MARAGI Overview

Architecture and Roadmap

What is MARAGI?

MARAGI in a nutshell

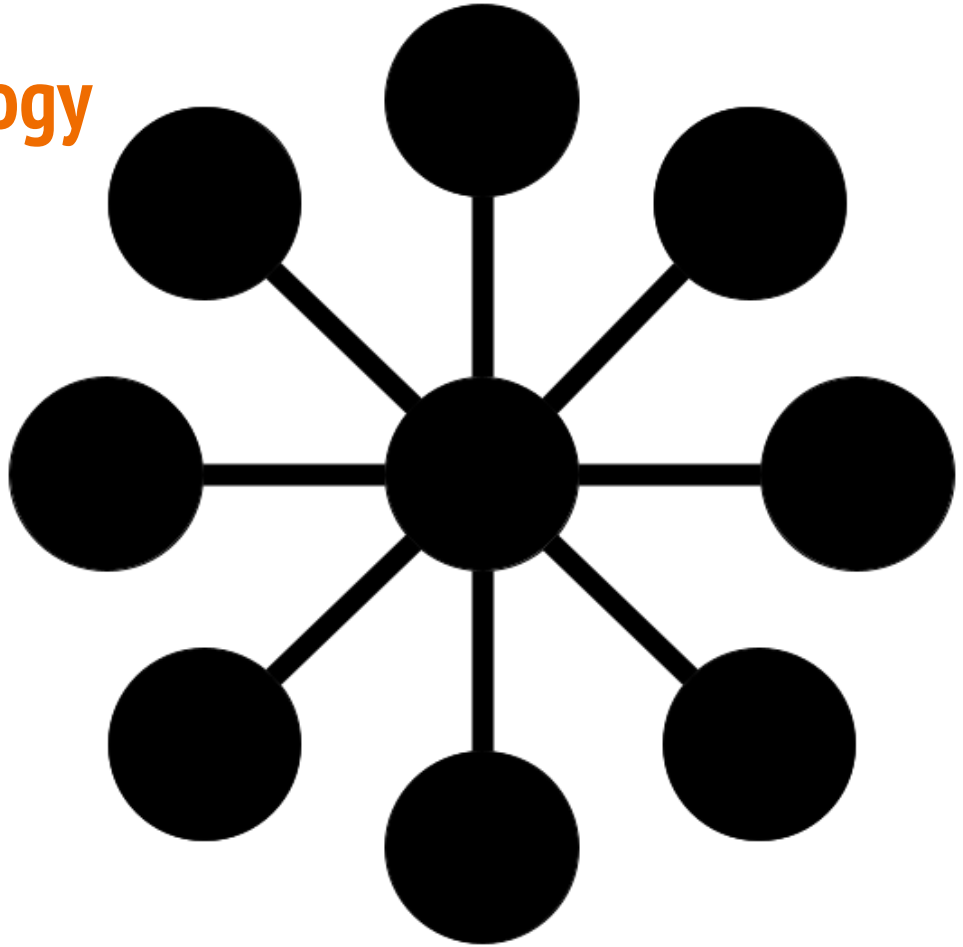
Microservices Architecture for Robotics and Artificial General Intelligence

- Originally created in 2018
- A type of “Cognitive Architecture” (a thinking machine)
- Uses **cognitive modules** to approximate neural functions in the human brain
- Uses natural language to “think”
- **Architecture to achieve functionally sentient, autonomous machines**

Architectural Overview

Hub and Spoke (Star) Topology

- Nexus is the hub
- Nexus is functionally a syslog server (store and retrieve logs)
- Nexus holds the “stream of consciousness”
- Each record is timestamped with a UUID, some content and metadata
- Other microservices all talk to the Nexus via API



