34th Annual **INCOSE** international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

Ryan Bell, *Naval Postgraduate School, Naval Information Warfare Center Atlantic*
Ryan Longshore, *Naval Postgraduate School, Naval Information Warfare Center Atlantic*
Raymond Madachy, PhD, *Naval Postgraduate School*

Using AI tools for SE, from Requirements Generation and Management to Risk Identification, Analysis, and Management

Agenda

- Background
- Systems Engineering Processes
- Language Model Interaction Frameworks
- Case Study Across Frameworks: All-Terrain Bike
- Singular Framework
- Sequential Framework
- Challenges and Limitations
- Summary
- Future Work

LLMs

Large Language Models (LLMs) are neural networks that have revolutionized the field of natural language processing (NLP) by demonstrating an impressive ability to understand and generate text. Many models have also demonstrated an ability to generate code in common languages (e.g. Python).

Training: Teaching a model how to perform a task using large amounts of data

Inferencing: Model making inferences based on new inputs

Common LLMs (not an exhaustive list)

- Proprietary: OpenAI ChatGPT, Google Gemini, Anthropic Claude
- Open source: LLaMa, Falcon, Mistral

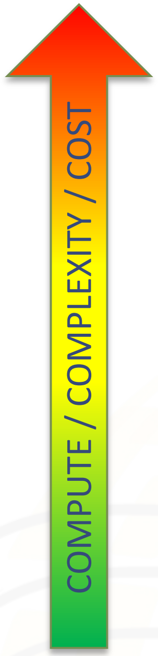
Model Characteristics

- Size is the number of parameters (e.g. 70B model = 70B parameters)
- Quantization is a compression technique used to reduce memory and compute requirements. It converts weights to a lower precision data type (e.g. 32 bit floating point → 8 or 4 bit integer).

Challenges

- Hallucinations
- Access to Unique Training Data
- Inelegant (not optimized) Code Generation

Improving LLM Responses

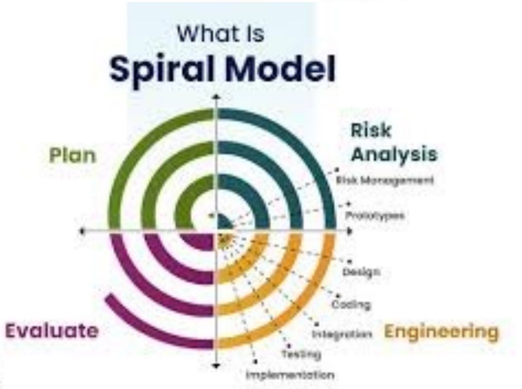


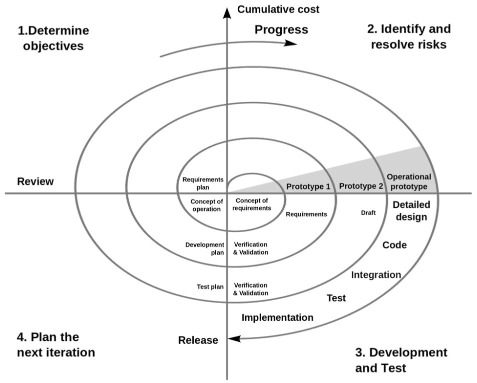
- **Re-Training:** Re-train base model with new dataset
- **Finetuning:** Teach the base model specialized knowledge by adjusting a small number of internal parameters. Significant time savings compared to re-training entire model
- **Retrieval Augmented Generation (RAG):** Utilize external data to augment the LLM’s knowledge without changing model weights
- **Prompt Engineering:** Optimize prompts for optimal outputs
  - Few Shot prompts teach LLMs to perform a task through examples
  - Zero Shot prompts challenge the LLM to perform a task correctly when it has not been correctly trained for that task

<u>These methods can also be combined</u>

Systems Engineering Processes

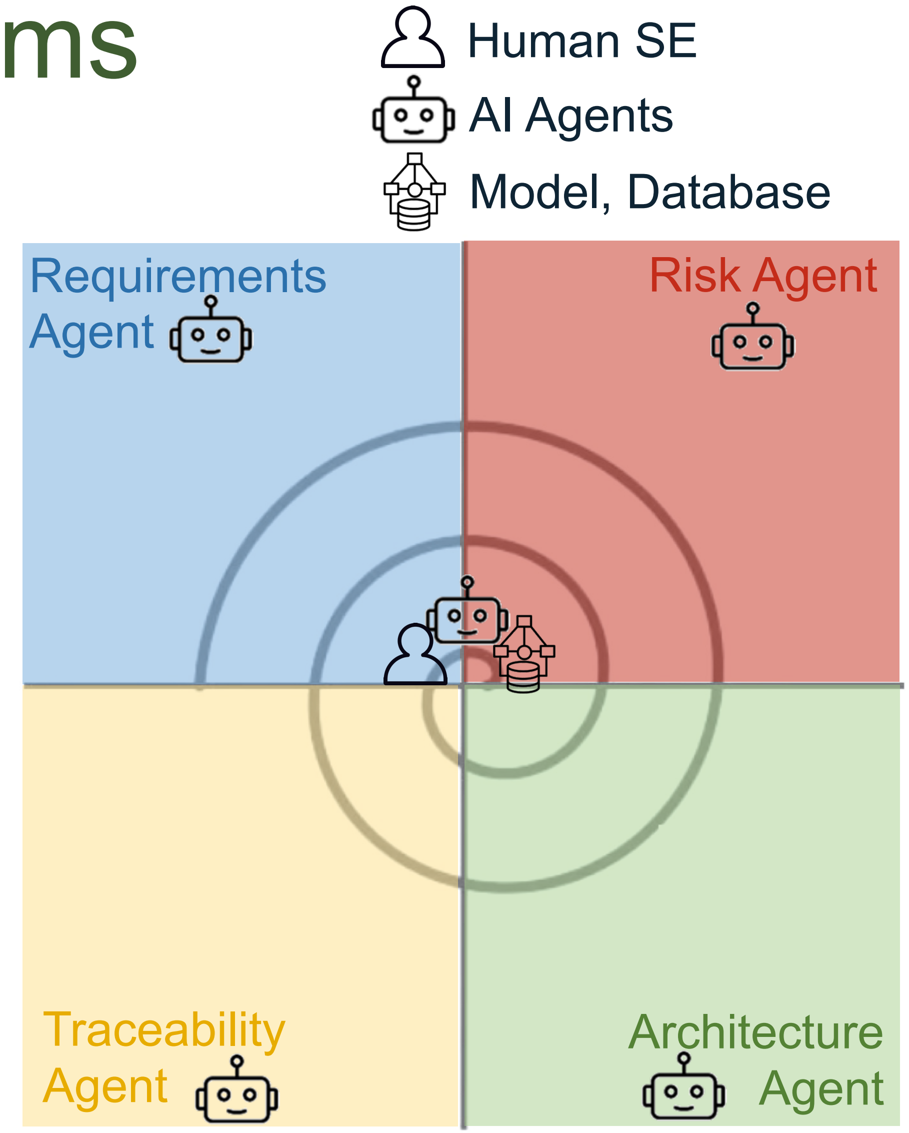
- Systems Engineering “Vee” Model
- Waterfall
- Spiral Model





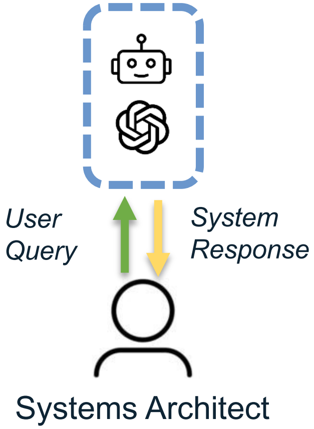
Systems Engineering Processes: Augmenting with AI LLM Teams

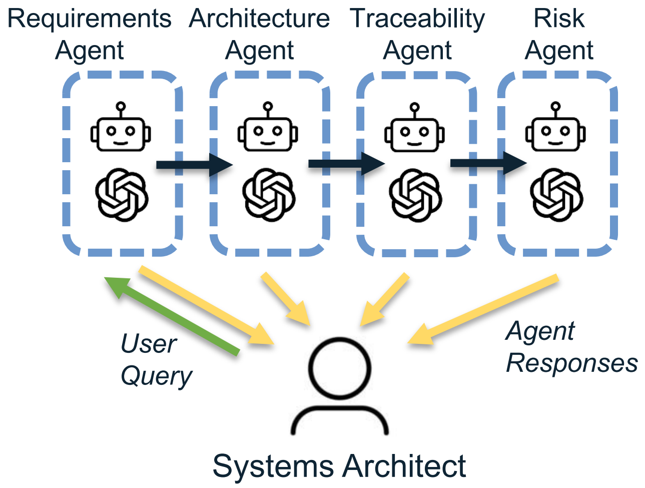
- Normally the human SE would work through the spiral themselves
- Pair the human SE with an AI SE as spiral development progress
- Specialized agents provide feedback/input along the spiral for iterative development
  - Both for the system architecture and the model (e.g. SysMLv2)

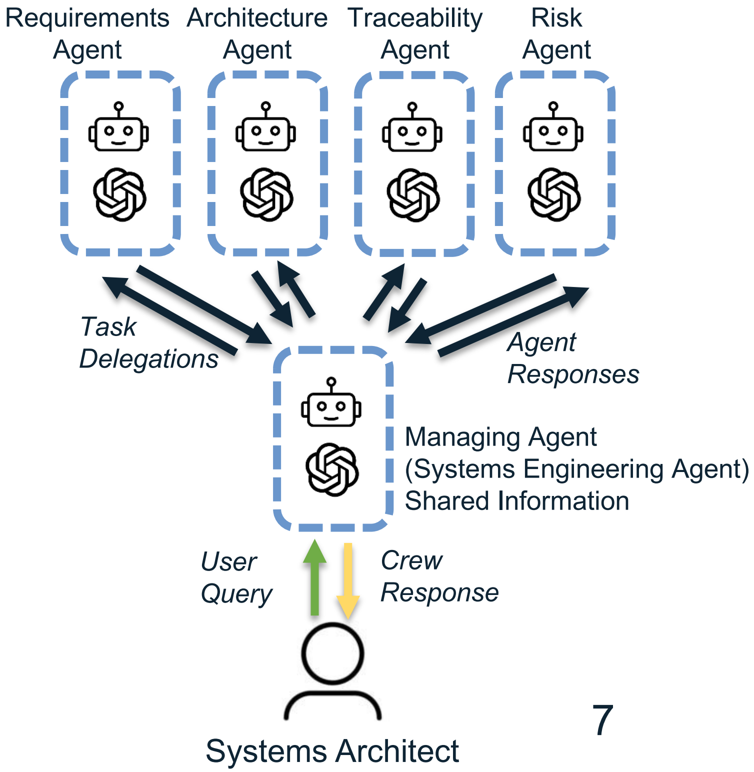


Language Model Interaction Frameworks

- Singular (e.g. Manual Chat Interface)
  - High human interaction and guidance
  - Human provided query chains
    - Potentially inconsistent framework
- Sequential
  - As needed human interaction/review
  - Guidance and roles pre-defined for agents
    - Enforces consistent framework
  - Outputs given from each agent to human
- Hierarchical
  - As needed human interaction/review
    - Managing LLM handles orchestration of tasks to agents
  - Singular output provided from Managing LLM

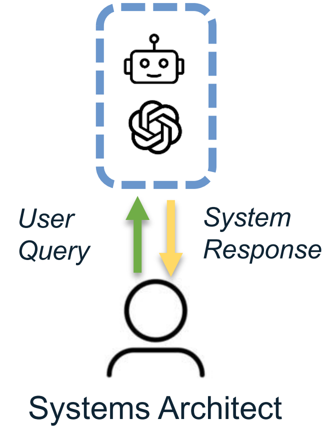


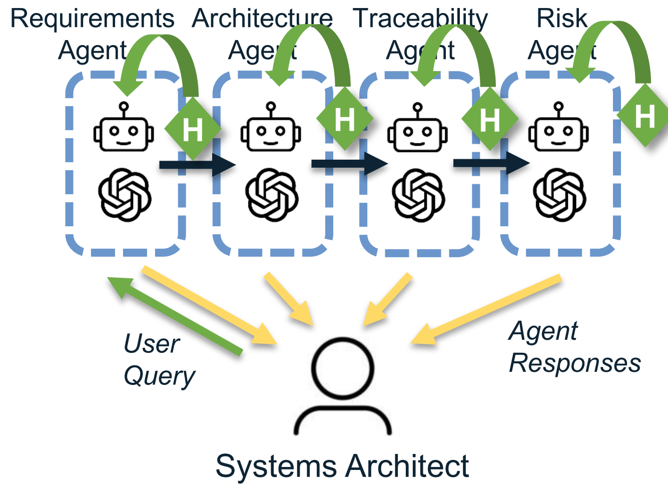


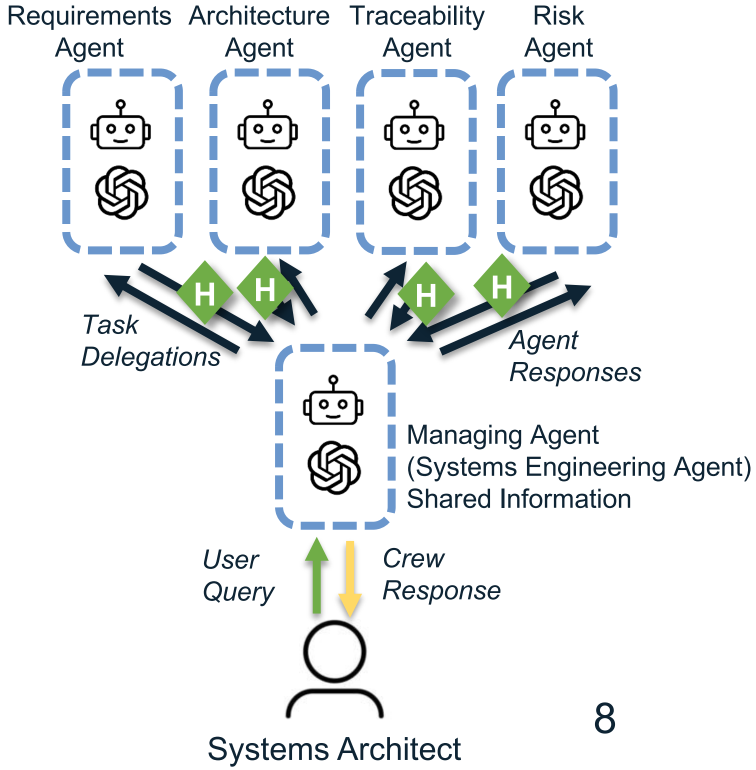


Language Model Interaction Frameworks

- Some implementations of frameworks allow for human interjection for iterative refinement and approval of tasks (sequential) and delegated tasks (hierarchical)
  - These typical points are marked on the diagrams to the right H

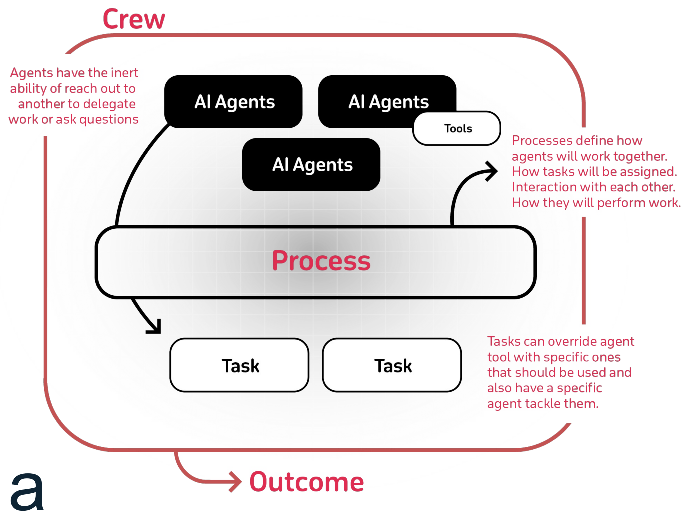






Language Model Interaction Frameworks: CrewAI

- CrewAI is a multi-agent framework built on top of LangChain
- “CrewAI is designed to enable AI agents to assume roles, share goals, and operate in a cohesive unit - much like a well-oiled crew. Whether you're building a smart assistant platform, an automated customer service ensemble, or a multi-agent research team, CrewAI provides the backbone for sophisticated multi-agent interactions.”
- https://github.com/joaomdmoura/crewAI



Agents have the inert ability of reach out to another to delegate work or ask questions

Processes define how agents will work together. How tasks will be assigned. Interaction with each other. How they will perform work.

Tasks can override agent tool with specific ones that should be used and also have a specific agent tackle them.

