A. Functional Capabilities

This dimension consolidates all architectural options that define an agent's core operational mechanisms, such as how it perceives, reasons, and acts.

4.1 Input Modality

Modality defines whether an agent operates using a single type of data input or multiple.

- **Single-modality**: These agents process one type of input, such as text, vision, or audio. This approach is suited for tasks where computational efficiency and simpler data interpretation are priorities.
- **Multi-modality**: These agents are designed to integrate and process a combination of inputs, including text, audio, images, and video, to achieve a more holistic understanding of their environment.

4.2 Context

Context management involves gathering and organizing information about the agent's operating environment to better understand user intentions and goals. It is classified into two sub-dimensions: context types and goal-seeking strategies.

- **Context Types**:
  - **Interactional Context**: Refers to direct user actions and behaviors, such as mouse clicks, typing patterns, and gestures.
  - **Environmental Context**: Describes the state of the digital environment in which the user is operating.
  - **Semantic Context**: Pertains to the underlying meaning and intent, extracted from sources like user annotations or prompt content.
- **Goal Seeking**:
  - **Passive Suggestion**: The agent analyzes goals that are explicitly articulated by the user, often through a dialogue interface.
  - **Proactive Suggestion**: The agent anticipates user goals by interpreting multimodal context from UI elements and user interactions, going beyond explicit commands.

4.3 Role-Playing

In multi-agent systems, assigning distinct roles is a crucial architectural decision that defines functions and interactions.

- **Agent-as-a-coordinator**: This agent formulates high-level strategies and orchestrates task execution by delegating responsibilities to other agents or systems.
- **Agent-as-a-worker**: This agent acts as the core executor, responsible for generating local, sub-task level strategies to complete assigned tasks.

4.4 Goal Type

Agents establish goals to guide their planning and action phases.

- **Task Completion**: Goals that direct an agent to achieve specific, complex objectives, such as crafting items in a virtual environment or executing predefined functions.
- **Communication**: Goals involving interaction with other agents or humans to exchange information or collaborate on tasks.
- **Learning**: Goals that require agents to navigate and adapt to unfamiliar settings, balancing exploration of new areas against the exploitation of known resources.

4.5 Planning

Planning is operationalized through a reasoning process where the agent uses cognitive steps and logical frameworks to solve complex problems.

- **Planning Engine**: The central component that orchestrates high-level strategic activities, enabling the agent to process reasoning, generate plans, and adapt.
  - **Internal Planner**: Embedded within the agent's core architecture, it uses the agent's intrinsic capabilities to autonomously generate and execute plans.
  - **External Planner**: Specialized, often domain-specific tools integrated to handle complex tasks requiring detailed planning logic.
  - **Hybrid Approach**: Combines an internal FM-based planner for flexible problem translation with a classical external planner for efficient execution.
- **Tool Selection**: The process of selecting appropriate tools to execute a plan.

  - **Internal Tools**: Capabilities and algorithms built directly into the agent's system to provide tailored solutions.
  - **External Tools**: Outside services and data sources that the agent interacts with, typically via APIs, to extend its capabilities.
- **Plan Generation**: Translates the reasoning process into a sequence of actionable steps.
  - **Single-path Plan Generator**: Creates a linear, step-by-step plan, ideal for tasks in predictable environments.
  - **Multi-path Plan Generator**: Devises several potential routes to achieve a goal, allowing the agent to dynamically adjust its plan in complex or uncertain scenarios.

4.6 Memory

Memory is a critical architectural dimension that enables agents to store, retrieve, and use information from past interactions.