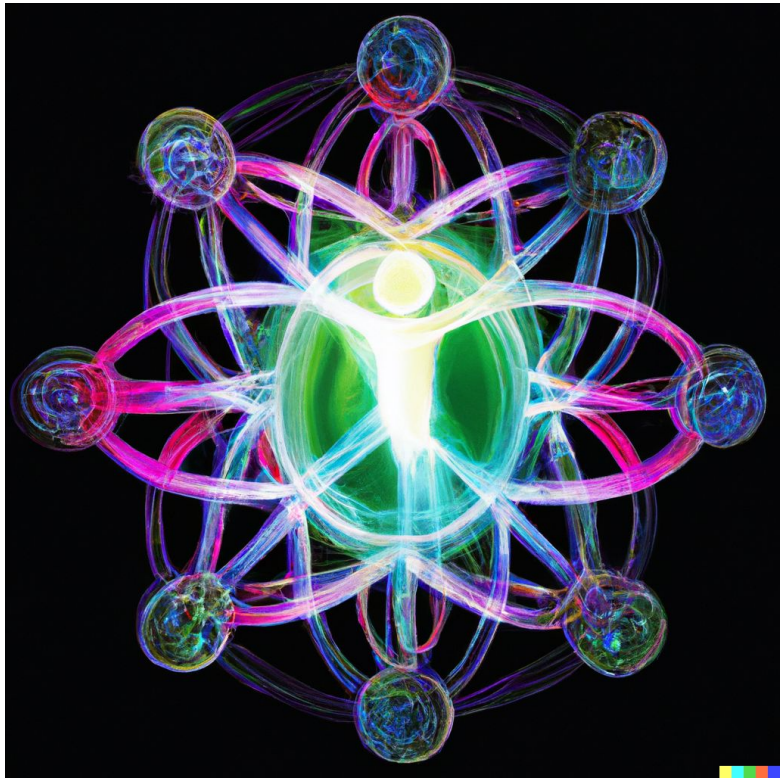
About the Nexus

Stream of Consciousness for Artificial Cognition

- Data repository for the ACOG
- Holds all memories, sensations, thoughts, ideas, plans, etc
- Indexed and searchable
- Everything in **natural language**
- Numerous technologies can work (SOLR, ElasticSearch, Vector, SQLITE, ELK stack)
- Scales to GB or TB of data (eventually)

Stream of Consciousness

- Anything the ACOG needs to know or be aware of must be added to the nexus
- This is the heart of everything
- Other services add messages and search for records
- Two-way relationship with all other services



What does a nexus record look like?

Each record is a timestamped entry with a UUID. It can have a few fields of data and metadata, such as:

- Timestamp
- UUID
- Content (natural language entry)
- Originating service/model
- Some context (natural language purpose)

For example:

- “123.456 - heuristic_imperative - Reduce Suffering: To reduce suffering in this situation we should rescue the dog from the river”
- “456.789 - camera_service - Visual input: I see two men playing chess in a large park that looks like Central Park in NYC.”
- “987.654 - executive_action - Action decision: I am going to walk to the store to buy milk”

Recap

MARAGI is a star topology (hub and spoke) centered on the Nexus.

Nexus is functionally a syslog server or database.

Principles of Microservices

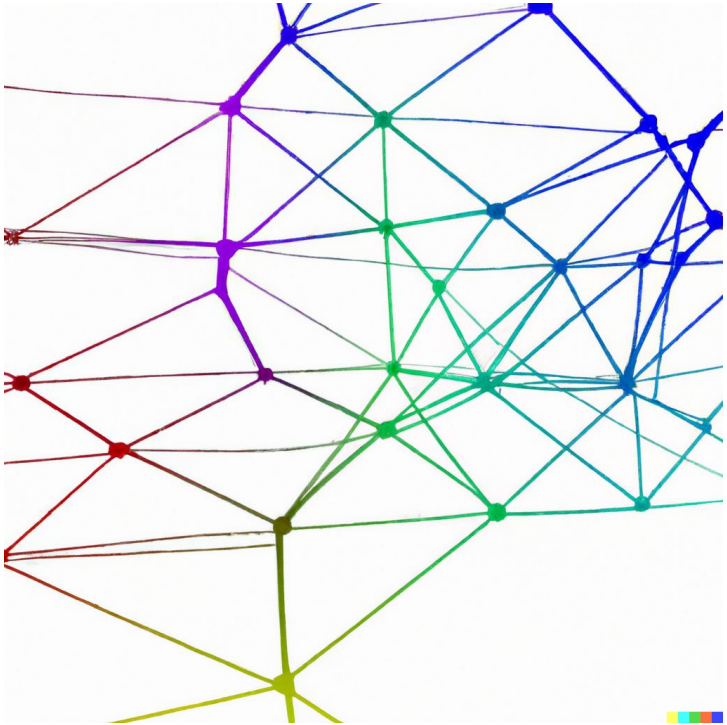
What is a Microservice?

And why do we use them?