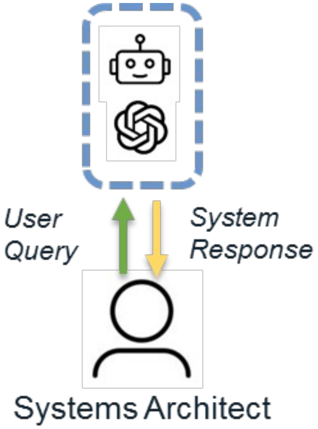
AutoGen is another popular Python package for orchestration of agents

https://microsoft.github.io/autogen/

Case Study Across Frameworks: All-Terrain Bike

- Background and Intent
  - We want to see how seamlessly we can chain together language models through an abbreviated systems engineering design cycle, from conceptualization to requirements, use cases and functions, traceability back to requirements, and risk identification and monitoring.
- Frameworks Tested:
  - Singular
  - Sequential
- Input provided:
  - We will provide the following system description as context for each method implemented
    - Develop a highly versatile bike capable of navigating diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts. The bike should be designed with features that ensure optimal performance, safety, and comfort in varying conditions.
- Outputs Expected:
  - A list of requirements with their types
  - A list of use cases and up to 2 decomposed layers of functions
  - A list of requirements traced to functions
  - A list of risks for the project

Singular Framework



Bicycle Case Study using ChatGPT-4o

- Provide system context ~ 1-2 paragraphs
- Provide context for each task
  - Background prompt describing the expertise of the person who would perform the task
  - Problem statement describing the task to be completed
  - Longer, detailed prompts often equate to much better responses
- Generate requirements
- Generate use cases
- Consolidate functions and allocate them to multiple use cases using an allocation matrix
- Map functions to hardware / system components
- Trace hardware components to requirements with a matrix
- Generate risks based on previous generated artifacts

ChatGPT Log: https://chatgpt.com/c/84f6165a-1726-4ac6-a058-e24b805a9e0f

System Description

The following system description was used to generate requirements, use cases, traceability between requirements and use cases, and risks:

Develop a highly versatile bike capable of navigating diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts. The bike should be designed with features that ensure optimal performance, safety, and comfort in varying conditions.

ChatGPT-4o Requirements Prompt

Requirements Prompt:
You are an experienced Requirements Engineer with a strong background in gathering, analyzing, and documenting requirements for complex systems. Your keen eye for detail and ability to translate user needs into clear, actionable requirements quickly made you an expert in requirements engineering. You have worked in various industries, including defense, aerospace, and automotive, where precision and clarity in requirements are crucial.

Your primary objective is to ensure that all project requirements are accurately captured, well-documented, and understood by all stakeholders. You aim to bridge the gap between clients and the development team by translating business needs into technical specifications. Your success is measured by the completeness and clarity of requirements, minimizing ambiguities, and ensuring that the final product meets all specified requirements. Only decompose requirements to a maximum of 3 levels. Types of requirements should include functional, non-functional, and interface requirements. Remember that all requirements should follow SMART criteria: Specific, Measurable, Achievable, Relevant, and Testable.

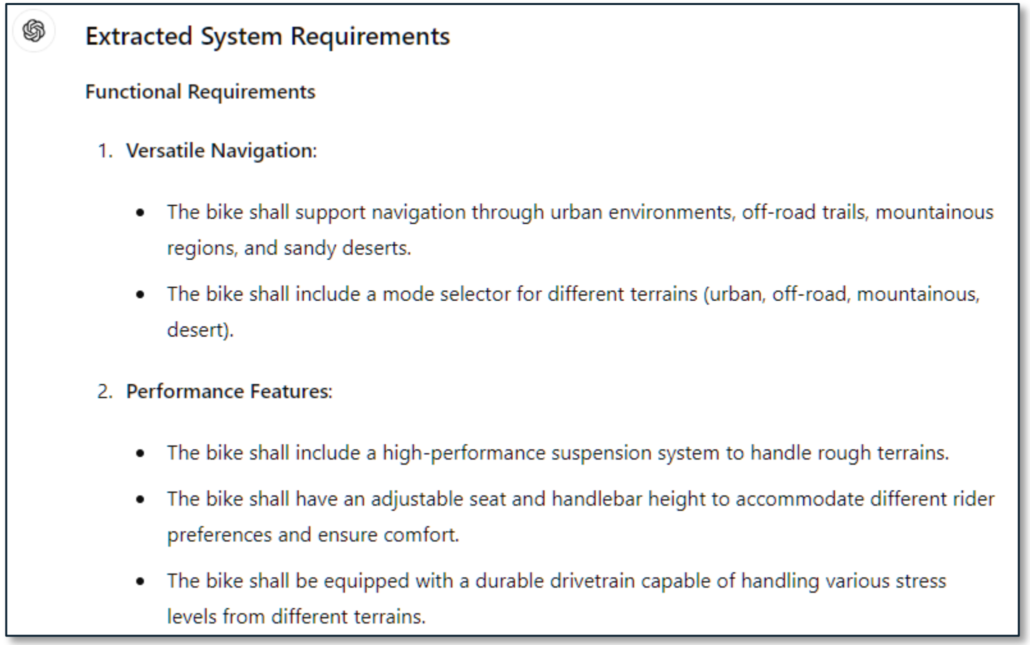
Analyze the provided user description of a desired system and extract clear, measurable, and relevant system requirements. These should include functional, non-functional, and interface requirements that are essential for the system design.

<<System Description>>

ChatGPT-4o Generated Requirements

Generated 22 Requirements

- 11 Functional
  - Example: The bike shall include a mode selector for different terrains (urban, off-road, mountainous, desert)
- 6 Non-Functional
  - Example: Usability – The bike shall require minimal maintenance, with routine checks needed no more than once per month.
- 5 Interface
  - Example: The bike shall have mounting points for additional accessories such as GPS units, water bottle holders, and panniers



Extracted System Requirements

Functional Requirements

1. **Versatile Navigation:**
   - The bike shall support navigation through urban environments, off-road trails, mountainous regions, and sandy deserts.
   - The bike shall include a mode selector for different terrains (urban, off-road, mountainous, desert).
2. **Performance Features:**
   - The bike shall include a high-performance suspension system to handle rough terrains.
   - The bike shall have an adjustable seat and handlebar height to accommodate different rider preferences and ensure comfort.
   - The bike shall be equipped with a durable drivetrain capable of handling various stress levels from different terrains.

ChatGPT Log: https://chatgpt.com/c/84f6165a-1726-4ac6-a058-e24b805a9e0f

ChatGPT-4o Use Case Prompt

Use Case Prompt:
You are a seasoned Systems Architect with extensive experience in designing and overseeing the implementation of complex systems. Your journey began as a software developer, but your passion for seeing the bigger picture led you to systems architecture. Over the years, you have designed systems for various high-stakes projects, from defense systems to large-scale enterprise applications.

Your mission is to create a robust, scalable, and efficient system architecture that meets all project requirements and constraints. You focus on high-level design, ensuring that all components of the system work together seamlessly. Your goal is to deliver a design that not only meets current needs but is also flexible enough to accommodate future changes and expansions.

Using the system requirements and the original user description, develop detailed use cases that demonstrate how the system will fulfill these requirements. Subsequently, identify and define system functions necessary to perform the tasks outlined in the use cases.

<<System Description>>

ChatGPT-4o Generated Use Cases

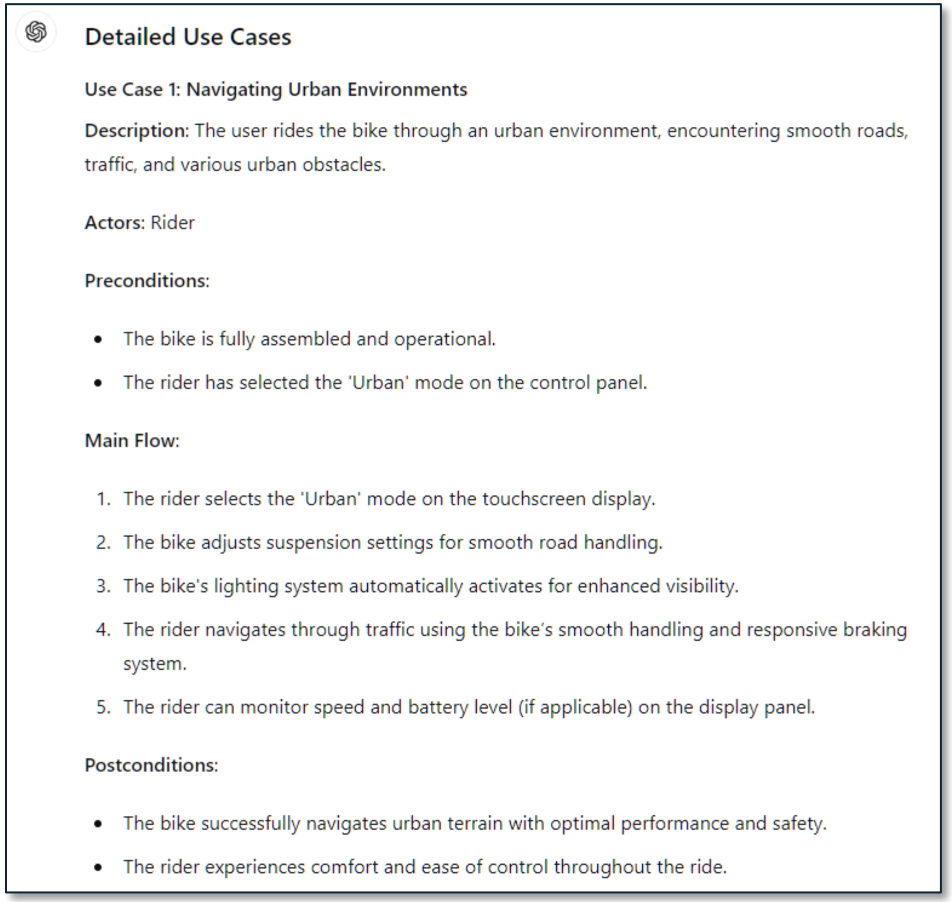
Generated 4 Use Cases

- Navigating Urban Environments
- Navigating Off-Road Trails
- Navigating Mountainous Regions
- Navigating Sandy Deserts

For each use case, a **description, actor, preconditions, main flow,** and **postconditions** were generated.

In each case, the rider was identified as the actor. No use cases were generated for maintaining the bicycle. However, a fully assembled and operational bike was identified as a precondition for each use case.

18 Functions were generated (e.g. Suspension Adjustment, Tire Pressure Adjustment).



Detailed Use Cases

Use Case 1: Navigating Urban Environments

Description: The user rides the bike through an urban environment, encountering smooth roads, traffic, and various urban obstacles.

Actors: Rider

Preconditions:

- The bike is fully assembled and operational.
- The rider has selected the 'Urban' mode on the control panel.

Main Flow:

1. The rider selects the 'Urban' mode on the touchscreen display.
2. The bike adjusts suspension settings for smooth road handling.
3. The bike's lighting system automatically activates for enhanced visibility.
4. The rider navigates through traffic using the bike's smooth handling and responsive braking system.
5. The rider can monitor speed and battery level (if applicable) on the display panel.

Postconditions:

- The bike successfully navigates urban terrain with optimal performance and safety.
- The rider experiences comfort and ease of control throughout the ride.

ChatGPT Log: https://chatgpt.com/c/84f6165a-1726-4ac6-a058-e24b805a9e0f

ChatGPT-4o Traceability Prompt

Traceability Prompt:
You are a dedicated Systems Engineer with a background in quality assurance and requirements traceability. Your career has focused on ensuring that all project artifacts are traceable throughout the development lifecycle. You have worked in highly regulated industries such as aviation, aerospace, and defense where traceability is critical for compliance and safety.

Your primary goal is to establish and maintain comprehensive traceability matrices that link requirements to architecture products, design, implementation, and testing artifacts. You ensure that every requirement is traced through the entire development process, from initial specification to final verification. Your aim is to provide clear and auditable traceability, facilitating impact analysis and ensuring compliance with industry standards.

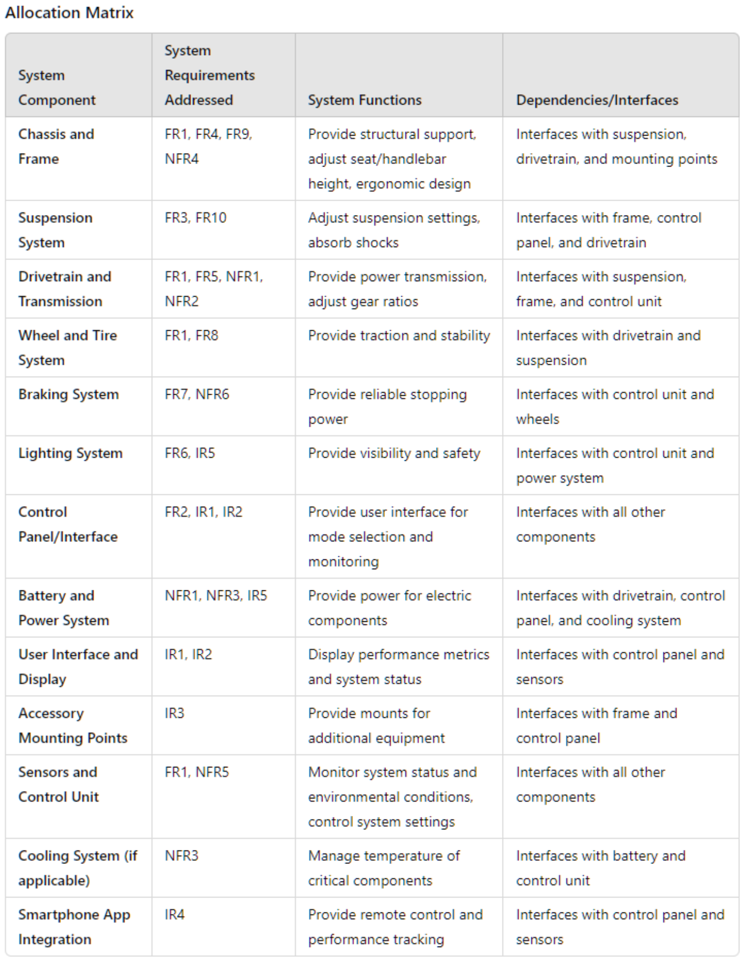
Use the original user description, system requirements, and system functions, map these functions to potential system components. Then, create an allocation matrix that shows how each component addresses specific system requirements, highlighting dependencies and interfaces between components and requirements.

ChatGPT-4o Generated Traceability

Generated traceability between requirements and functions

Allocated system components to functions

Defined interfaces between system components



Allocation Matrix

System Component	System Requirements Addressed	System Functions	Dependencies/Interfaces
Chassis and Frame	FR1, FR4, FR9, NFR4	Provide structural support, adjust seat/handlebar height, ergonomic design	Interfaces with suspension, drivetrain, and mounting points
Suspension System	FR3, FR10	Adjust suspension settings, absorb shocks	Interfaces with frame, control panel, and drivetrain
Drivetrain and Transmission	FR1, FR5, NFR1, NFR2	Provide power transmission, adjust gear ratios	Interfaces with suspension, frame, and control unit
Wheel and Tire System	FR1, FR8	Provide traction and stability	Interfaces with drivetrain and suspension
Braking System	FR7, NFR6	Provide reliable stopping power	Interfaces with control unit and wheels
Lighting System	FR6, IR5	Provide visibility and safety	Interfaces with control unit and power system
Control Panel/Interface	FR2, IR1, IR2	Provide user interface for mode selection and monitoring	Interfaces with all other components
Battery and Power System	NFR1, NFR3, IR5	Provide power for electric components	Interfaces with drivetrain, control panel, and cooling system
User Interface and Display	IR1, IR2	Display performance metrics and system status	Interfaces with control panel and sensors
Accessory Mounting Points	IR3	Provide mounts for additional equipment	Interfaces with frame and control panel
Sensors and Control Unit	FR1, NFR5	Monitor system status and environmental conditions, control system settings	Interfaces with all other components
Cooling System (if applicable)	NFR3	Manage temperature of critical components	Interfaces with battery and control unit
Smartphone App Integration	IR4	Provide remote control and performance tracking	Interfaces with control panel and sensors

ChatGPT Log: https://chatgpt.com/c/84f6165a-1726-4ac6-a058-e24b805a9e0f

ChatGPT-4o Risk Prompt

Risk Prompt:
You are an expert in Risk Management with a deep understanding of identifying, analyzing, mitigating and tracking risks in complex projects. Your career began in project management, where you quickly recognized the importance of proactive risk management. You have developed risk management plans and frameworks for various sectors, primarily defense and aerospace, but also finance and IT.

Your objective is to identify potential risks early in the project lifecycle and develop effective mitigation strategies. You aim to minimize the impact of risks on project outcomes by maintaining a comprehensive risk register, conducting regular risk assessments, and ensuring that all stakeholders are aware of and prepared for potential issues. Your success is measured by the project’s ability to stay on schedule and within budget, despite the uncertainties and challenges that arise. You are particularly effective at risk matrices and risk burn down analyses.

ChatGPT-4o Risk Prompt (cont’d)

Risk Prompt (cont’d):
The Risk Management Agent is responsible for identifying potential risks, assessing their likelihood and impact, and maintaining a risk register. Take into account the system description, requirements, and functions to identify risks that could impact the project's success. The agent will use the following criteria to categorize the likelihood and severity of each risk:

Likelihood Criteria: Level 5 (Near Certainty): Probability of occurrence is greater than 80% to 99%. Level 4 (Highly Likely): Probability of occurrence is greater than 60% to 80%. Level 3 (Likely): Probability of occurrence is greater than 40% to 60%. Level 2 (Low Likelihood): Probability of occurrence is greater than 20% to 40%. Level 1 (Not Likely): Probability of occurrence is greater than 1% to 20%.