- **Memory Types**:
  - **Long-term Memory**: Retains information over long periods, essential for tasks requiring historical data, knowledge, and past experiences.
  - **Short-term Memory**: Handles information relevant to immediate tasks, such as configuration, working context, and recent events.
  - **Selective Forgetting**: Enables agents to discard irrelevant or outdated information to maintain optimal performance.
- **Memory Format**:
  - **Natural Language Memory**: Stores information in a flexible, easily understandable form, preserving rich semantic details.
  - **Embedding Memory**: Encapsulates memory information into compact embedding vectors, enhancing the efficiency of memory retrieval.
  - **Databases**: Use databases to enable precise memory operations through structured queries, providing efficient data management.
  - **Structured Lists**: Organizes memory in lists or hierarchical structures to define relationships between elements and facilitate rapid access.
- **Memory Operation**:

- **Memory Reading**: Involves extracting valuable information based on recency, relevance, and importance to enhance the agent's actions.
- **Memory Writing**: Focuses on storing perceived environmental information while managing duplication and preventing overflow.
- **Memory Reflection**: Enables agents to summarize and infer high-level insights from past experiences for more abstract decision-making.
- **Memory Sharing**: Allows agents to access and contribute to a common memory pool, enhancing collaborative abilities.

4.7 Action

This dimension encompasses the architectural choices governing how an agent executes tasks and interacts with its environment.

- **Agent Coordination**: Mechanisms to ensure multiple agents can work together effectively.
  - **Centralized Coordination**: A single "manager" agent dictates tasks to other agents in a command-and-control model.
  - **Federated Coordination**: A central "orchestrator" handles high-level tasks while granting high autonomy to individual agents for their own execution logic.
  - **Fully Distributed Coordination**: No central authority; all agents are peers that coordinate directly through peer-to-peer communication.
- **Agent Communication**: Strategies that dictate how agents share information to collaborate.
  - **Full Transparency**: All information is openly shared among agents to maximize cooperation.
  - **Partial Transparency**: Information sharing is restricted through patterns like Goal-based sharing, Role-based sharing, Sensitive data withholding, and Context-aware sharing.
- **Tool Calling**: A fundamental capability that enables agents to access and manipulate external tools.
  - **Tool Invocation**: Defines how an agent interacts with a tool. This can be Configuration-free (using predefined interfaces) or Customizable (dynamically adjusting parameters).

   - **Tool Interface**: The layer through which an agent operates a tool. This can be an API (structured, machine-to-machine interface) or UI Understanding (perceiving and interacting with a user interface).
   - **Tool User Type**: Determines the source of feedback for tool refinement. This can be User experience-driven (based on human feedback) or Agent experience-driven (based on the agent's own performance data).
   - **Tool Learning**: Describes how agents acquire new tool-use capabilities. This can be API-based learning (mastering structured interfaces) or UI-based learning (adapting to evolving user interfaces).
   - **Output Integration**: The process of incorporating tool outputs into an agent's workflow. This can be Inline integration (direct use of output for immediate execution) or Multi-source integration (synthesizing outputs from multiple tools).
- **Actuation**: An agent's capability to execute actions that affect a physical or digital environment.
   - **Physical Actuation**: Involves actions in the real world, such as robotic movements.
   - **Virtual Actuation**: Encompasses actions within software environments, such as automated data entry or network configuration changes.

4.8 Reflection

This dimension equips an agent with meta-cognitive capabilities to introspectively analyze its own performance and behavior to learn and improve.

- **Reflected Artifacts**:
   - **Workflow/Plan Generation**: Reflection on the agent's initial goals, prompts, and planning steps before execution.
   - **Intermediate Result**: Reflection on logs of reasoning processes, planning steps, and tool use during operation.
   - **Final Output**: Evaluation of the final result against the initial goals to verify the outcome.
- **Provider**: The source of guidance for plan refinement.
   - **Self-reflection**: The agent generates its own feedback to evaluate and refine its plans.

- **Cross-reflection**: One or more external agents or FMs are used to provide feedback.
- **Human Reflection**: The agent collects feedback directly from human users.
- **Environmental Reflection**: The agent obtains feedback from the external world, such as task completion signals or errors.