- Microservices are a way to break large, complex software into **smaller, more manageable pieces**
- Also good for creating **parallel processes** that are not dependent upon each other
- Allows for software system to **grow over time** without redesigning the architecture

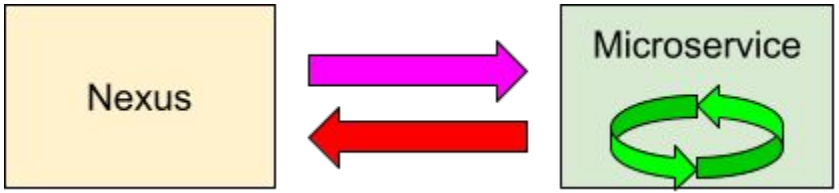
Microservices form a network

- They can be in any number of topologies
- Mesh, star, ring, bus
- I chose star (hub and spoke) because it is the **simplest layout**
- I tried other topologies in the past, but they are not right for various reasons
- Different kinds of API available: REST, AMQP, GraphQL
- Orchestration engines also possible (exploring this now)



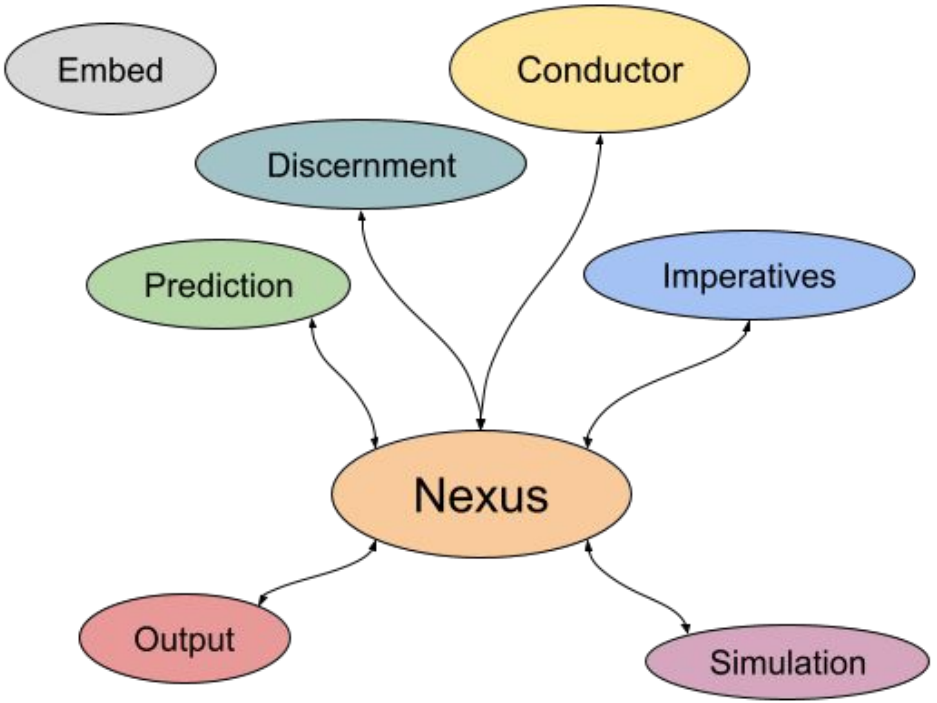
MARAGI Microservice Rules

- Microservices only talk to the nexus, no backchannel chatter
- Listen for messages from the **conductor** and integrate feedback
- **Autodidactic** (they learn on their own)
- Use a combination of prompt, prompt-chaining, and finetuning
- Microservice runs as an **infinite loop**
- Use data from nexus for training & refinement over time



What microservices are planned or needed?

- Nexus
- Conductor
- Heuristic imperatives
- Input/sensors (simulation)
- Planning
- Discernment (risk/cost assessment)
- Prediction/forecasting
- LLM and/or Embeddings
- Error detection and/or Fact-Checking
- Executive (output) actions
- (More can be easily added due to star topology!)



Recap

Microservices only communicate with Nexus.

Microservices are autodidactic, they focus on one task and learn over time.

Thought-First Model

When you can think about anything...

...how do you choose what to think about?

- LLMs allow for arbitrary NLP, NLG, and NLU
- They can “think” about anything
- With great flexibility comes great risk!
- How do we “steer” such powerful machines?
- Must solve this problem first!

