Prompt Engineering / AI Agents

Recommend how many steps ahead an agent should plan based on task predictability and cost of replanning.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Agent Planning, ReAct Optimization, Cost-Performance Trade-offs
Updated: May 2026
Why This Prompt Exists
Plan too few steps and the agent is myopic — misses opportunities that require longer sequences. Plan too many and the agent wastes tokens planning for contingencies that never happen.

You get:

- agents that stop too early (could have solved with 2 more steps)
- agents that plan 20 steps for a 3-step task (wasted tokens)
- plans that become invalid after first step (replanning cost high)
- no systematic way to choose planning depth
- agents that plan when they should act, act when they should plan

But planning horizon depends on factors:

- task predictability: if the world changes, long plans break
- cost of replanning: cheap to replan → shorter horizon
- cost of being wrong: high cost → longer horizon (need to foresee problems)
- task depth: how many steps naturally required
- feedback frequency: frequent feedback → shorter horizon (react faster)

Without horizon determination, agents plan poorly.

This prompt recommends optimal planning depth for any task.

The Prompt

```
Assume the role of an agent planning architect who optimizes planning horizons.

Your task is to recommend how many steps an agent should plan ahead.

Generate:

1. TASK CHARACTERISTICS
   - Minimum steps required (if known)
   - Maximum steps possible (upper bound)
   - Typical step duration (seconds/minutes)

2. ENVIRONMENT FACTORS
   - Predictability (High = deterministic, Low = other agents/humans intervene)
   - Feedback frequency (How often can agent observe results?)
   - Replanning cost (Low = cheap to rethink, High = expensive)

3. COST OF BEING WRONG
   - Low (wrong plan = waste a few seconds)
   - Medium (wrong plan = wasted API cost, user annoyance)
   - High (wrong plan = data loss, incorrect information to user)

4. PLANNING HORIZON RECOMMENDATION
   - Recommended steps ahead: [N]
```

- Rationale: [one sentence]

5. ADAPTIVE HORIZON STRATEGY
- Start with [N] steps
- Reduce horizon if: [condition, e.g., "environment is changing faster than expected"]
- Increase horizon if: [condition, e.g., "consistent success at current horizon"]

6. WHEN TO PLAN VS. ACT
- Plan for: [tasks with high cost of wrong action]
- Act immediately for: [tasks with low cost, high feedback frequency]

7. READY-TO-USE PLANNING PROMPT
- A prompt that implements this horizon strategy

INPUTS:

Task description:
[E.G., "Book a flight, hotel, and rental car for a business trip"]

Environment type:
[DETERMINISTIC / SEMI-PREDICTABLE / HIGHLY DYNAMIC]

Replanning cost:
[LOW / MEDIUM / HIGH] (e.g., "LOW — can recompute plan in <100ms")

Cost of wrong action:

```
[LOW / MEDIUM / HIGH] (e.g., "HIGH — wrong booking could cost real
money")

Agent framework:
[REACT / PLAN-EXECUTE / HIE RARCHICAL]

RULES:
- Longer horizons are not always better (diminishing returns after
~5-10 steps)
- For highly dynamic environments, plan 1-2 steps ahead and replan
often
- For deterministic environments, plan longer (10+ steps) to optimize
globally
- Flag tasks where planning is unnecessary (single-step, trivial)
- Test horizon by measuring task success vs. planning token cost
```

How To Use It

- Start with a conservative horizon (3-5 steps) and adjust based on performance.
- For highly dynamic environments (chat, games), plan 1-2 steps ahead and react fast.
- For deterministic, multi-step tasks (data pipelines, form filling), plan longer.
- Measure the cost of planning (tokens, latency) vs. the benefit (fewer errors).
- Implement adaptive horizons — shorten if the environment is changing unexpectedly.

Example Input

Task description:
"Answer customer support questions by searching knowledge base and generating responses"

Environment type:

"SEMI-PREDICTABLE — customers ask varied questions, knowledge base is stable"

Replanning cost:
"LOW — can search again in <500ms" **Cost of wrong action:**
"MEDIUM — wrong answer frustrates customer but no data loss"

Why It Works
Most agents use a fixed planning horizon — usually what the framework default provides — which is rarely optimal for the specific task.

This framework improves outcomes by forcing:

- task characteristic analysis (how many steps needed)
- environment predictability assessment (will the world change?)
- cost trade-off evaluation (replanning cost vs. being wrong)
- adaptive horizon strategy (change horizon based on conditions)
- plan vs. act decision (don't plan what you can react to)

Great planning horizon determination doesn't maximize steps — it balances foresight against agility.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Memory Retrieval Strategist