- **Mechanisms**: Specific techniques an agent employs to structure its reflection process.
  - **Chain of Thought (CoT)**: The agent articulates its reasoning process step-by-step to trace decisions and identify errors.
  - **Tree of Thoughts (ToT)**: An advanced mechanism that allows an agent to explore multiple reasoning pathways simultaneously.

4.9 Learning Capability

These are fundamental to an agent's ability to adapt and improve its performance over time.

- **Self-learning**: Allows an agent to autonomously improve by updating its knowledge base and refining its algorithms based on historical data.
- **In-context learning**: Enables an agent to adapt to new tasks on-the-fly by interpreting the provided context, typically through prompt engineering, without altering its internal model.
- **Group learning**: Involves multiple agents sharing knowledge to enhance collective performance. It can be implemented through Peer-to-peer learning (direct knowledge sharing) or Hierarchical learning (information processed across different levels of an organization).

4.10 Workflow

This dimension governs the architectural pattern for the procedural execution of tasks, distinct from Planning and Agent Coordination.

- **Centralized Workflow**: Task management is concentrated at a single control point, enhancing consistency but risking performance bottlenecks.
- **Decentralized Workflow**: Task control is distributed across multiple agents, increasing system flexibility and resilience.

4.11 Access to Underlying Models

This dimension addresses architectural choices related to the agent's core engine: the foundation model(s).

- **Underlying Model Types**: Determines the agent's fundamental scope of competence.
    - **Domain-specific AI-based agents**: Engineered to excel at a narrow set of specialized tasks within a particular domain.
    - **General Purpose AI (GPAI)-based agents**: Designed for versatility, capable of performing various tasks across different domains.
- **Model Composition**: Defines how many and what kinds of models constitute the agent's core engine.
    - **Single-model**: The agent relies on a single, monolithic model to handle all tasks.
    - **Multi-model**: The agent employs multiple models (e.g., mixture of experts, ensemble method) to enhance performance or task-specific capabilities.
- **Model Sourcing**: The strategic decision of how an agent's core AI technology is acquired.
    - **Sovereign AI**: The model is developed and maintained entirely in-house.
    - **Partially Sourced AI**: A hybrid approach combining internal development with external resources, like fine-tuning a base model on proprietary data.
    - **Fully Externally Sourced AI**: The agent relies entirely on third-party AI models and services, typically via APIs.
- **Training Method**: A critical architectural decision based on the system's data architecture.
    - **Centralized Training**: The traditional approach where a complete dataset is aggregated in a single location to train a model.
    - **Collaborative Learning**: Used when data cannot be centralized (e.g., due to privacy); enables multiple parties to jointly train a model without sharing raw data, using patterns like federated learning.

B. Non-functional Qualities

These qualities are critical system-level attributes that determine how well an agent performs its functions, constraining the architectural design and dictating the overall quality.

- **Performance**: A multifaceted quality encompassing Accuracy (correctness of outputs) and Efficiency (resources consumed relative to results achieved).
- **Reliability**: The degree to which an agent performs its specified functions correctly under normal conditions.
- **Robustness**: The agent's ability to maintain functionality in the presence of invalid, unexpected, or stressful inputs.
- **Resilience**: The ability of an agent to prepare for, absorb, recover from, and adapt to operational failures.
- **Security**: Addresses the protection of the agent, its data, and services, encompassing Confidentiality, Integrity, and Availability.
- **Maintainability**: Concerns the ease with which an agent's architecture can be modified. It is characterized by Modifiability, Testability, and Reusability.
- **Responsible & Trustworthy AI (RAI)**: A cross-cutting concern that integrates ethics and accountability into the agent's design. It includes characteristics such as Privacy, Fairness, Transparency, Observability, and Explainability.
- **Usability**: Addresses the effectiveness, efficiency, and satisfaction with which users can interact with an agent.
- **Compatibility**: An agent's ability to exchange information and perform its functions while sharing an environment with other systems.