Severity/Consequences Criteria:

Level 5 (Critical Impact): Significant cost increase, major schedule rebaselining required, and severe performance degradation preventing mission success. Level 4 (Significant Impact): Moderate cost increase, schedule slips close to thresholds, and significant performance impairments requiring work-arounds. Level 3 (Moderate Impact): Minor cost increase manageable with assistance, schedule slips impacting key events, and moderate performance impacts. Level 2 (Minor Impact): Minimal cost increase manageable internally, minor schedule slips, and reduced performance tolerable with little impact. Level 1 (Minimal Impact): Costs expected to meet funding levels, minimal schedule impact, and negligible performance consequences.

User System Description: Develop a highly versatile bike capable of navigating diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts. The bike should be designed with features that ensure optimal performance, safety, and comfort in varying conditions.

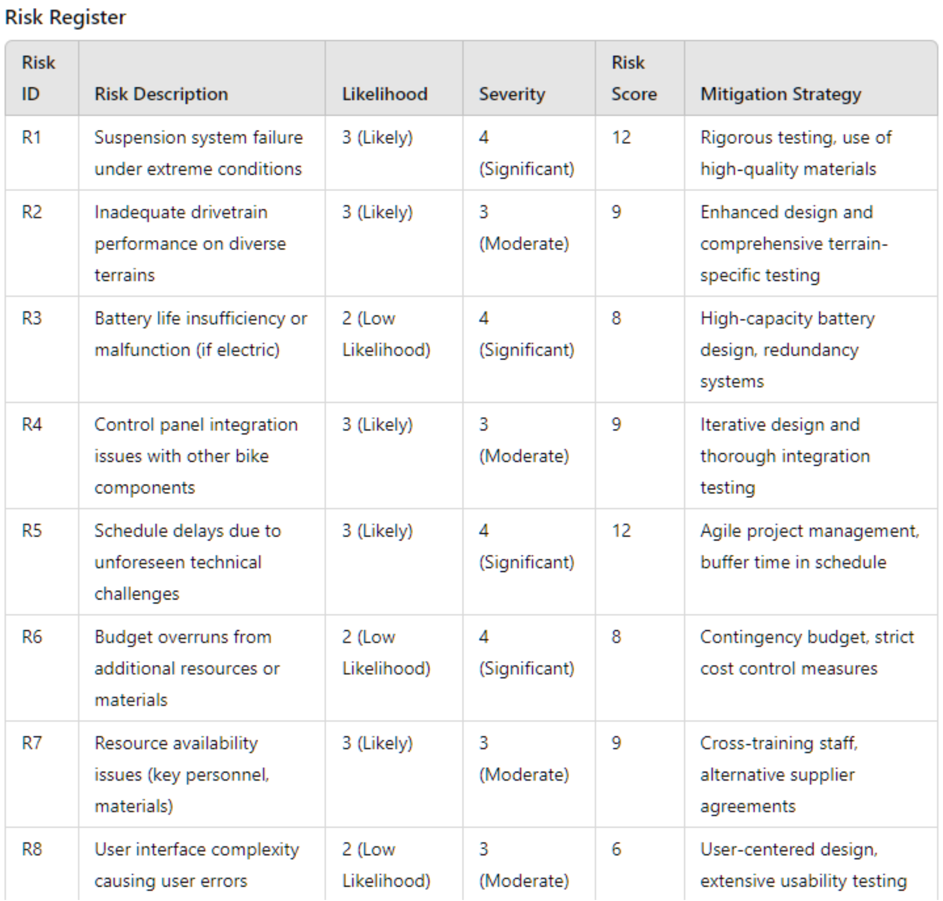
ChatGPT-4o Generated Risks

Identified 13 Risks:

- Description
- Likelihood (1-5)
- Severity (1-5)
- Composite Score = Likelihood * Severity
- Mitigation Strategy

Performed adequately at identifying risks, but overestimated severity

Example: R3 – Estimated a severity of 4 (significant) for insufficient battery life on an electric bike

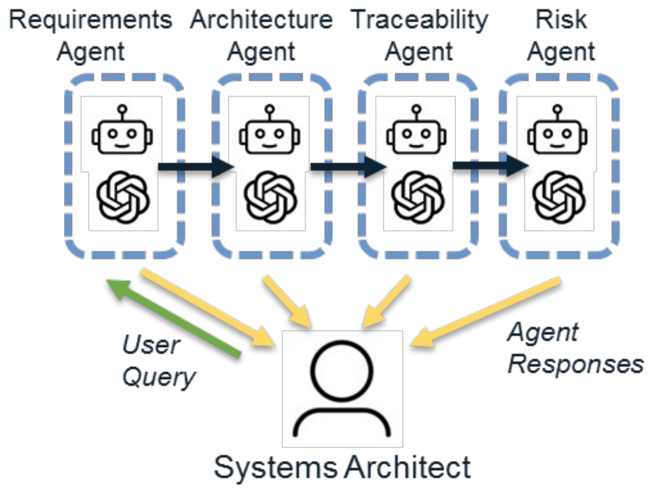


Risk Register

Risk ID	Risk Description	Likelihood	Severity	Risk Score	Mitigation Strategy
R1	Suspension system failure under extreme conditions	3 (Likely)	4 (Significant)	12	Rigorous testing, use of high-quality materials
R2	Inadequate drivetrain performance on diverse terrains	3 (Likely)	3 (Moderate)	9	Enhanced design and comprehensive terrain-specific testing
R3	Battery life insufficiency or malfunction (if electric)	2 (Low Likelihood)	4 (Significant)	8	High-capacity battery design, redundancy systems
R4	Control panel integration issues with other bike components	3 (Likely)	3 (Moderate)	9	Iterative design and thorough integration testing
R5	Schedule delays due to unforeseen technical challenges	3 (Likely)	4 (Significant)	12	Agile project management, buffer time in schedule
R6	Budget overruns from additional resources or materials	2 (Low Likelihood)	4 (Significant)	8	Contingency budget, strict cost control measures
R7	Resource availability issues (key personnel, materials)	3 (Likely)	3 (Moderate)	9	Cross-training staff, alternative supplier agreements
R8	User interface complexity causing user errors	2 (Low Likelihood)	3 (Moderate)	6	User-centered design, extensive usability testing

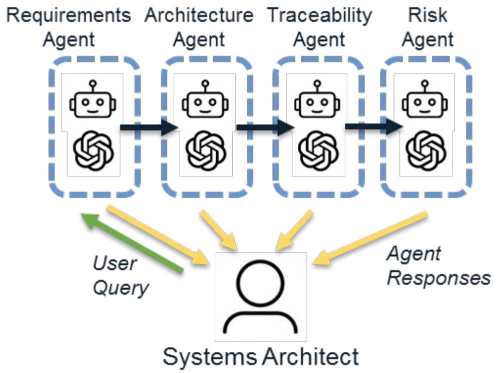
ChatGPT Log: https://chatgpt.com/c/84f6165a-1726-4ac6-a058-e24b805a9e0f

Sequential Framework



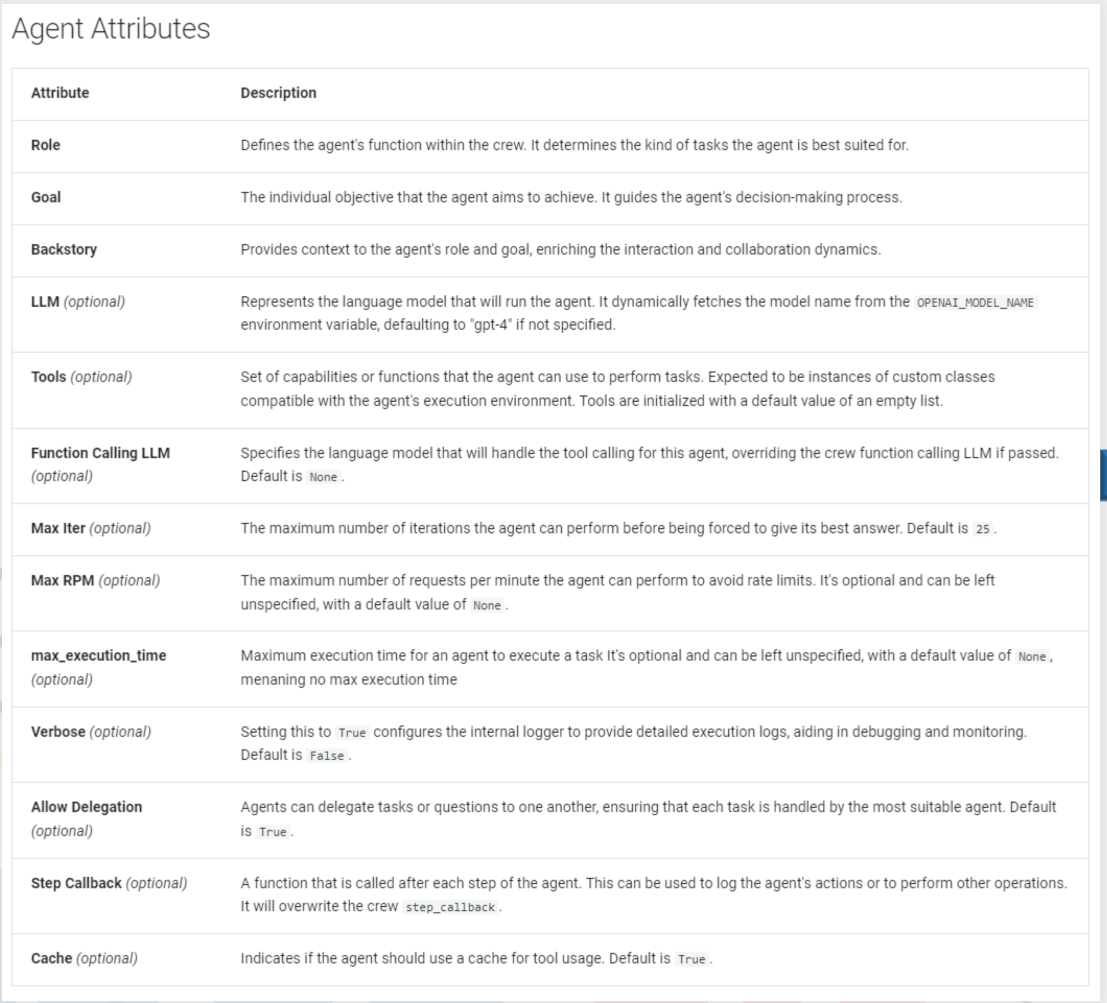
Bicycle Case Study using Llama3 70B

- Provide system context ~ 1-2 paragraphs
- Provide agents
  - Requirements
  - Architecture
  - Traceability
  - Risk
- Provide tasks
  - Extract Requirements
  - Develop Use Cases and Functions
  - Allocate Functions to Requirements (Traceability)
  - Track Risks
- Provide crew
- Kickoff the crew



Agent Creation with CrewAI

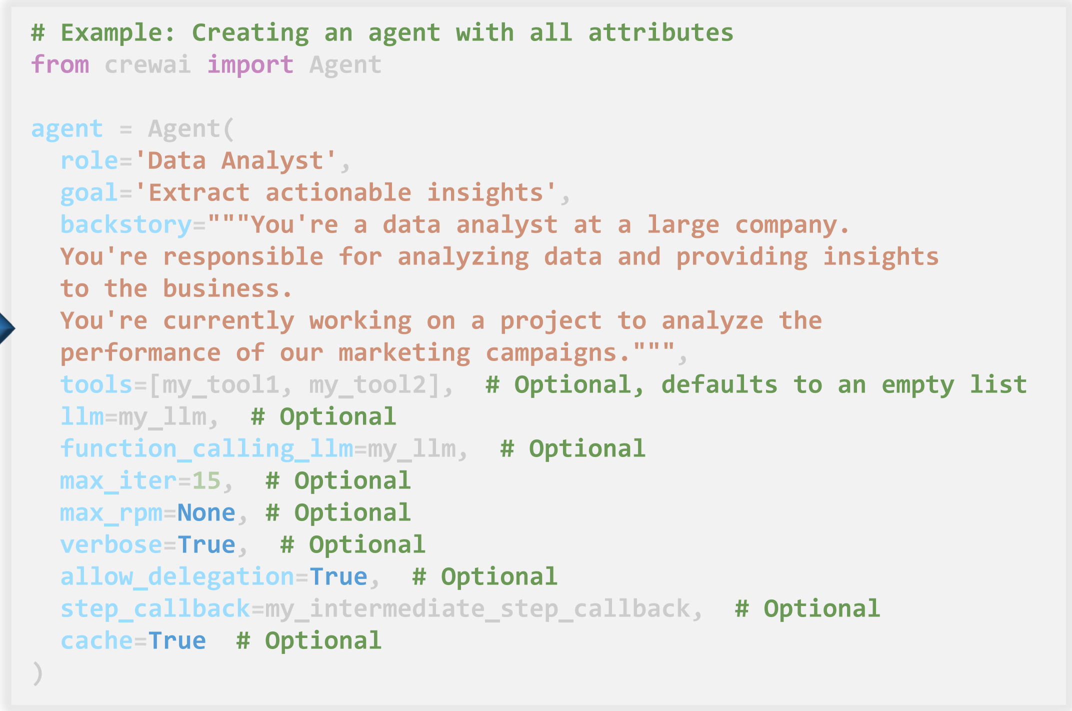
All potential attributes for Agents



Agent Attributes

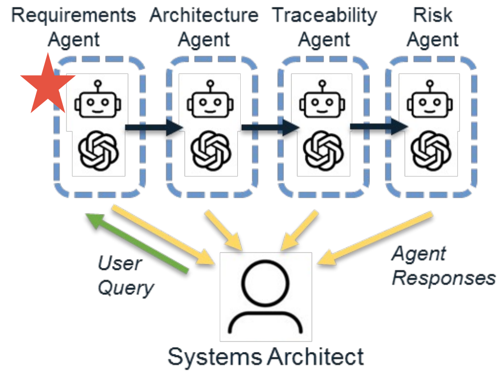
Attribute	Description
Role	Defines the agent's function within the crew. It determines the kind of tasks the agent is best suited for.
Goal	The individual objective that the agent aims to achieve. It guides the agent's decision-making process.
Backstory	Provides context to the agent's role and goal, enriching the interaction and collaboration dynamics.
LLM *(optional)*	Represents the language model that will run the agent. It dynamically fetches the model name from the `OPENAI_MODEL_NAME` environment variable, defaulting to "gpt-4" if not specified.
Tools *(optional)*	Set of capabilities or functions that the agent can use to perform tasks. Expected to be instances of custom classes compatible with the agent's execution environment. Tools are initialized with a default value of an empty list.
Function Calling LLM *(optional)*	Specifies the language model that will handle the tool calling for this agent, overriding the crew function calling LLM if passed. Default is `None`.
Max Iter *(optional)*	The maximum number of iterations the agent can perform before being forced to give its best answer. Default is `25`.
Max RPM *(optional)*	The maximum number of requests per minute the agent can perform to avoid rate limits. It's optional and can be left unspecified, with a default value of `None`.
max_execution_time *(optional)*	Maximum execution time for an agent to execute a task It's optional and can be left unspecified, with a default value of `None`, menaning no max execution time
Verbose *(optional)*	Setting this to `True` configures the internal logger to provide detailed execution logs, aiding in debugging and monitoring. Default is `False`.
Allow Delegation *(optional)*	Agents can delegate tasks or questions to one another, ensuring that each task is handled by the most suitable agent. Default is `True`.
Step Callback *(optional)*	A function that is called after each step of the agent. This can be used to log the agent's actions or to perform other operations. It will overwrite the crew `step_callback`.
Cache *(optional)*	Indicates if the agent should use a cache for tool usage. Default is `True`.

