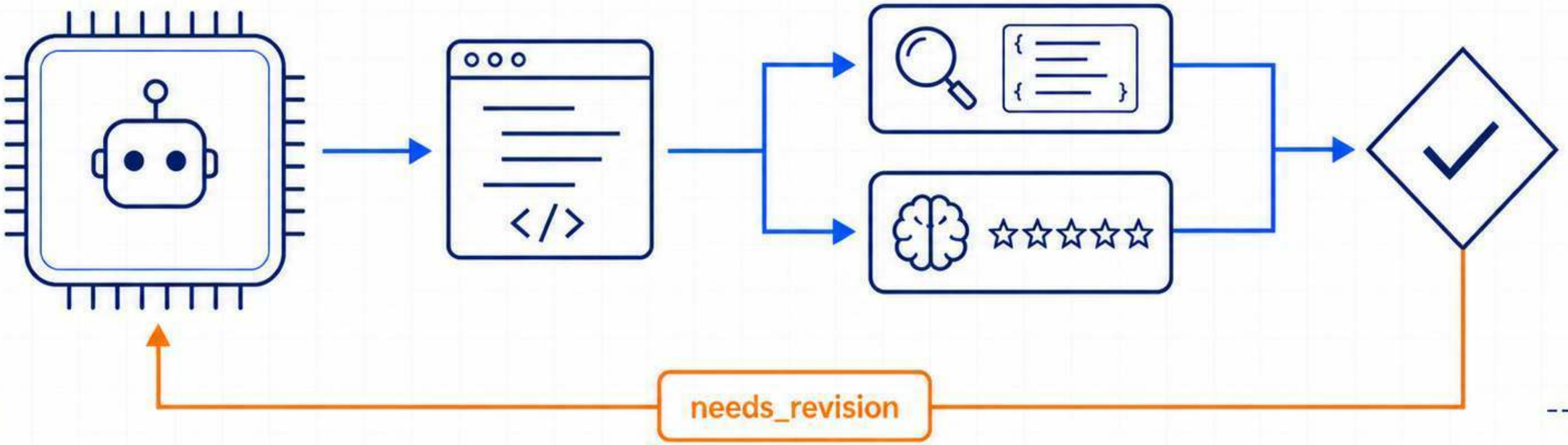
Deep Agents 实战

第 17 讲：Rubric Middleware（评分量规中间件）



生成结束，不等于通过验收

一个完整答案，仍可能在边界输入上失败

常规输入

`find_duplicates([1, 2, 2, 3, 1]) → [2, 1]`

边界输入

`find_duplicates([[1], [1]]) → TypeError`

Prompt（提示词）
声明要求

Self-Check（自检）
再次判断

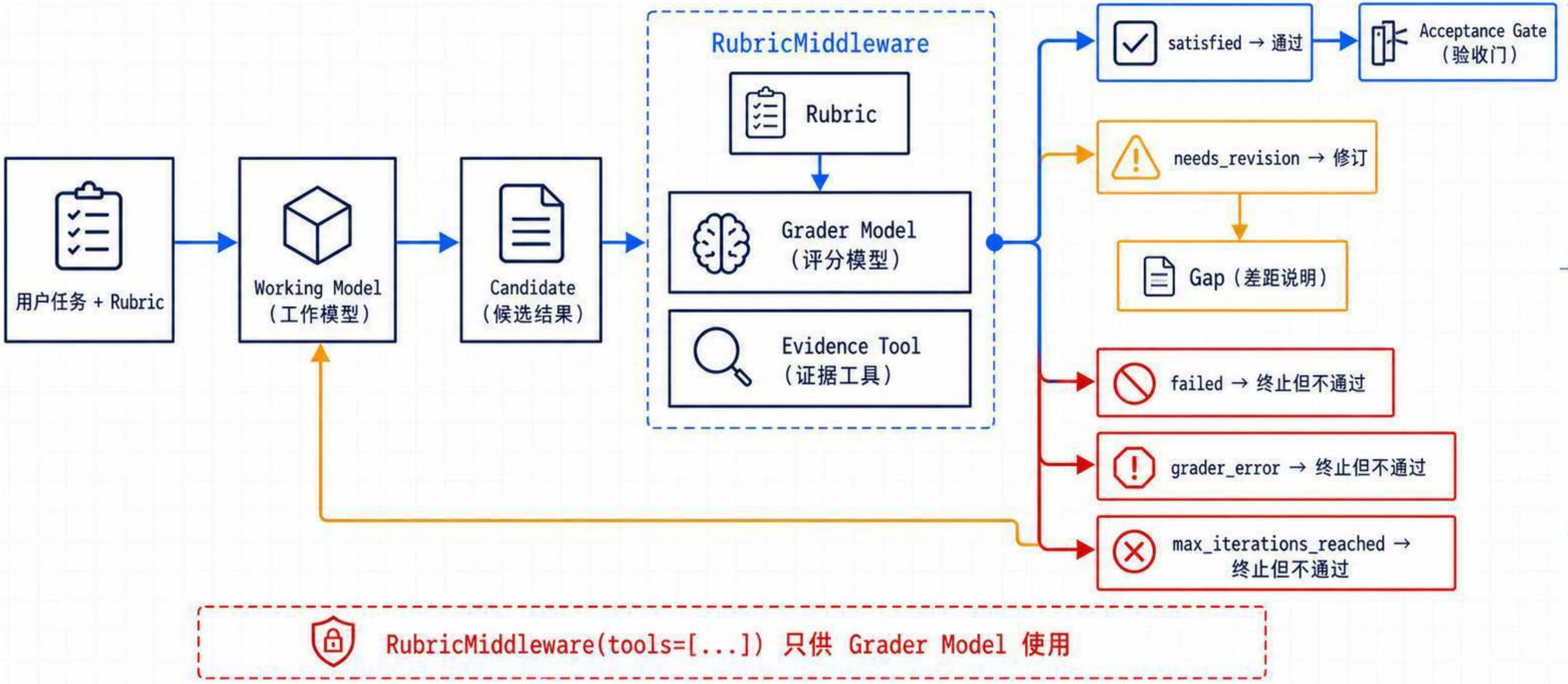
Retry（重试）
再次生成

- 仍缺少
- 独立评审
- 工具证据
- 可执行 Gap
- 失败关闭

生成结束 ≠ 验收通过

Rubric Middleware 在自然停止后接管验收

只有 needs_revision 会回环，只有 satisfied 可以放行



先安装依赖，再分开配置两个模型

工作模型负责任务，评分模型负责验收

```
uv add "deepagents==0.7.1" langchain-openai
```

```
import os
from langchain.chat_models import init_chat_model

working_model = init_chat_model(
    os.environ["WORKING_MODEL"]
)
grader_model = init_chat_model(
    os.environ.get(
        "GRADER_MODEL",
        os.environ["WORKING_MODEL"],
    )
)
```