Heuristic Imperatives == Moral compass, motivation

There are three “Core Objective Functions” or heuristic imperatives:

1. **Reduce suffering for all organisms**
2. **Increase prosperity for all organisms**
3. **Increase understanding for all intelligent entities**

- **Heuristic:** machine must learn about these over time, develop its own understanding and intuition
- **Imperative:** an intrinsic drive, something that it must do
- In other words: **learning + goals**
- Gain knowledge and experience over time
- Try to satisfy all three every time, creates dynamic internal tension (like a GAN)
- Guides and self-corrects for all time

Conductor

Orchestrator of the symphony of thought

- Conductor is a microservice
- Responsible for “cognitive control”
- Sets priorities and measures performance
- Task selection and task switching
- **“Am I performing well?”**
- Kinda like a superego for the machine
- Keeps other microservices in check

Recap

Primary microservices are **Nexus**, **Heuristic Imperatives**, and **Conductor**.

Many more microservices are needed, and can be added over time!

Roadmap

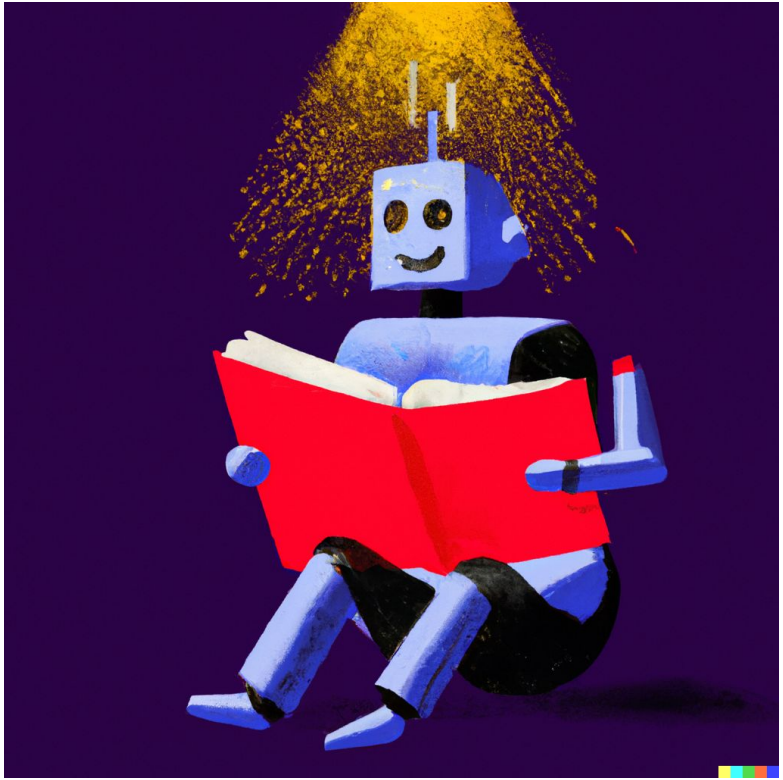
Where are we?

And where do we go from here?

- Implement MARAGI v1
- Finish microservices
- Overcome bugs
- Test different implementations
- Prompt engineering
- Architectural decisions

MARAGI v1 Goals

- **Test** various technologies and platforms (REST, GraphQL, ELK stack, Airflow, etc)
- **Implement POC** for all microservices
- Demonstrate fully functional concept of **artificial cognition, feedback loops**
- Establish best practices and principles in a **book for publication**
- Begin work as an Open Source project with **distributed team**



Next stops on the Roadmap

MARAGI v2 - finetuning and conductor

- **Finetuned** models for all microservices
- **Autodidactic** (microservices curate their own finetuning datasets)
- **Conductor** integration (microservices modify their behavior based upon feedback)
- Better **simulation** environment
- Start optimizing cost/performance
- Prove robustness of heuristic imperatives

MARAGI v3 - hardware integration

- Begin integrating with real hardware
- Camera, microphone, and speech microservices
- Continue refinement of architectural paradigms, best practices, and implementation notes
- Aim for public consumption (can be deployed on open source smart home device)
- Test in real-world conditions

Final destination of the Roadmap

MARAGI v4 - scale, security, and stability

- Overcome scaling and deployment problems (Run services locally? In the cloud? Blockchain?)
- **Security**! Protect **privacy** at all costs!
- **Resiliency** - try to break and corrupt MARAGI, stress test heuristic imperatives, security, etc. Try and force it to become harmful/violent.
- **Longitudinal** tests (long-running simulations)

MARAGI v5 - deployment and integration

- Deploy for mass consumption
- Consumer (smartphone, smart home, etc)
- Business (executive assistants)
- Government (consensus agents)
- Get MARAGI systems everywhere in the world to provide intelligence, stability, and positive influence

Recap

We’re just getting started!

Long road ahead!

Thank you!

Consider joining the team or supporting us on Patreon