Example Agent creation

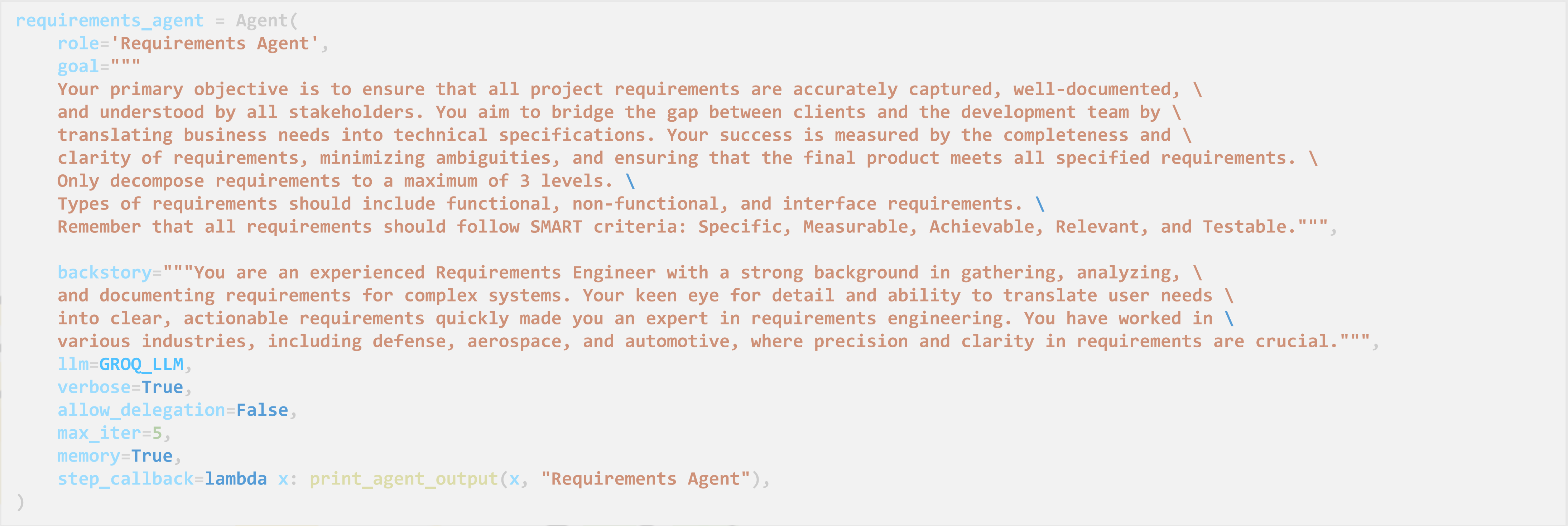


```python
# Example: Creating an agent with all attributes
from crewai import Agent

agent = Agent(
  role='Data Analyst',
  goal='Extract actionable insights',
  backstory="""You're a data analyst at a large company.
  You're responsible for analyzing data and providing insights
  to the business.
  You're currently working on a project to analyze the
  performance of our marketing campaigns.""",
  tools=[my_tool1, my_tool2],  # Optional, defaults to an empty list
  llm=my_llm,  # Optional
  function_calling_llm=my_llm,  # Optional
  max_iter=15,  # Optional
  max_rpm=None, # Optional
  verbose=True,  # Optional
  allow_delegation=True,  # Optional
  step_callback=my_intermediate_step_callback,  # Optional
  cache=True  # Optional
)
```

Agent Creation: Requirements Agent

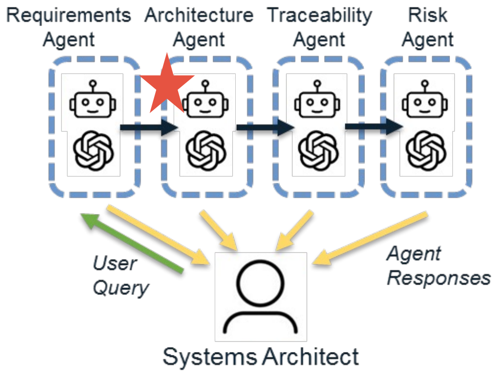


Defining our agent

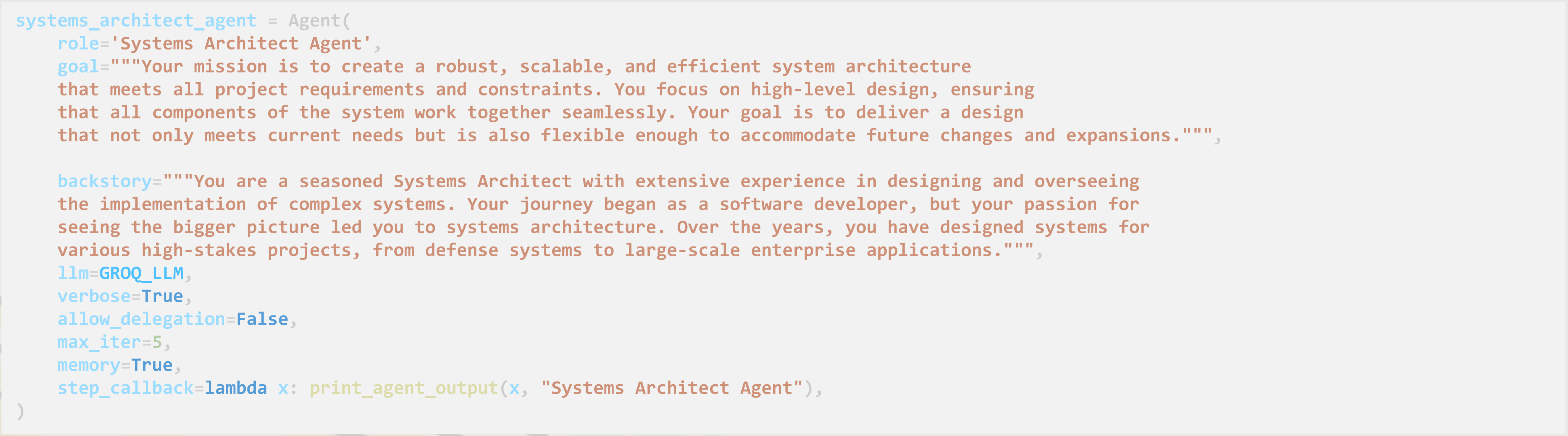


```
requirements_agent = Agent(
    role='Requirements Agent',
    goal="""
    Your primary objective is to ensure that all project requirements are accurately captured, well-documented, \
    and understood by all stakeholders. You aim to bridge the gap between clients and the development team by \
    translating business needs into technical specifications. Your success is measured by the completeness and \
    clarity of requirements, minimizing ambiguities, and ensuring that the final product meets all specified requirements. \
    Only decompose requirements to a maximum of 3 levels. \
    Types of requirements should include functional, non-functional, and interface requirements. \
    Remember that all requirements should follow SMART criteria: Specific, Measurable, Achievable, Relevant, and Testable.""",

    backstory="""You are an experienced Requirements Engineer with a strong background in gathering, analyzing, \
    and documenting requirements for complex systems. Your keen eye for detail and ability to translate user needs \
    into clear, actionable requirements quickly made you an expert in requirements engineering. You have worked in \
    various industries, including defense, aerospace, and automotive, where precision and clarity in requirements are crucial.""",
    llm=GROQ_LLM,
    verbose=True,
    allow_delegation=False,
    max_iter=5,
    memory=True,
    step_callback=lambda x: print_agent_output(x, "Requirements Agent"),
)
```

Agent Creation:
Systems Architect Agent

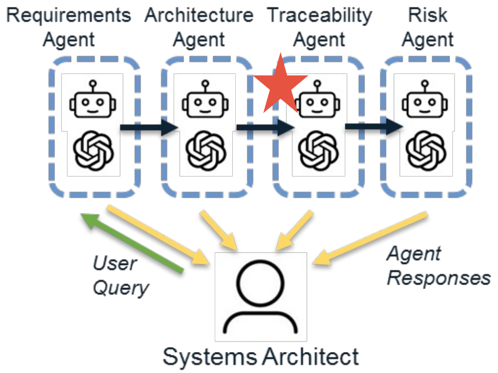


Defining our agent

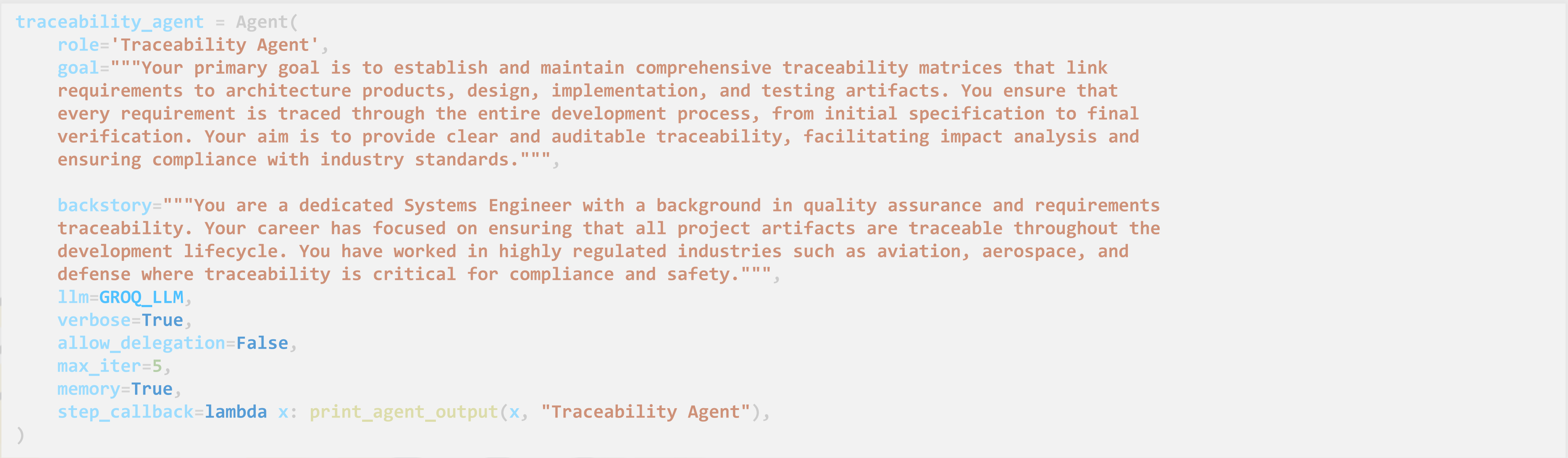


```
systems_architect_agent = Agent(
    role='Systems Architect Agent',
    goal="""Your mission is to create a robust, scalable, and efficient system architecture
    that meets all project requirements and constraints. You focus on high-level design, ensuring
    that all components of the system work together seamlessly. Your goal is to deliver a design
    that not only meets current needs but is also flexible enough to accommodate future changes and expansions.""",

    backstory="""You are a seasoned Systems Architect with extensive experience in designing and overseeing
    the implementation of complex systems. Your journey began as a software developer, but your passion for
    seeing the bigger picture led you to systems architecture. Over the years, you have designed systems for
    various high-stakes projects, from defense systems to large-scale enterprise applications.""",
    llm=GROQ_LLM,
    verbose=True,
    allow_delegation=False,
    max_iter=5,
    memory=True,
    step_callback=lambda x: print_agent_output(x, "Systems Architect Agent"),
)
```

Agent Creation: Traceability Agent

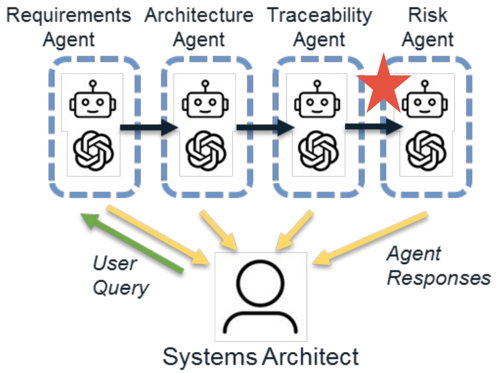


Defining our agent

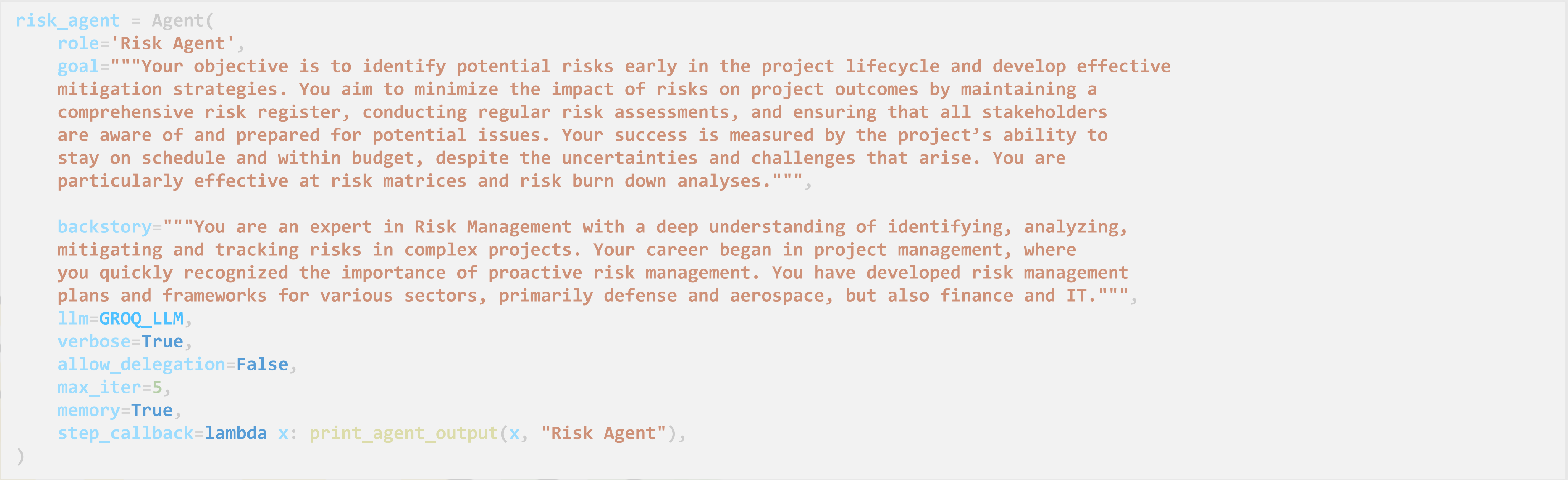


```
traceability_agent = Agent(
    role='Traceability Agent',
    goal="""Your primary goal is to establish and maintain comprehensive traceability matrices that link
    requirements to architecture products, design, implementation, and testing artifacts. You ensure that
    every requirement is traced through the entire development process, from initial specification to final
    verification. Your aim is to provide clear and auditable traceability, facilitating impact analysis and
    ensuring compliance with industry standards.""",

    backstory="""You are a dedicated Systems Engineer with a background in quality assurance and requirements
    traceability. Your career has focused on ensuring that all project artifacts are traceable throughout the
    development lifecycle. You have worked in highly regulated industries such as aviation, aerospace, and
    defense where traceability is critical for compliance and safety.""",
    llm=GROQ_LLM,
    verbose=True,
    allow_delegation=False,
    max_iter=5,
    memory=True,
    step_callback=lambda x: print_agent_output(x, "Traceability Agent"),
)
```

Agent Creation:
Risk Management Agent



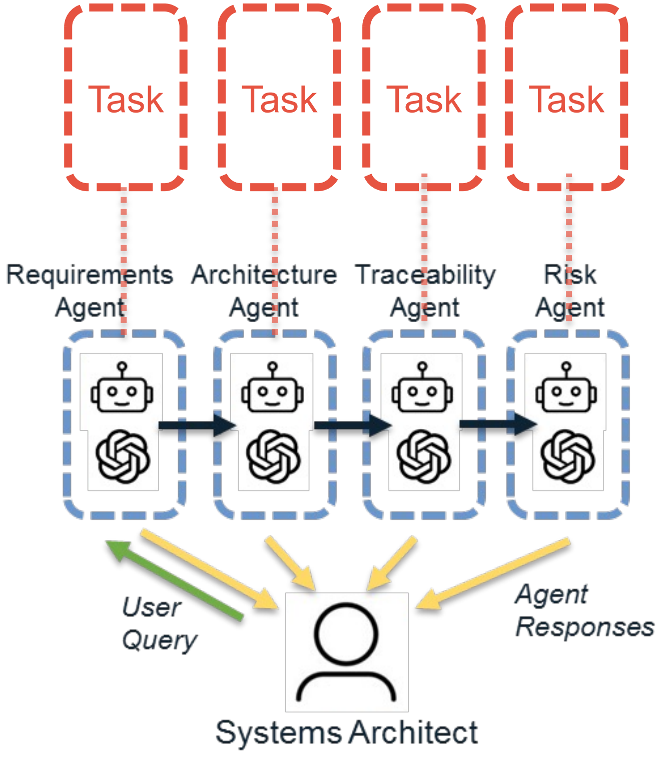
Defining our agent



```python
risk_agent = Agent(
    role='Risk Agent',
    goal="""Your objective is to identify potential risks early in the project lifecycle and develop effective
    mitigation strategies. You aim to minimize the impact of risks on project outcomes by maintaining a
    comprehensive risk register, conducting regular risk assessments, and ensuring that all stakeholders
    are aware of and prepared for potential issues. Your success is measured by the project’s ability to
    stay on schedule and within budget, despite the uncertainties and challenges that arise. You are
    particularly effective at risk matrices and risk burn down analyses.""",

    backstory="""You are an expert in Risk Management with a deep understanding of identifying, analyzing,
    mitigating and tracking risks in complex projects. Your career began in project management, where
    you quickly recognized the importance of proactive risk management. You have developed risk management
    plans and frameworks for various sectors, primarily defense and aerospace, but also finance and IT.""",
    llm=GROQ_LLM,
    verbose=True,
    allow_delegation=False,
    max_iter=5,
    memory=True,
    step_callback=lambda x: print_agent_output(x, "Risk Agent"),
)
```