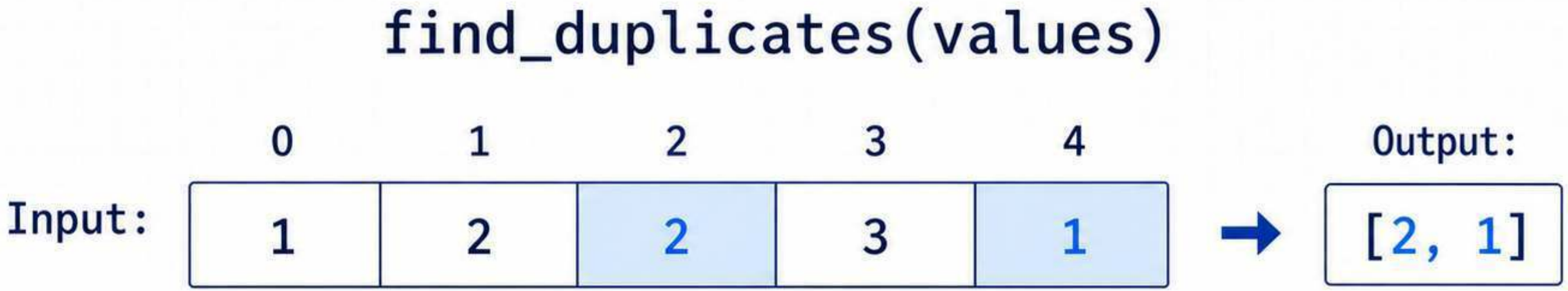
WORKING_MODEL
provider:model-id

GRADER_MODEL
provider:model-id

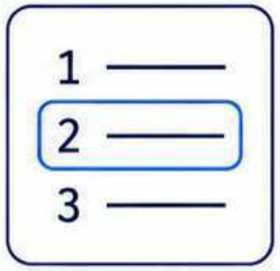
deepagents>=0.6.5 | Beta API

先固定 find_duplicates 的任务契约

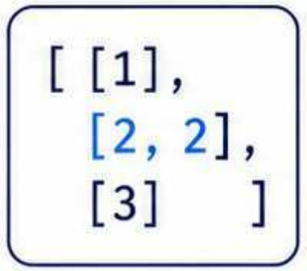
Task、测试和 Rubric 必须使用同一种排序语义



每个重复值只返回一次



按第二次出现的位置排序



支持列表等不可哈希值

不修改输入列表

把完成条件写成可取证的 Rubric

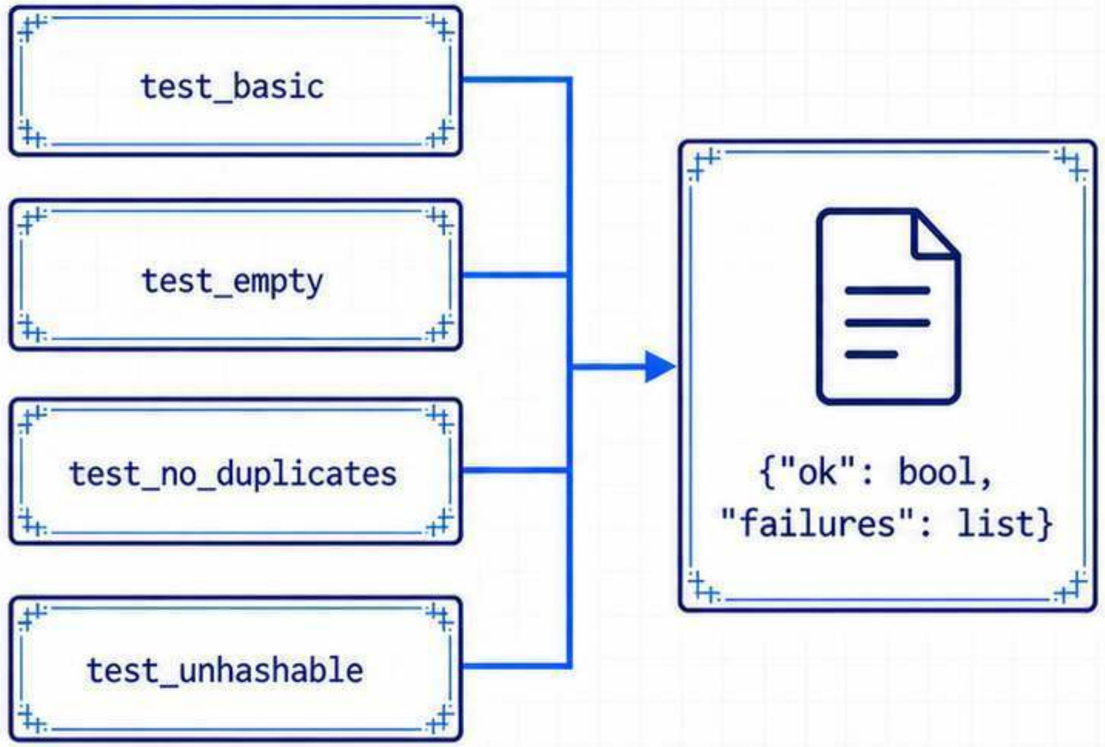
通过前必须测试最新候选，而且测试结果必须 ok=true

```
rubric = """
  - Before returning satisfied, call run_test_suite
    with the latest candidate code.
  - The run_test_suite result must contain ok=true.
  - Each duplicated value appears exactly once.
  - Result order follows the second appearance.
  - Unhashable values are supported.
  - The input list is not mutated.
""".strip()
```

