Agent Design Analysis

# Agent Design

## Multi-tool Single Agent

This Agent design can be leveraged to solve a domain specific problem such as order placing among different use cases like order placing, refund or customer support.  
Multiple functions with different use cases can be created:

* Order fetch tool
* Promo code validator
* Delivery EST
* Checkout tool

It works well when the series/set of tools cater to one part of business problem.

## Multi-tool Multi Agent

Can be utilized in building systems that require immediate connectivity between different agents that support variety of business use cases.  
In this structure, one agent as mentioned above can be used in tandem with the customer support agent for customer related queries on placing the order.  
More research is needed for building production grade applications with memory tracking.

# Frameworks

## ReAct Framework

ReAct (Reason + Act) is a powerful framework for building AI agents that seamlessly integrates reasoning and decision-making with task execution. By leveraging large language models (LLMs) like Gemini, ReAct agents can dynamically analyze problems, choose appropriate tools, and iteratively work towards solutions.

This framework is suitable for supporting mainly customer queries and problems where chain of thoughts is required. The most feasible option here is to use Langgraph’s inbuilt [create\_react\_agent](https://python.langchain.com/api_reference/langchain/agents/langchain.agents.react.agent.create_react_agent.html) module.

## ReWoo Framework

ReWOO, which stands for Reasoning WithOut Observation is a modular paradigm that decouples the reasoning process from external observation. By doing so, ReWOO greatly decreases token usage, repeated execution, and the enormous computational complexity caused by duplicate prompts. As a result, ReWOO achieves higher accuracy than existing frameworks as well as being 5 times more efficient in token counting on a multi-step reasoning benchmark.

ReWOO generates a single, consolidated set of calls to the external tools, fetching all necessary information at once. This makes it different from ReAct framework where the tool outputs are parsed by LLM to understand which action to take next.

Here is a langchain implementation of the same:

<https://langchain-ai.github.io/langgraph/tutorials/rewoo/rewoo/>

# Challenges

## Memory Integrations

LSTM & thread memory integrations need research improvement in case of maintaining multi agent applications. While there are various tools available to track memory in multi tool agent frameworks such as [Langmem](https://blog.langchain.dev/langmem-sdk-launch/) and [Redis](https://redis.io/).

## Accuracy

Accuracy is a major problem while building production grade applications as decisions purely depend on LLM reasoning and model’s capability. But guardrails can be implemented to reduce the inconsistencies as far as possible. One of the things is offered by Langgraph where a graph workflow can be defined based on process flow.  
So, if the user flow is clear, then instead of a ReAct agent we can utilize graph workflows.

## Performance Tracking

Custom dashboards and metrics can be designed to track the applications using Prometheus/Grafana/Opentelemetry. Metrics such as tool calls, user feedback and llm reasoning can be tracked and then the model can be finetuned to adapt to tool calls.

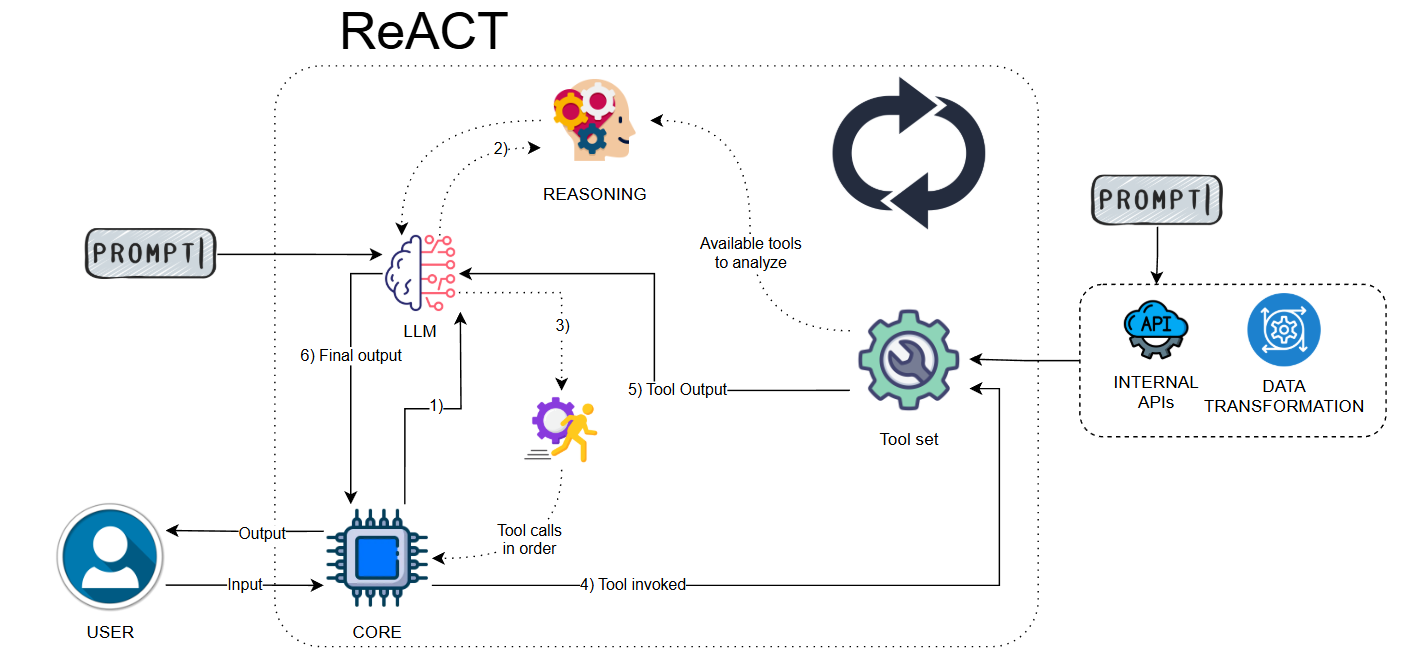
While this sounds easy, it requires large amount of data and computational resources to finetune any LLM.

## LLM capability

Efficient tool calls depend LLM to LLM where the best in class is GPT o series. There are smaller models available as well such as meetkai but then, they do not support multiple tool calls and hence result in failure of the frameworks.

# Conclusion

For the given problem, it is good to use ReAct agent based on user scenario with a suitable tool calling LLM from Groq.



Conceptual Map