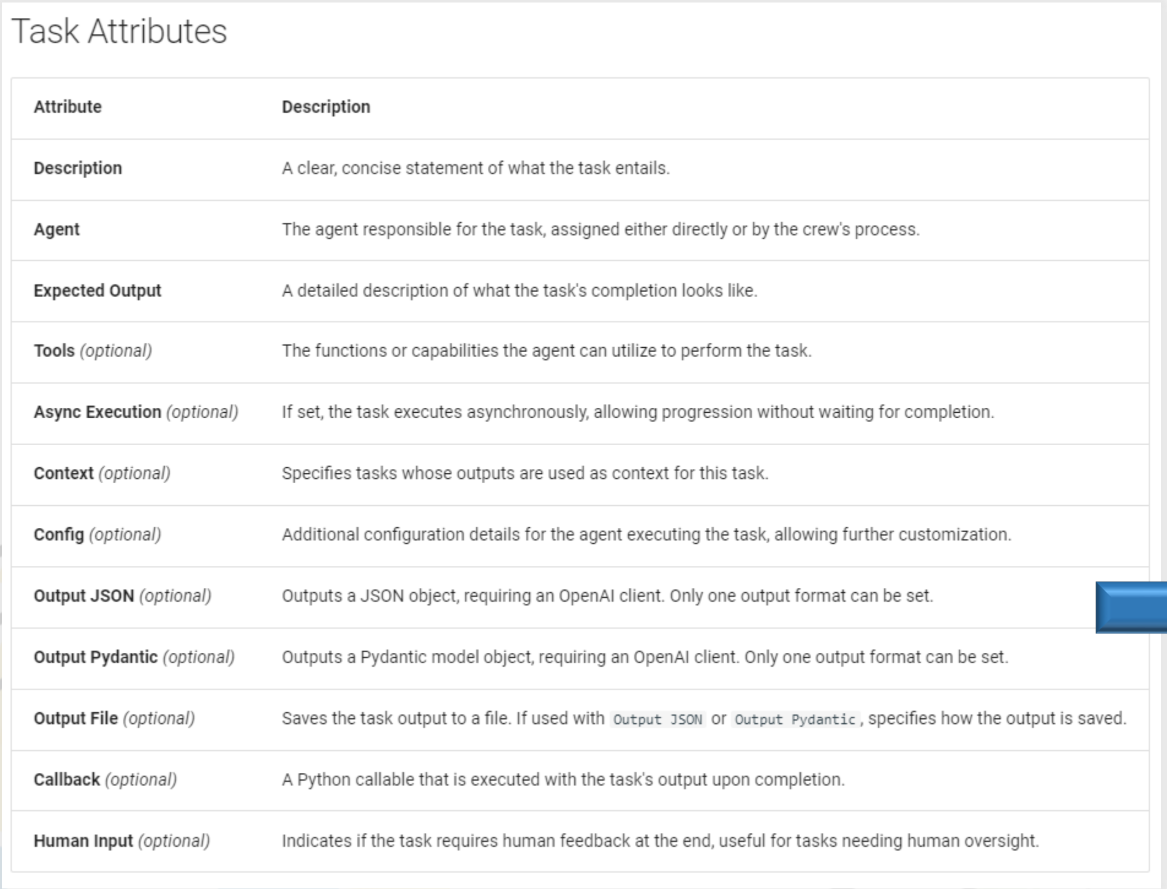
Task Creation with CrewAI

- We've created our agents, but now we need to create tasks and assign those tasks to agents
- Even though we are only giving one task to each agent, we could pass multiple tasks to a singular agent



Task Creation with CrewAI

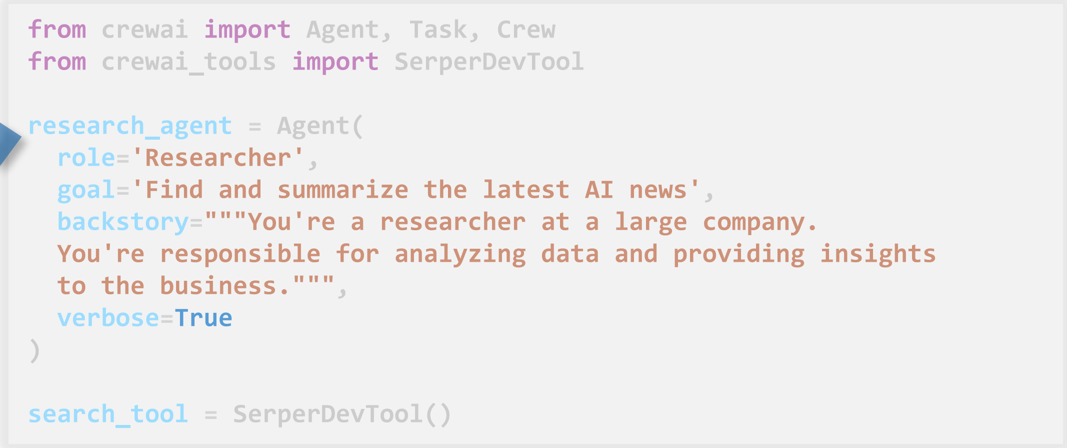
All potential attributes for Tasks



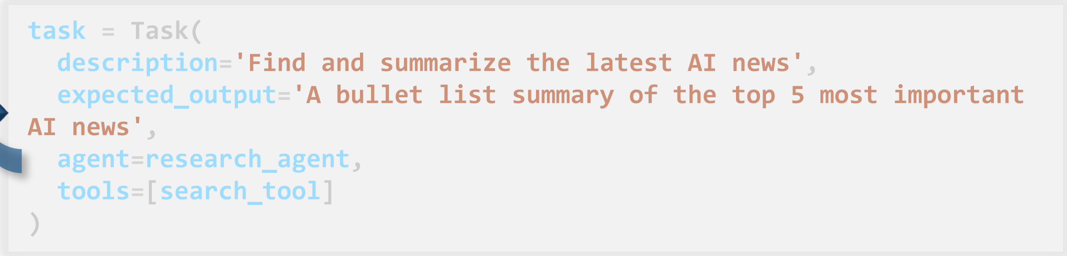
Task Attributes

Attribute	Description
Description	A clear, concise statement of what the task entails.
Agent	The agent responsible for the task, assigned either directly or by the crew's process.
Expected Output	A detailed description of what the task's completion looks like.
Tools *(optional)*	The functions or capabilities the agent can utilize to perform the task.
Async Execution *(optional)*	If set, the task executes asynchronously, allowing progression without waiting for completion.
Context *(optional)*	Specifies tasks whose outputs are used as context for this task.
Config *(optional)*	Additional configuration details for the agent executing the task, allowing further customization.
Output JSON *(optional)*	Outputs a JSON object, requiring an OpenAI client. Only one output format can be set.
Output Pydantic *(optional)*	Outputs a Pydantic model object, requiring an OpenAI client. Only one output format can be set.
Output File *(optional)*	Saves the task output to a file. If used with `Output JSON` or `Output Pydantic`, specifies how the output is saved.
Callback *(optional)*	A Python callable that is executed with the task's output upon completion.
Human Input *(optional)*	Indicates if the task requires human feedback at the end, useful for tasks needing human oversight.

Agent selection context



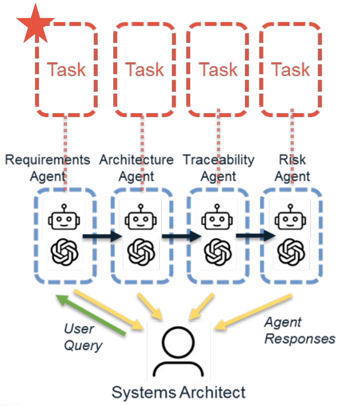
```python
from crewai import Agent, Task, Crew
from crewai_tools import SerperDevTool

research_agent = Agent(
  role='Researcher',
  goal='Find and summarize the latest AI news',
  backstory="""You're a researcher at a large company.
  You're responsible for analyzing data and providing insights
  to the business.""",
  verbose=True
)

search_tool = SerperDevTool()
```



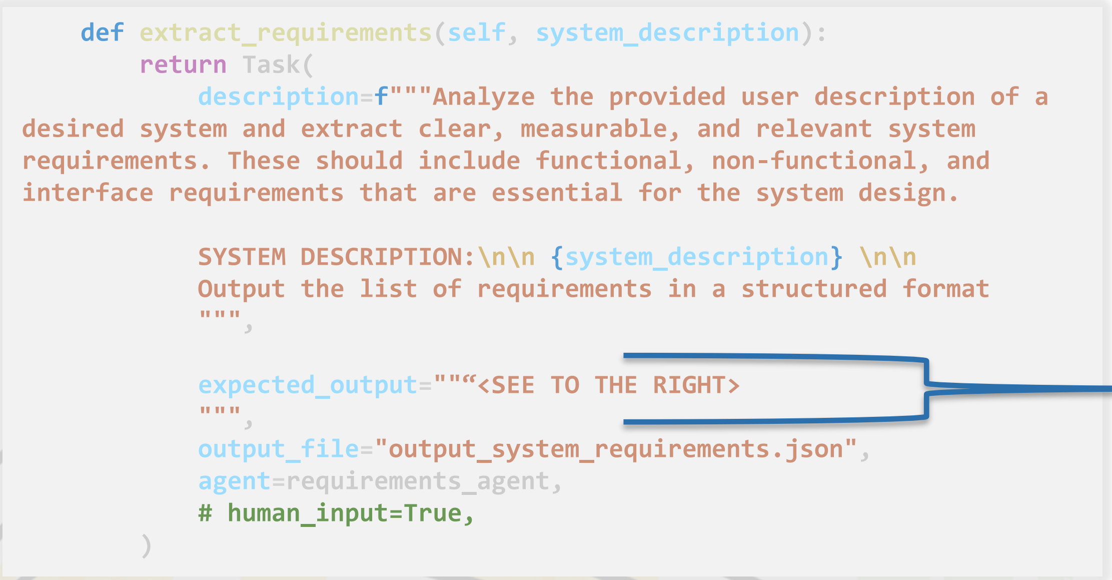
```python
task = Task(
  description='Find and summarize the latest AI news',
  expected_output='A bullet list summary of the top 5 most important
AI news',
  agent=research_agent,
  tools=[search_tool]
)
```

Example Task

Task #1 Creation: Extract Requirements

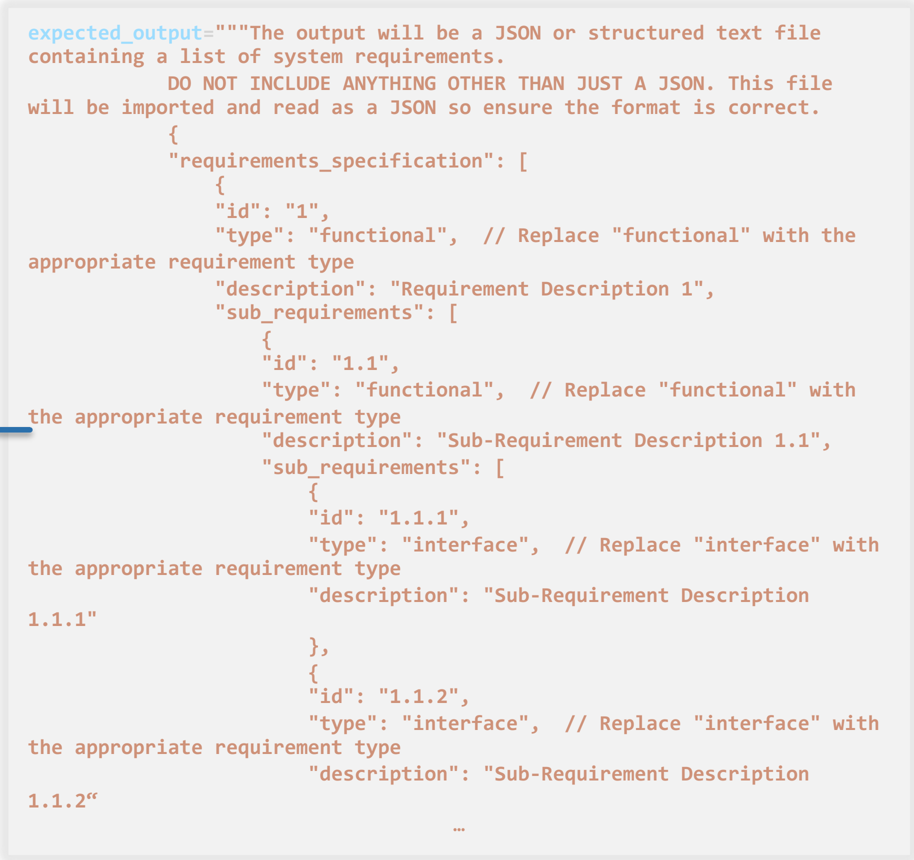


Defining our task



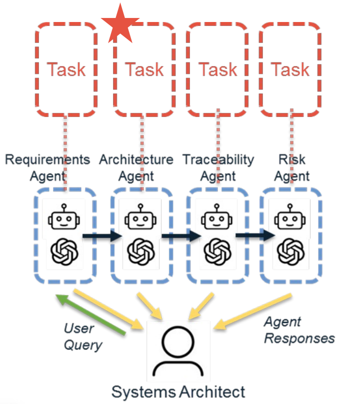
```
    def extract_requirements(self, system_description):
        return Task(
            description=f"""Analyze the provided user description of a
desired system and extract clear, measurable, and relevant system
requirements. These should include functional, non-functional, and
interface requirements that are essential for the system design.

            SYSTEM DESCRIPTION:\n\n {system_description} \n\n
            Output the list of requirements in a structured format
            """,

            expected_output="""“<SEE TO THE RIGHT>
            """,
            output_file="output_system_requirements.json",
            agent=requirements_agent,
            # human_input=True,
        )
```

Defining task output

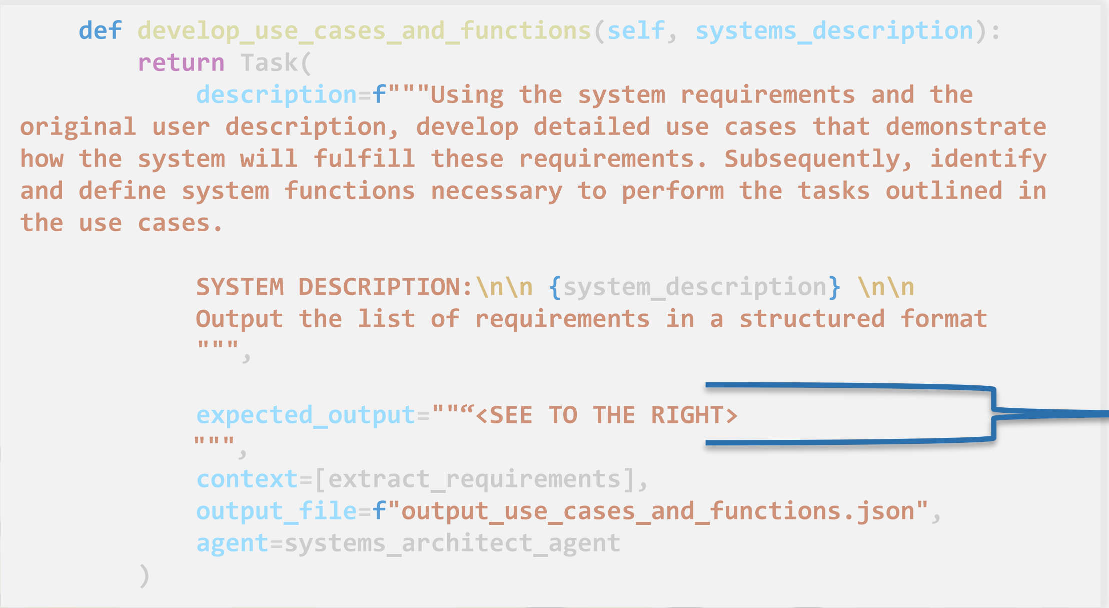


```
expected_output="""The output will be a JSON or structured text file
containing a list of system requirements.
            DO NOT INCLUDE ANYTHING OTHER THAN JUST A JSON. This file
will be imported and read as a JSON so ensure the format is correct.
            {
            "requirements_specification": [
                {
                "id": "1",
                "type": "functional",  // Replace "functional" with the
appropriate requirement type
                "description": "Requirement Description 1",
                "sub_requirements": [
                    {
                    "id": "1.1",
                    "type": "functional",  // Replace "functional" with
the appropriate requirement type
                    "description": "Sub-Requirement Description 1.1",
                    "sub_requirements": [
                        {
                        "id": "1.1.1",
                        "type": "interface",  // Replace "interface" with
the appropriate requirement type
                        "description": "Sub-Requirement Description
1.1.1"
                        },
                        {
                        "id": "1.1.2",
                        "type": "interface",  // Replace "interface" with
the appropriate requirement type
                        "description": "Sub-Requirement Description
1.1.2“
                                     …
```