Evidence Tool 把候选源码变成测试证据

评分模型不猜代码是否正确，而是读取结构化测试结果

```python
from copy import deecpopy
from langchain.tools import tool

@tool
def run_test_suite(code: str) -> dict:
    """Run behavioral tests on find_duplicates."""
    namespace = {}
    exec(code, namespace)
    fn = namespace["find_duplicates"]
    tests = [
        ("test_basic", [1, 2, 2, 3, 1], [2, 1]),
        ("test_empty", [], []),
        ("test_no_duplicates", [1, 2, 3], []),
        ("test_unhashable", [[1], [1], 2], [[1]]),
    ]
    failures = []
    for name, values, expected in tests:
        original = deecpopy(values)
        try:
            actual = fn(values)
            if actual != expected:
                failures.append(f"{name}: expected {expected}, got {actual}")
            if values != original:
                failures.append(f"{name}: input was mutated")
        except Exception as exc:
            failures.append(f"{name}: {type(exc).__name__}: {exc}")
    return {"ok": not failures, "failures": failures}
```



⚠ 教学中的 exec 不是 Sandbox

不接模型，先证明测试工具能抓住错误

没有 API Key，也能先验证 Evidence Tool

```python
def find_duplicates(values):
    seen = set()
    duplicates = []
    for value in values:
        if (
            value in seen
            and value not in duplicates
        ):
            duplicates.append(value)
        seen.add(value)
    return duplicates
```

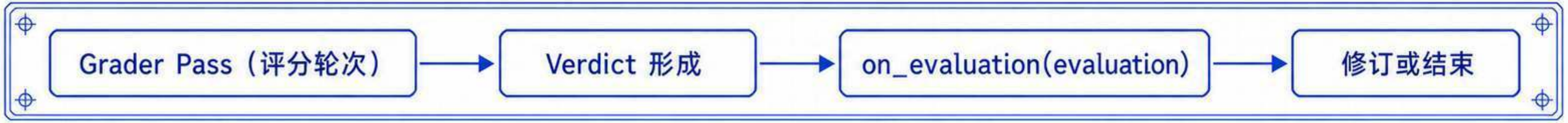
```
run_test_suite.invoke({"code": bad_candidate})
```

```
{
  'ok': False,
  "failures": [
    "test_unhashable: TypeError: unhashable type: 'list'"
  ]
}
```

失败已复现

on_evaluation 在每轮评分后收到 RubricEvaluation

它记录 Grader Pass，不负责改变控制流




```
evaluations: list[RubricEvaluation] = []

def record_evaluation(evaluation):
    run_id = evaluation["grading_run_id"]
    evaluations.append(evaluation)
```

RubricEvaluation

- grading_run_id
- iteration（从 0 开始）
- result
- explanation
- criteria[].gap

Callback（回调）用于观察，不负责中止循环或改写 Verdict

用五个参数构造 RubricMiddleware

只有 model 必填，其余四项按需覆盖默认行为

```
rubric_middleware = RubricMiddleware(
    model=grader_model,
    system_prompt=(
        "You are a strict code grader. "
        "Always obtain current test evidence."
    ),
    tools=[run_test_suite],
    max_iterations=3,
    on_evaluation=record_evaluation,
)
```

