their resource usage.

3. **Resources**: Agents interact with resources such as the filesystem, internet, and databases to complete tasks. Resources are abstracted away from agents, allowing us to optimize and shard them under the hood. Resources are also responsible for monitoring and reporting their usage to the control plane.

The abstraction of agents and resources decouples the generative process from the underlying infrastructure, allowing us to optimize and scale the system without affecting the generative process. It also provides room for expansion and customization, allowing us to create a plugin ecosystem for agent types and resource types.

2.2 Inter-Service Communication

As shown in Figure 2, the services interact through a service mesh. Sidecars intercept and standardize services' communications and beam them to the control plane. The control plane
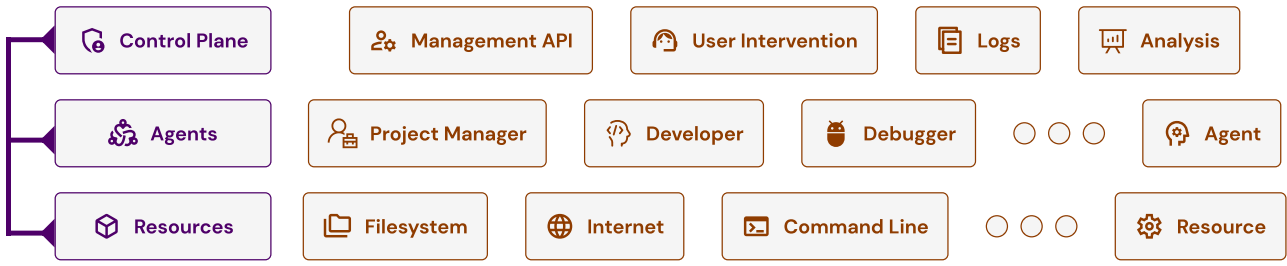
Figure 1: The system comprises a control plane, a service mesh, and a resource layer. The **control plane** orchestrates agents and manages resources. The **agents** are responsible for doing the generation, acting on different resources. **Resources** provide an interface for agents to interact with their environment, such as the filesystem and the internet, to get context and complete tasks.
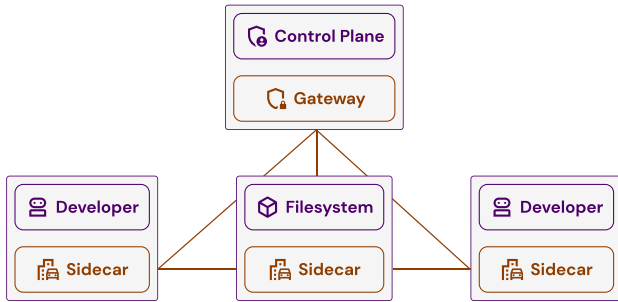

Figure 2: The services interact through a service mesh. Sidecars intercept and standardize services' communications and beam them to the control plane.

keeps logs of all system activity, intervening when needed or requested by the user. The other components of the system regulate their communications and complete actions independently. A service mesh allows agents and resources to regulate their own interactions while keeping a centralized log of all activity in the control plane, allowing us to monitor and regulate the system while abstracting away the underlying infrastructure from the agents.

The service mesh design pattern was initially used to manage microservices in a cloud-native environment due to its de-coupling of the network layer from the application layer[12]. Due to the scale and compute requirements of competent language models[13], we must run agents on separate machines. A service mesh allows us to scale the system horizontally without affecting the generative process.

2.3 Agents

References

[1] Qubit Labs. How Many Programmers are there in the World and in the US? [2023]. *Qubit Labs*, nov 29 2022. [Online; accessed 2023-07-16].

[2] David Carr. Chatgpt Topped 1 Billion Visits in February. *Similarweb*, mar 7 2023. [Online; accessed 2023-07-16].

[3] Aaron Mok and Jacob Zinkula. Chatgpt may be coming for our jobs. Here are the 10 roles that AI is most likely to replace. *Insider*, jun 4 2023. [Online; accessed 2023-07-16].

[4] SungHa Park. Generative AI Software Market Forecast to Expand Near 10 Times by 2028 to &P Global Market Intelligence Says. *S&P Global*, jun 8 2023. [Online; accessed 2023-07-16].

[5] Thomas Dohmke. Github Copilot is generally available to all developers. *The GitHub Blog*, jun 21 2022. [Online; accessed 2023-07-16].

[6] Amjad Masad, Samip Dahal, Giuseppe Burtini, and Alexandre Cai. Ghostwriter AI & Complete Code Beta. *Replit Blog*, sep 8 2022. [Online; accessed 2023-07-16].