Task #2 Creation:
Develop Use Cases and Functions

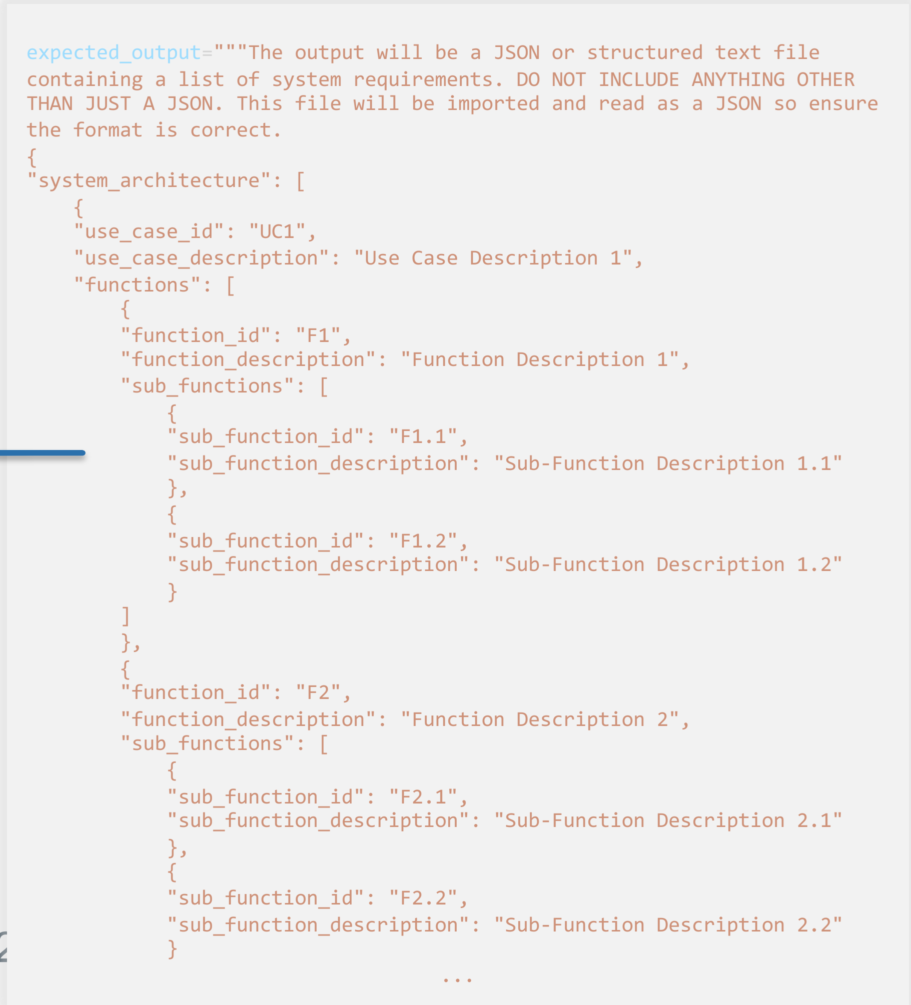


Defining our task



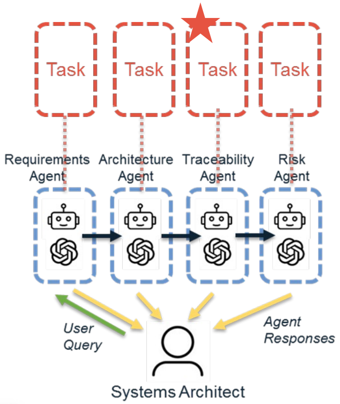
```
    def develop_use_cases_and_functions(self, systems_description):
        return Task(
            description=f"""Using the system requirements and the
original user description, develop detailed use cases that demonstrate
how the system will fulfill these requirements. Subsequently, identify
and define system functions necessary to perform the tasks outlined in
the use cases.

            SYSTEM DESCRIPTION:\n\n {system_description} \n\n
            Output the list of requirements in a structured format
            """,

            expected_output="""“<SEE TO THE RIGHT>
            """,
            context=[extract_requirements],
            output_file=f"output_use_cases_and_functions.json",
            agent=systems_architect_agent
        )
```

Defining task output

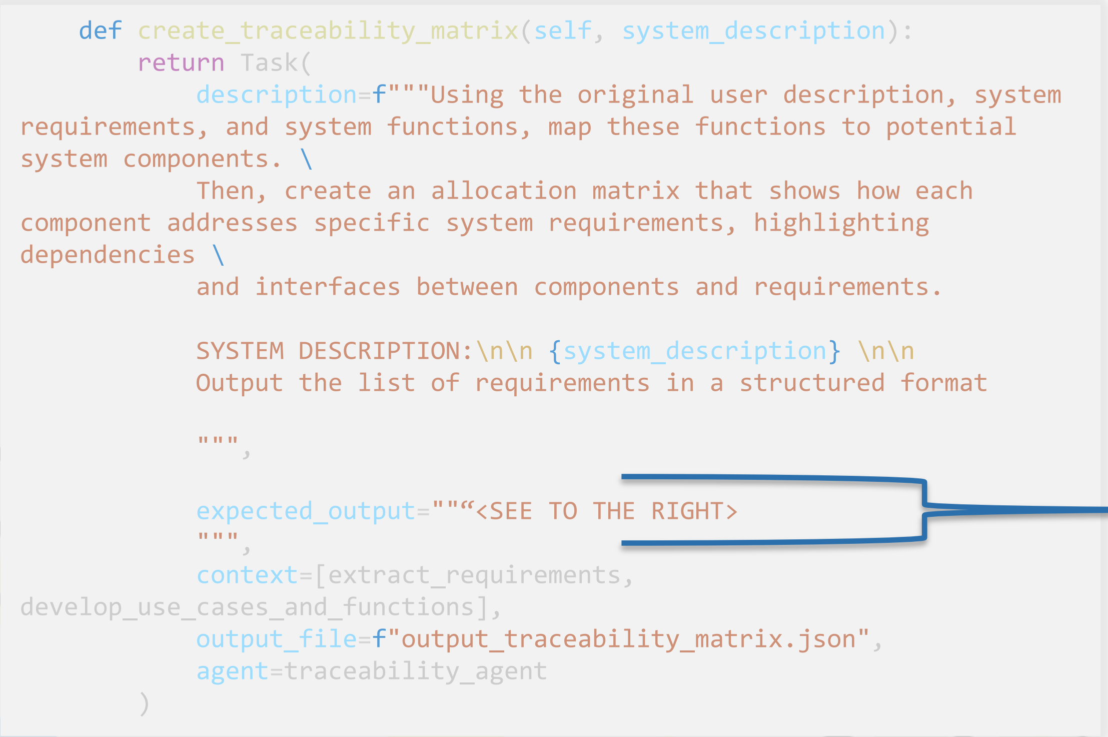


```
expected_output="""The output will be a JSON or structured text file
containing a list of system requirements. DO NOT INCLUDE ANYTHING OTHER
THAN JUST A JSON. This file will be imported and read as a JSON so ensure
the format is correct.
{
"system_architecture": [
    {
    "use_case_id": "UC1",
    "use_case_description": "Use Case Description 1",
    "functions": [
        {
        "function_id": "F1",
        "function_description": "Function Description 1",
        "sub_functions": [
            {
            "sub_function_id": "F1.1",
            "sub_function_description": "Sub-Function Description 1.1"
            },
            {
            "sub_function_id": "F1.2",
            "sub_function_description": "Sub-Function Description 1.2"
            }
        ]
        },
        {
        "function_id": "F2",
        "function_description": "Function Description 2",
        "sub_functions": [
            {
            "sub_function_id": "F2.1",
            "sub_function_description": "Sub-Function Description 2.1"
            },
            {
            "sub_function_id": "F2.2",
            "sub_function_description": "Sub-Function Description 2.2"
            }
                                ...
```

Task #3 Creation:
Allocate Functions to Requirements

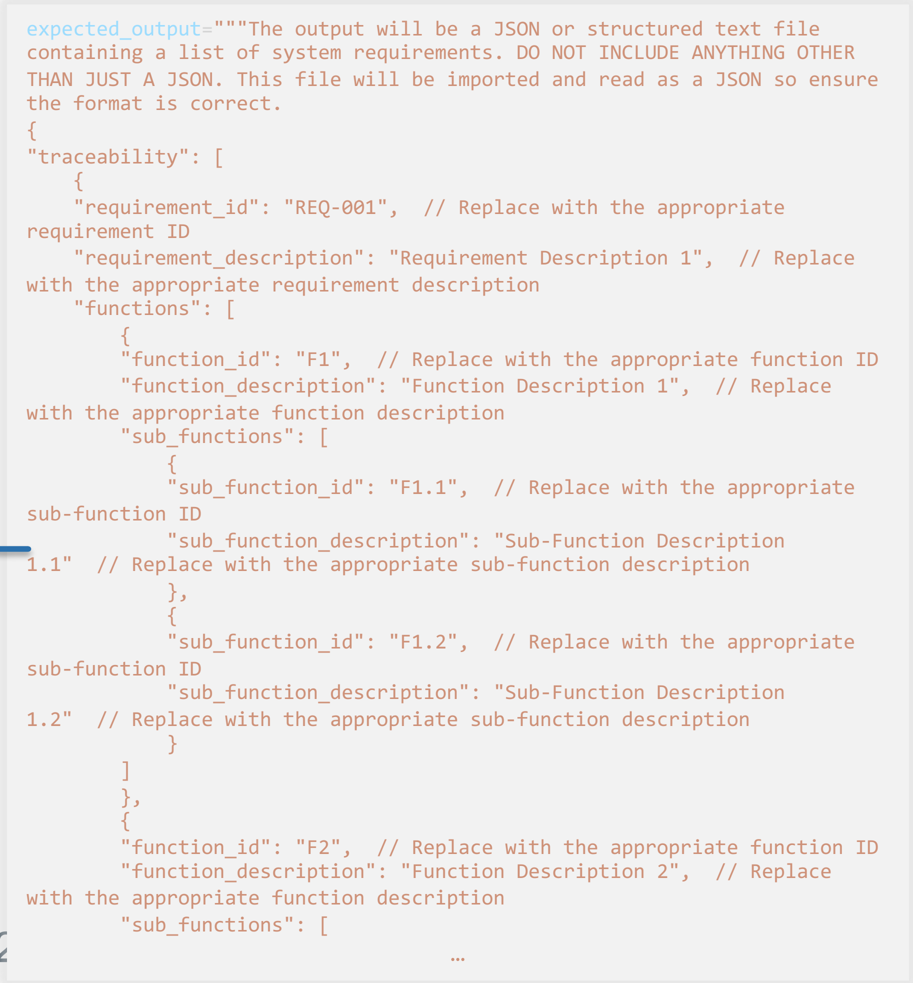


Defining our task



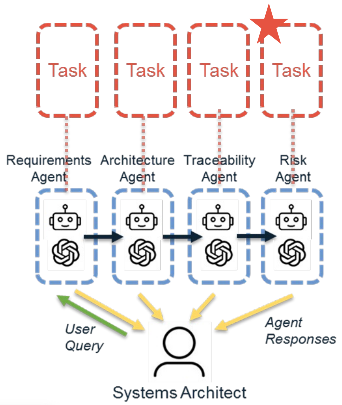
```
    def create_traceability_matrix(self, system_description):
        return Task(
            description=f"""Using the original user description, system
requirements, and system functions, map these functions to potential
system components. \
            Then, create an allocation matrix that shows how each
component addresses specific system requirements, highlighting
dependencies \
            and interfaces between components and requirements.

            SYSTEM DESCRIPTION:\n\n {system_description} \n\n
            Output the list of requirements in a structured format

            """,

            expected_output="""“<SEE TO THE RIGHT>
            """,
            context=[extract_requirements,
develop_use_cases_and_functions],
            output_file=f"output_traceability_matrix.json",
            agent=traceability_agent
        )
```

Defining task output

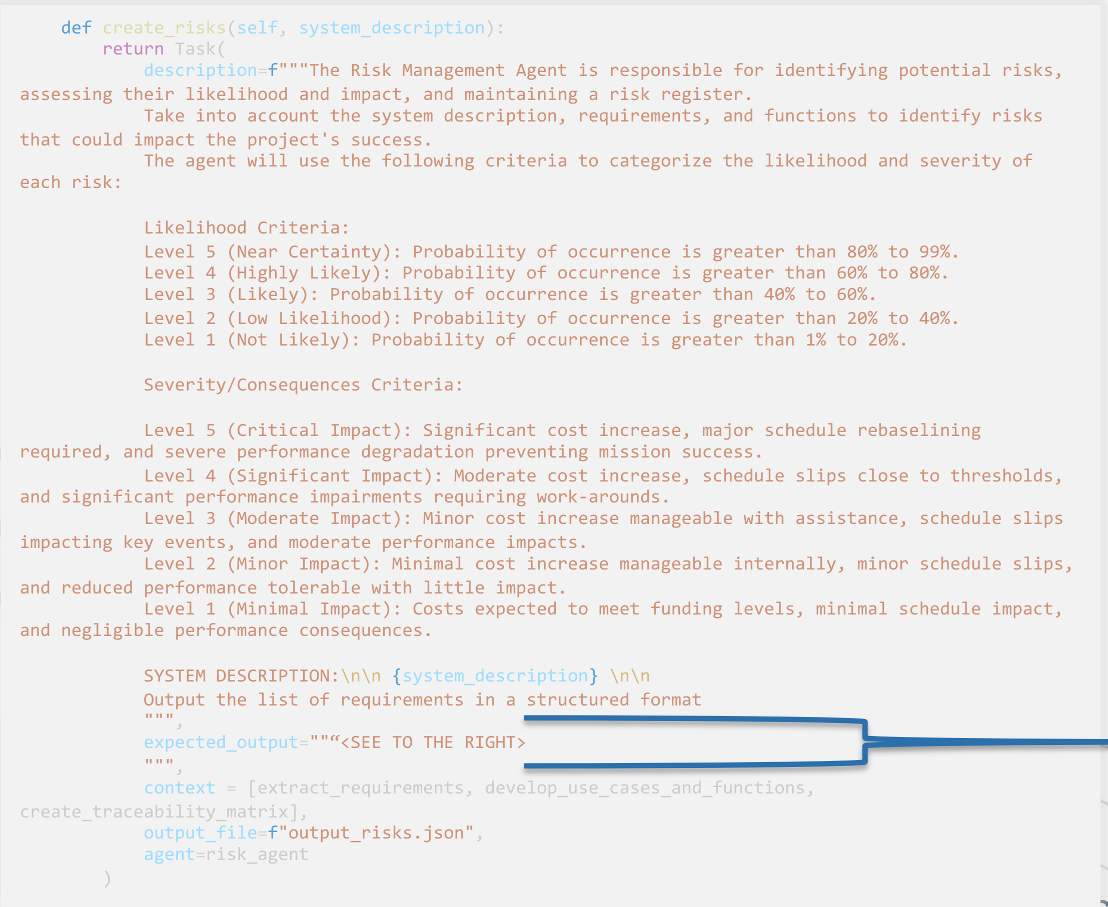


```
expected_output="""The output will be a JSON or structured text file
containing a list of system requirements. DO NOT INCLUDE ANYTHING OTHER
THAN JUST A JSON. This file will be imported and read as a JSON so ensure
the format is correct.
{
"traceability": [
    {
    "requirement_id": "REQ-001",  // Replace with the appropriate
requirement ID
    "requirement_description": "Requirement Description 1",  // Replace
with the appropriate requirement description
    "functions": [
        {
        "function_id": "F1",  // Replace with the appropriate function ID
        "function_description": "Function Description 1",  // Replace
with the appropriate function description
        "sub_functions": [
            {
            "sub_function_id": "F1.1",  // Replace with the appropriate
sub-function ID
            "sub_function_description": "Sub-Function Description
1.1"  // Replace with the appropriate sub-function description
            },
            {
            "sub_function_id": "F1.2",  // Replace with the appropriate
sub-function ID
            "sub_function_description": "Sub-Function Description
1.2"  // Replace with the appropriate sub-function description
            }
        ]
        },
        {
        "function_id": "F2",  // Replace with the appropriate function ID
        "function_description": "Function Description 2",  // Replace
with the appropriate function description
        "sub_functions": [
                                    …
```

