1. Chain-of-Agents and Multi-Agent Reasoning Systems

The **Chain-of-Agents** framework [1] introduces a pipeline of collaborative LLMs, where each agent refines the output of the previous one. While it excels in long-context tasks, it lacks flexibility in handling diverse task categories (e.g., UI vs. API design). Similarly, **Corex** [2] proposes cooperative reasoning across multiple LLMs, but focuses more on abstract reasoning rather than structured software development.

Other frameworks like **ReConcile** [6] and **Mixture-of-Agents** [4] aim to improve reasoning by combining LLMs with diverse knowledge and prompting styles. These systems show promise in boosting factual accuracy but are often limited to textual tasks like QA or summarization. CodeCodez extends this line of work by integrating role-specific LLMs and directing them via hierarchical structures (trees and graphs) to generate software artifacts.

2. Task Decomposition Frameworks

Systems like **HALO** [3] and **TDAG** [12] use logical task decomposition for orchestrating multiple LLM agents in hierarchical structures. These systems focus on automating decision-making processes or workflow orchestration. However, their application in software engineering remains experimental. CodeCodez leverages similar DAG structures but applies them specifically to the software development lifecycle, with specialized attention to sequencing, dependency resolution, and validation.

The **Tree-of-Code** framework [13] is one of the few that blends token-level tree parsing with practical code generation. Yet, its scope remains limited to function-level or class-level generation. In contrast, CodeCodez expands the scope to full software pipelines, integrating documentation, testing, and configuration scripts alongside code.

3. Code Generation Frameworks

There are several noteworthy systems focusing purely on code generation:

- **AgentCoder** [15] and **AdaCoder** [16] propose multi-agent code generation workflows with built-in testing and self-reflection loops. However, they largely rely on linear or chain-based reasoning rather than parallel DAG-style execution. Their debugging and optimization loops are also static in nature, while CodeCodez incorporates dynamic loopbacks based on token-category classification and verification results.