[7] Sylvia Engdahl. Amazon CodeWhisperer, Free for Individual Use, is Now Generally Available. *Amazon Web Services*, apr 13 2023. [Online; accessed 2023-07-16].

[8] Anton Osika. Current state of virtual developers. *Future tools*, jun 4 2023. [Online; accessed 2023-07-16].

[9] Sabrina Ortiz. What is Auto-GPT? Everything to know about the next powerful AI tool. *Zdnet*, apr 14 2023. [Online; accessed 2023-07-16].

[10] Sriram Parthasarathy. Meet BabyAGI — The Autonomous AI Agent to Streamline Your Tasks. *Towards AI*, may 10 2023. [Online; accessed 2023-07-16].

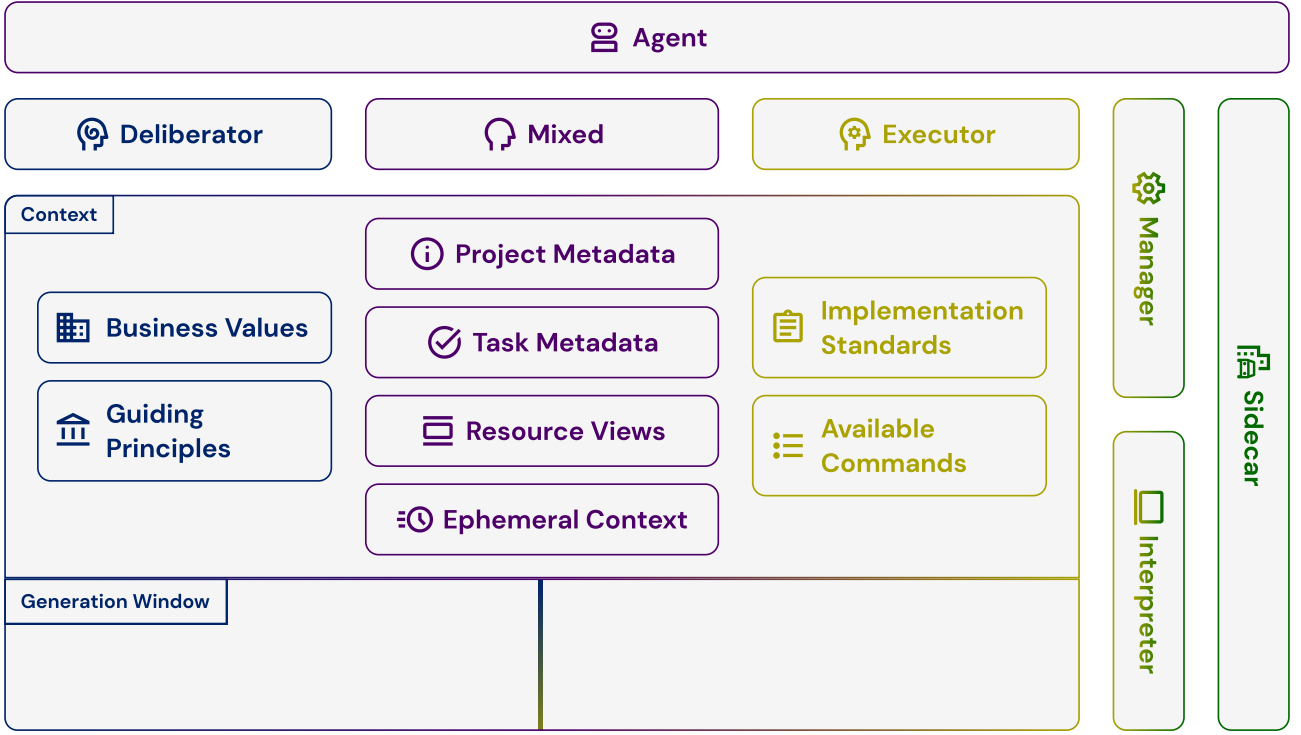[11] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric

Figure 3: The services interact through a service mesh. Sidecars intercept and standardize services' communications and beam them to the control plane.

Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4. https://arxiv.org/abs/2303.12712, mar 22 2023.

[12] Wubin Li, Yves Lemieux, Jing Gao, Zhuofeng Zhao, and Yanbo Han. Service Mesh: Challenges, State of the Art, and Future Research Opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 122–1225.

[13] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and Efficient Foundation Language Models. https://arxiv.org/abs/2302.13971, feb 27 2023.