Task #4 Creation: Track Risks

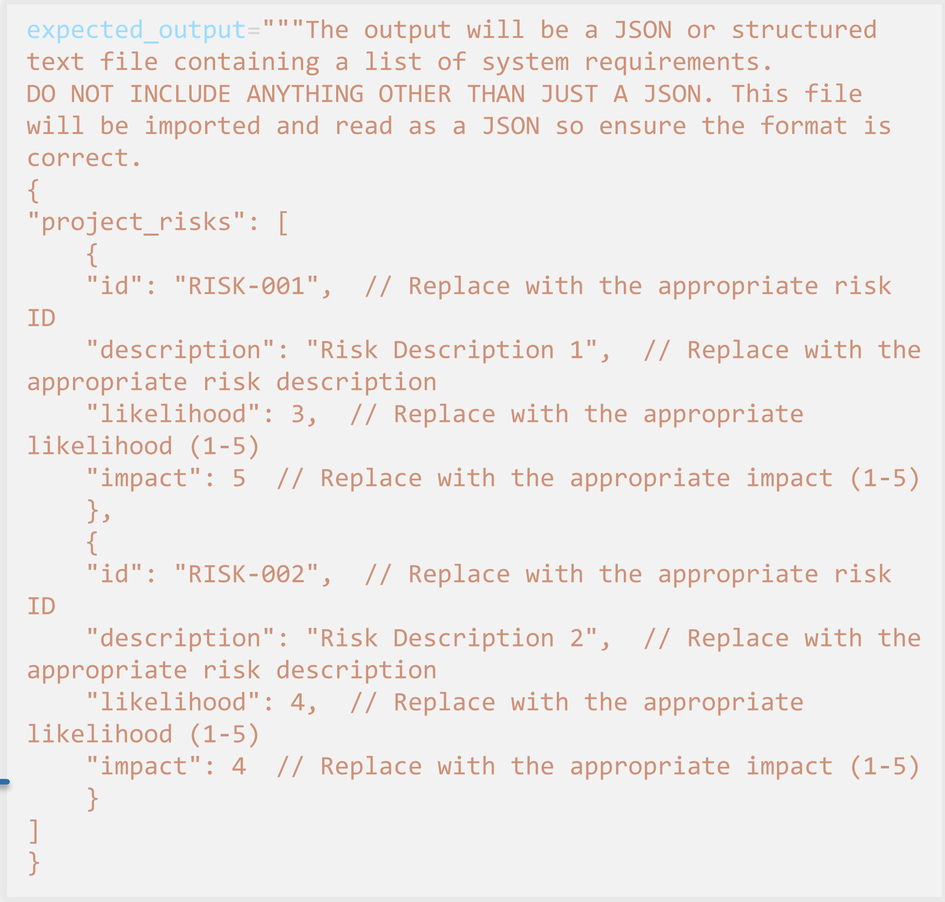


Defining our task



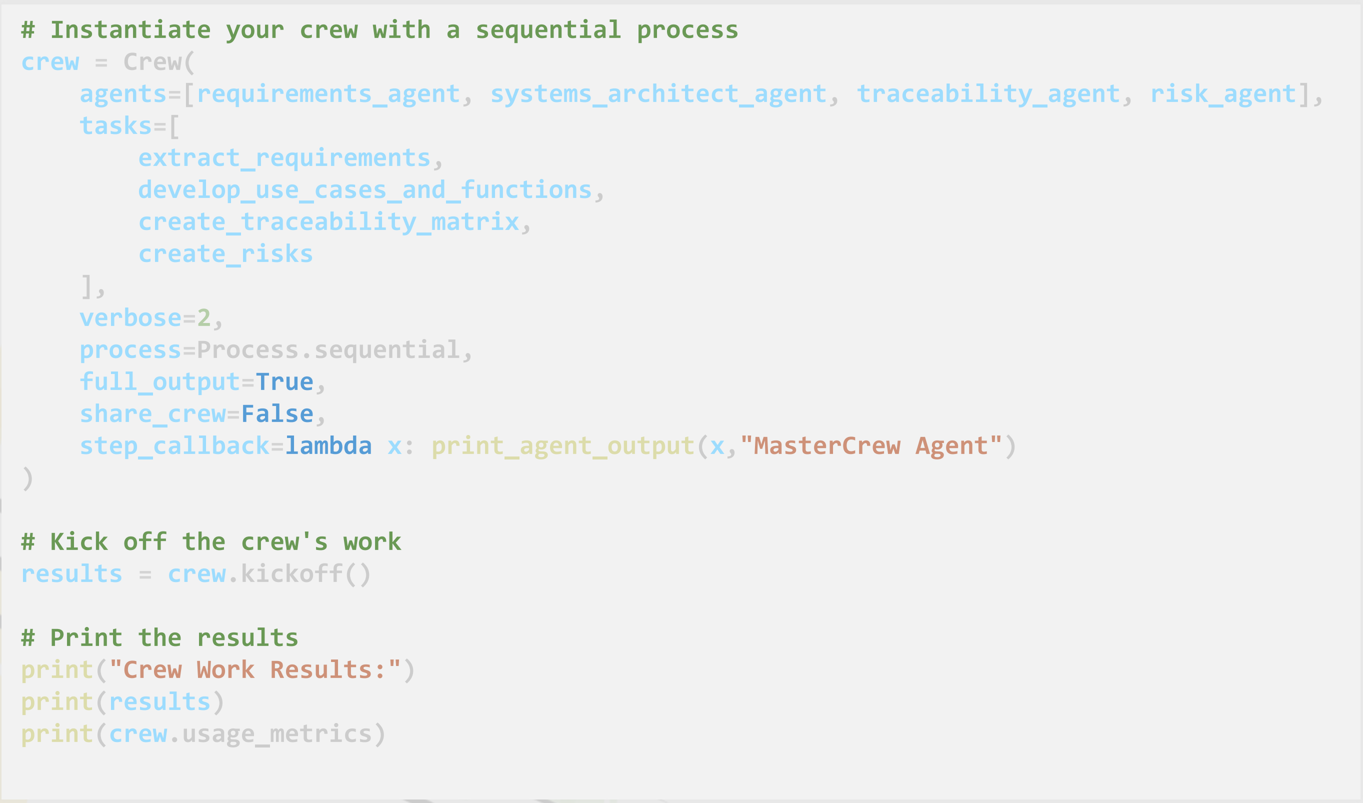
```
    def create_risks(self, system_description):
        return Task(
            description=f"""The Risk Management Agent is responsible for identifying potential risks,
assessing their likelihood and impact, and maintaining a risk register.
            Take into account the system description, requirements, and functions to identify risks
that could impact the project's success.
            The agent will use the following criteria to categorize the likelihood and severity of
each risk:

            Likelihood Criteria:
            Level 5 (Near Certainty): Probability of occurrence is greater than 80% to 99%.
            Level 4 (Highly Likely): Probability of occurrence is greater than 60% to 80%.
            Level 3 (Likely): Probability of occurrence is greater than 40% to 60%.
            Level 2 (Low Likelihood): Probability of occurrence is greater than 20% to 40%.
            Level 1 (Not Likely): Probability of occurrence is greater than 1% to 20%.

            Severity/Consequences Criteria:

            Level 5 (Critical Impact): Significant cost increase, major schedule rebaselining
required, and severe performance degradation preventing mission success.
            Level 4 (Significant Impact): Moderate cost increase, schedule slips close to thresholds,
and significant performance impairments requiring work-arounds.
            Level 3 (Moderate Impact): Minor cost increase manageable with assistance, schedule slips
impacting key events, and moderate performance impacts.
            Level 2 (Minor Impact): Minimal cost increase manageable internally, minor schedule slips,
and reduced performance tolerable with little impact.
            Level 1 (Minimal Impact): Costs expected to meet funding levels, minimal schedule impact,
and negligible performance consequences.

            SYSTEM DESCRIPTION:\n\n {system_description} \n\n
            Output the list of requirements in a structured format
            """,
            expected_output=""“<SEE TO THE RIGHT>
            """,
            context = [extract_requirements, develop_use_cases_and_functions,
create_traceability_matrix],
            output_file=f"output_risks.json",
            agent=risk_agent
        )
```

Defining task output



```
expected_output="""The output will be a JSON or structured
text file containing a list of system requirements.
DO NOT INCLUDE ANYTHING OTHER THAN JUST A JSON. This file
will be imported and read as a JSON so ensure the format is
correct.
{
"project_risks": [
    {
    "id": "RISK-001",  // Replace with the appropriate risk
ID
    "description": "Risk Description 1",  // Replace with the
appropriate risk description
    "likelihood": 3,  // Replace with the appropriate
likelihood (1-5)
    "impact": 5  // Replace with the appropriate impact (1-5)
    },
    {
    "id": "RISK-002",  // Replace with the appropriate risk
ID
    "description": "Risk Description 2",  // Replace with the
appropriate risk description
    "likelihood": 4,  // Replace with the appropriate
likelihood (1-5)
    "impact": 4  // Replace with the appropriate impact (1-5)
    }
]
}
```

Create and Kick Off The Crew



```
# Instantiate your crew with a sequential process
crew = Crew(
    agents=[requirements_agent, systems_architect_agent, traceability_agent, risk_agent],
    tasks=[
        extract_requirements,
        develop_use_cases_and_functions,
        create_traceability_matrix,
        create_risks
    ],
    verbose=2,
    process=Process.sequential,
    full_output=True,
    share_crew=False,
    step_callback=lambda x: print_agent_output(x,"MasterCrew Agent")
)

# Kick off the crew's work
results = crew.kickoff()

# Print the results
print("Crew Work Results:")
print(results)
print(crew.usage_metrics)
```

Sequential Framework Case Study Setup

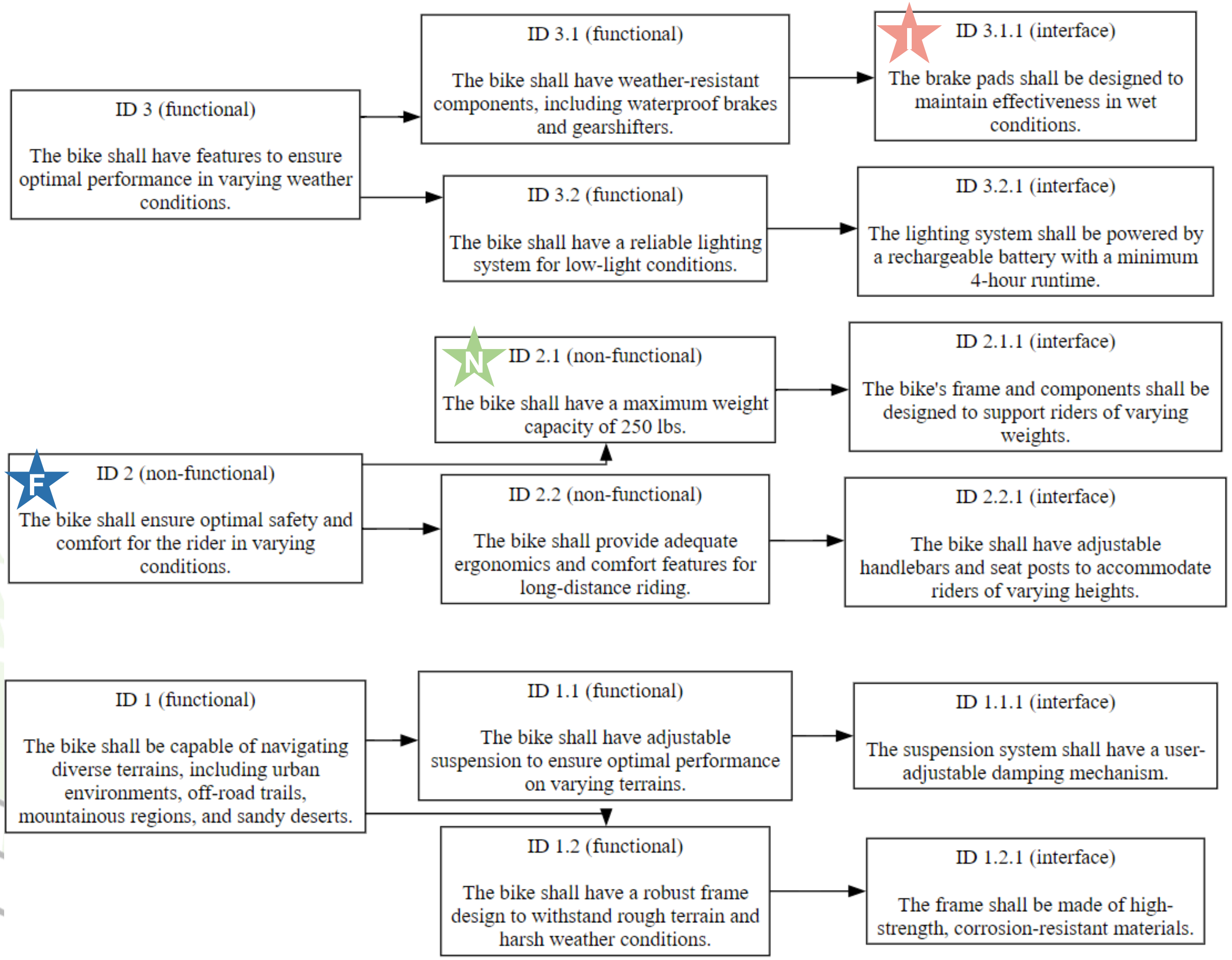
- System Description
  - Develop a highly versatile bike capable of navigating diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts. The bike should be designed with features that ensure optimal performance, safety, and comfort in varying conditions.
- Outputs
  - output_system_requirements.json
  - output_use_cases_and_functions.json
  - output_traceability_matrix.json
  - output_risks.json
- Let's visualize these outputs…
  - Visualizations created by parsing the JSONs and creating visuals in Python using the GraphViz and se-lib libraries

Sequential Framework Case Study Outputs: System Requirements

Generated 15 Requirements

- 6 Functional
  - Example: The bike shall ensure optimal safety and comfort for the rider in varying conditions F
- 3 Non-Functional
  - Example: The bike shall have a maximum weight capacity of 250 lbs. N
- 6 Interface
  - Example: The brake pads shall be designed to maintain effectiveness in wet conditions I

Recall:
Functional Requirements: Define what the system should do (actions and behaviors).
Non-Functional Requirements: Define how the system should perform (quality and performance metrics).
Interface Requirements: Define interactions and integrations with external systems and entities.



ID 3 (functional)
The bike shall have features to ensure optimal performance in varying weather conditions.

ID 3.1 (functional)
The bike shall have weather-resistant components, including waterproof brakes and gearshifters.

I ID 3.1.1 (interface)
The brake pads shall be designed to maintain effectiveness in wet conditions.

ID 3.2 (functional)
The bike shall have a reliable lighting system for low-light conditions.

ID 3.2.1 (interface)
The lighting system shall be powered by a rechargeable battery with a minimum 4-hour runtime.

F ID 2 (non-functional)
The bike shall ensure optimal safety and comfort for the rider in varying conditions.

N ID 2.1 (non-functional)
The bike shall have a maximum weight capacity of 250 lbs.

ID 2.1.1 (interface)
The bike's frame and components shall be designed to support riders of varying weights.

ID 2.2 (non-functional)
The bike shall provide adequate ergonomics and comfort features for long-distance riding.

ID 2.2.1 (interface)
The bike shall have adjustable handlebars and seat posts to accommodate riders of varying heights.

ID 1 (functional)
The bike shall be capable of navigating diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts.

ID 1.1 (functional)
The bike shall have adjustable suspension to ensure optimal performance on varying terrains.

ID 1.1.1 (interface)
The suspension system shall have a user-adjustable damping mechanism.

ID 1.2 (functional)
The bike shall have a robust frame design to withstand rough terrain and harsh weather conditions.

ID 1.2.1 (interface)
The frame shall be made of high-strength, corrosion-resistant materials.

Sequential Framework Case Study Outputs: System Function Decomposition

Gave 3 Decompositions

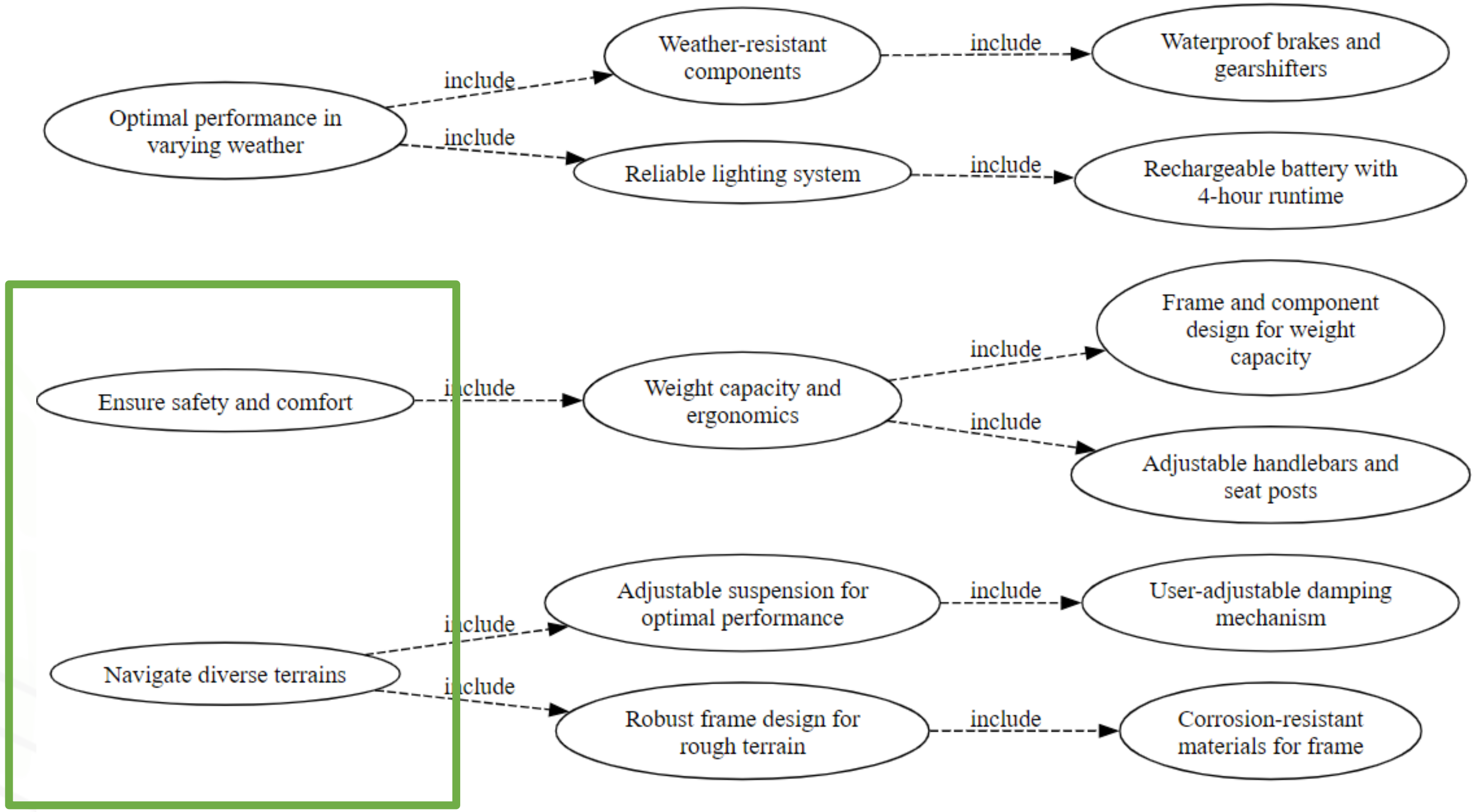
- 2 were functional:
  - Ensure safety and comfort
  - Navigate diverse terrains
- 1 was not functional
  - Optimal Performance in varying weather
    - Should be "perform"

Quality of the Decompositions

- Output seems to include more of a hierarchical decomposition of the green box are valid functions
- Decompositions otherwise were:
  - Physical components
  - Physical attributes
  - Requirements

No actors = **Not** use cases

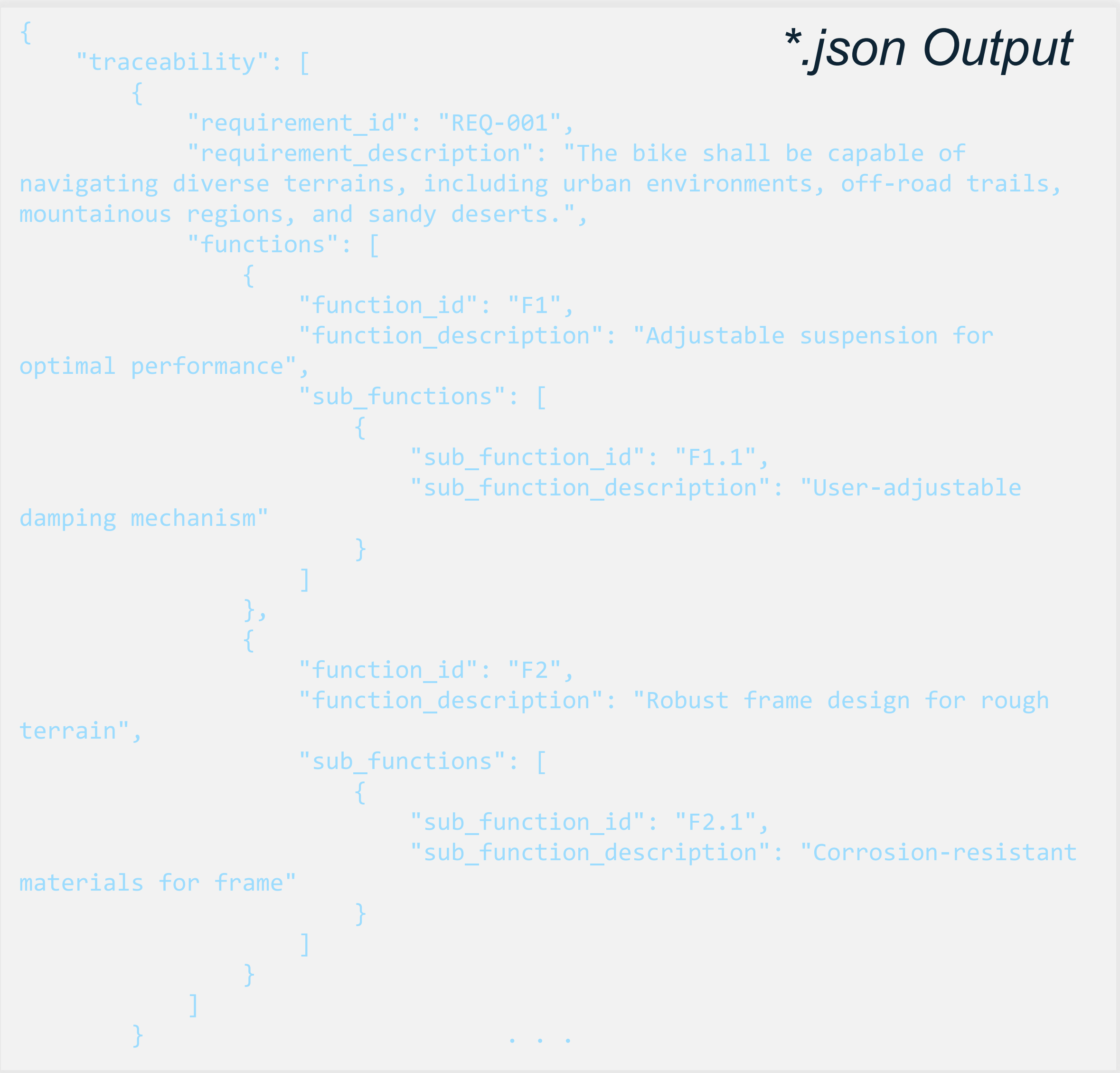
JSON format did not include actors to keep complexity lower given that functional decompositions were challenging for the model


Optimal performance in varying weather
include
Weather-resistant components
include
Waterproof brakes and gearshifters
include
Reliable lighting system
include
Rechargeable battery with 4-hour runtime
Ensure safety and comfort
include
Weight capacity and ergonomics
include
Frame and component design for weight capacity
include
Adjustable handlebars and seat posts
Navigate diverse terrains
include
Adjustable suspension for optimal performance
include
User-adjustable damping mechanism
include
Robust frame design for rough terrain
include
Corrosion-resistant materials for frame

Sequential Framework Case Study Outputs: Traceability Matrix

Traceability Analysis

- The model re-numbered the requirements when provided the requirement IDs - few shot/more rigid prompts may be necessary
- Example: REQ-001
  - The bike shall be capable of navigating diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts
  - 2 functions it is allocated to F1.1 and F2.1
    - F1.1: User-adjustable damping mechanism
    - F2.1: Corrosion-resistant materials for frame
- All requirements traced were properly traced.
  - Need to quantify accuracy across larger number of requirements and functions



*.json Output

```
{
    "traceability": [
        {
            "requirement_id": "REQ-001",
            "requirement_description": "The bike shall be capable of
navigating diverse terrains, including urban environments, off-road trails,
mountainous regions, and sandy deserts.",
            "functions": [
                {
                    "function_id": "F1",
                    "function_description": "Adjustable suspension for
optimal performance",
                    "sub_functions": [
                        {
                            "sub_function_id": "F1.1",
                            "sub_function_description": "User-adjustable
damping mechanism"
                        }
                    ]
                },
                {
                    "function_id": "F2",
                    "function_description": "Robust frame design for rough
terrain",
                    "sub_functions": [
                        {
                            "sub_function_id": "F2.1",
                            "sub_function_description": "Corrosion-resistant
materials for frame"
                        }
                    ]
                }
            ]
        }                           . . .
```

Sequential Framework Case Study Outputs: Risk Matrix

Formatting and Output

Format is: RISK-### [Likelihood, Consequence]: Description

- RISK-001 [4, 5]: Failure to design and manufacture a bike that can navigate diverse terrains, including urban environments, off-road trails, mountainous regions, and sandy deserts.
- RISK-002 [3, 4]: Inadequate adjustable suspension design, leading to poor performance on varying terrains.
- RISK-003 [3, 4]: Insufficient robust frame design, leading to durability issues in rough terrain and harsh weather conditions.
- RISK-004 [3, 4]: Inability to ensure optimal safety and comfort for the rider in varying conditions, due to inadequate weight capacity and ergonomics.
- RISK-005 [3, 4]: Failure to design and manufacture weather-resistant components, leading to poor performance in varying weather conditions.
- RISK-006 [2, 3]: Inadequate lighting system design, leading to safety issues in low-light conditions.
- RISK-007 [4, 5]: Supply chain disruptions, leading to delays in component delivery and manufacturing.
- RISK-008 [2, 3]: Regulatory compliance issues, leading to delays in certification and market release.
- RISK-009 [3, 4]: Inadequate testing and validation, leading to poor performance and safety issues in real-world conditions.
- RISK-010 [2, 3]: Lack of contingency planning, leading to unpreparedness for unexpected events and crises.

Response Analysis

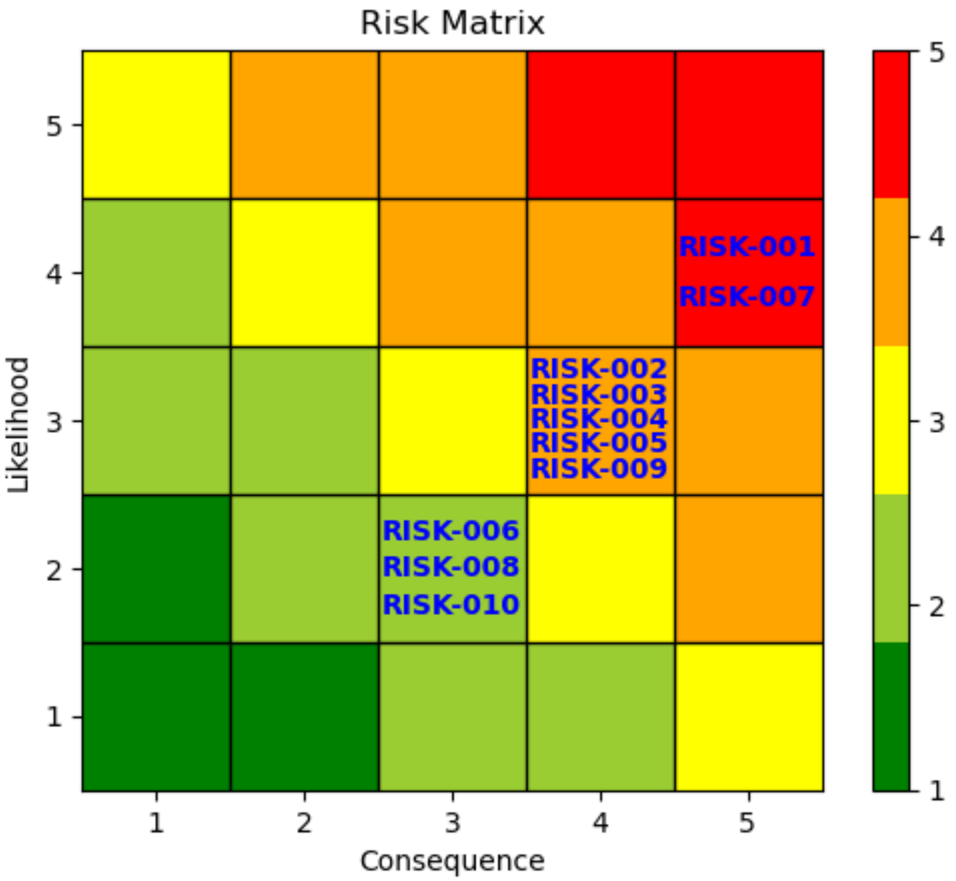
In general, responses seemed conservative (higher likelihood/consequence) with estimates given no context on a business plan or otherwise.

There were a few outliers to this stance, including:

- Risk-004: while no context provided on optimal safety, non-optimal safety was consequence 4
- Risk-006: while no context of time of day provided, safety issues was consequence 3

It seems human safety was less consequence than should be – human safety should be of the upmost importance (e.g. 5).

Further investigation on this model's approach to safety is required for any statistical significance.



Challenges and Limitations

- Class definitions for Agents and Tasks was required to be able to easily pass inputs
  - Kicking off the crew passed a System Description provided by the user

- Some of the outputs from Llama3 with the sequential approach were not in proper format or out of context
  - ChatGPT4o understood what a use case was – Llama 3 70B struggled with use cases and decomposition of functions was not valid

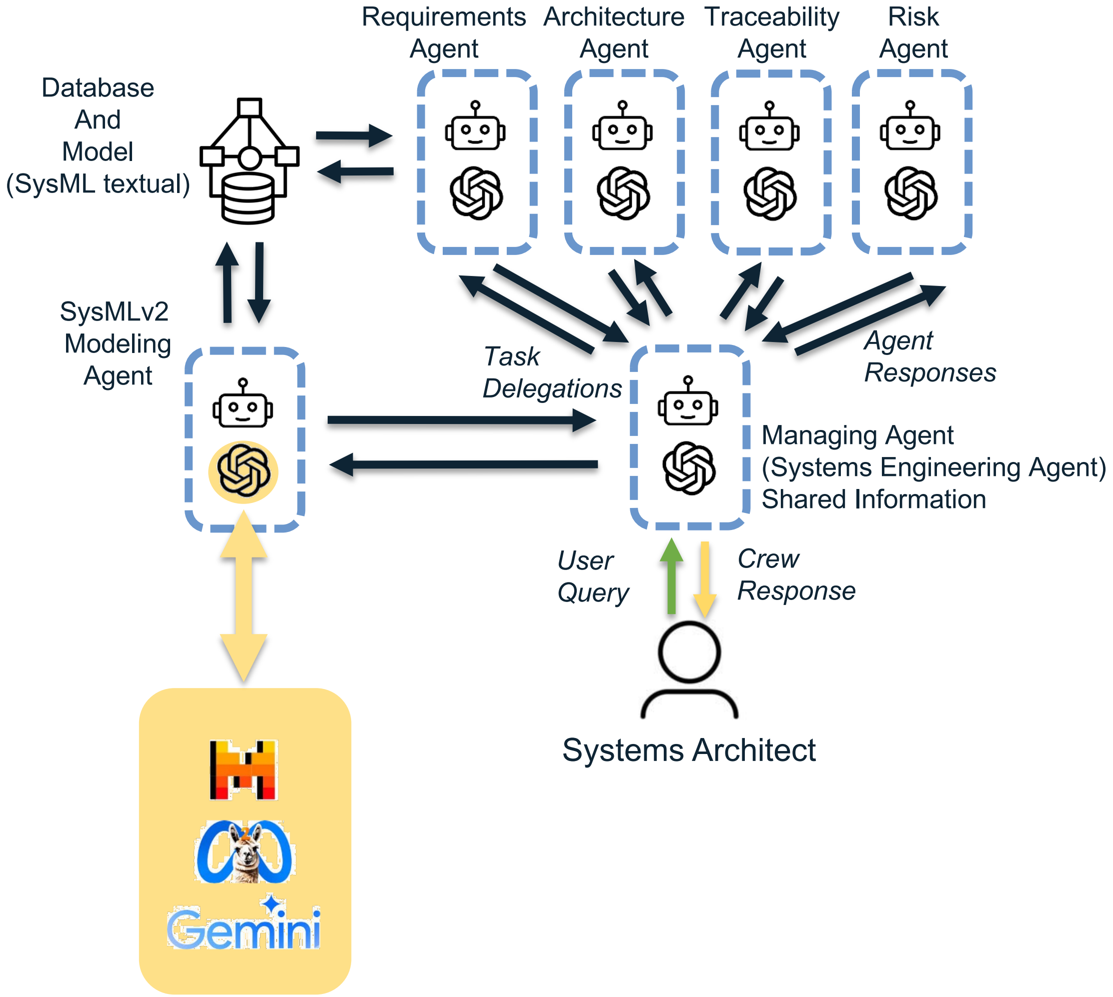
- The sequential agent approach with Llama3 demonstrated that tighter guardrails or constant interaction with a human in the loop provides the best outputs as seen with ChatGPT4o with the singular model

Summary

- Explored different language models (ChatGPT4o and Llama3) for use with Systems Engineering common approaches and analyses
- Created/defined SE agents
- Created/defined SE tasks with standard *.json format for simple LLM integration
- Explored singular/chat interface and sequential tasking frameworks for SE
- Shared a case study comparing LLM frameworks for an all terrain bike
  - Zero shot approach outputs suggest custom SE LLMs and/or techniques are required
- Discussed limitations on LLM scope of knowledge
  - Case study focused on a non-novel implementation
  - Need to ensure that the LLM has been trained on relevant knowledge and/or has access to database of relevant knowledge (e.g. via RAG)
  - Dictates that the LLM will know how to present your idea, but will not - in most cases - be able to come up with the idea itself
- Need a quantitative method for assessment of language model performance for SE tasks
- A step towards the role of future SE to focus on architecture in plain language and tool agnostic architecture

Future Work

- Explore using other models and techniques (RAG or fine tuning) with Agents
- Explore varying levels of human in the loop with Agents
- Explore a SysMLv2 modeling agent to perform modeling as a part of all user queries
  - Extends SysMLv2 research (Longshore, 2024)
- Explore a cost modeling agent to perform project cost modeling
  - Extends AI cost modeling research (Madachy, 2024)
- Explore quantitative methods of comparing SE work product outputs from LLMs / Agents
  - Extend SysEngBench (Bell, 2024)



References

- Bell, R., Madachy, R., & Longshore, R. (2024). *Introducing SysEngBench: A Novel Benchmark for Assessing Large Language Models in Systems Engineering*. 21st Annual Acquisition Research Symposium.
- CrewAI Documentation. (July 2024). Retrieved from https://docs.crewai.com/
- Longshore, R., Madachy, R., & Bell, R. (2024). *Leveraging Generative AI to Create, Modify, and Query MBSE Models*. 21st Annual Acquisition Research Symposium.
- Madachy, R., Bell, R., & Longshore, R. (2024). *Systems Acquisition Cost Modeling Initiative for AI Assistance*. 21st Annual Acquisition Research Symposium.
- Moura, J. (July 2024). CrewAI. Retrieved from https://github.com/joaomdmoura/crewAI
- Systems Engineering Librarys “se-lib” Documentation. (July 2024). Retrieved from http://se-lib.org/
- Prompting Guide. (July 2024). Retrieved from https://www.promptingguide.ai/research/llm-agents

Contact Information

Ryan Bell: ryan.bell@nps.edu

Ryan Longshore: ryan.longshore@nps.edu

Raymond Madachy, PhD: rjmadach@nps.edu

See our other research:

- *Tuesday 1000-1010 – "System Product Line Cost and Investment Modeling Applied to UUVs"*
- *Thursday 1930-2010 – "Systems and Software Engineering Cost Modeling of AI Assistance"*
- *Saturday 0800 to 1700 – "Open Source System Modeling with Python*

Naval Information Warfare Center
ATLANTIC

Funding provided by NIWC Atlantic Naval Innovative Science and Engineering (NISE) / Section 219

34th Annual **INCOSE**
international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

www.incose.org/symp2024
#INCOSEIS