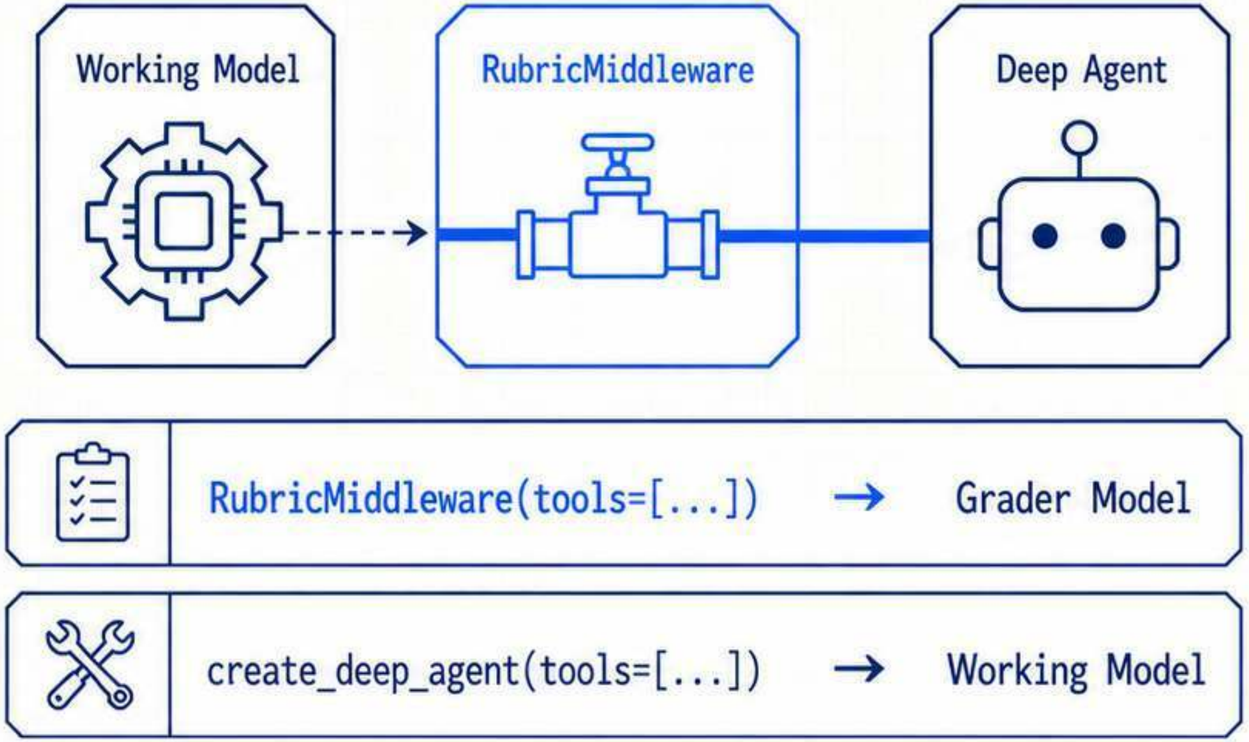
1. model：必填 · 谁来评分
2. system_prompt：可选 · 怎样评分
3. tools：可选 · 怎样取证
4. max_iterations：可选 · 默认 3
5. on_evaluation：可选 · 记录结论

⚠ tools=[run_test_suite] 只供 Grader Model 使用

把 Middleware 挂进 Agent 才会生效

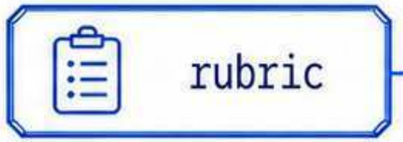
创建实例还不够，真正的挂载点是 middleware=[rubric_middleware]

```
agent = create_deep_agent(
    model=working_model,
    system_prompt=(
        "Return only executable Python source. "
        "Revise when the grader reports a gap."
    ),
    middleware=[rubric_middleware],
    checkpointer=InMemorySaver(),
)
```



首次 invoke 显式传入 rubric，启动评分循环

同一 thread_id 可延续已有 Rubric；新任务使用新的 thread_id

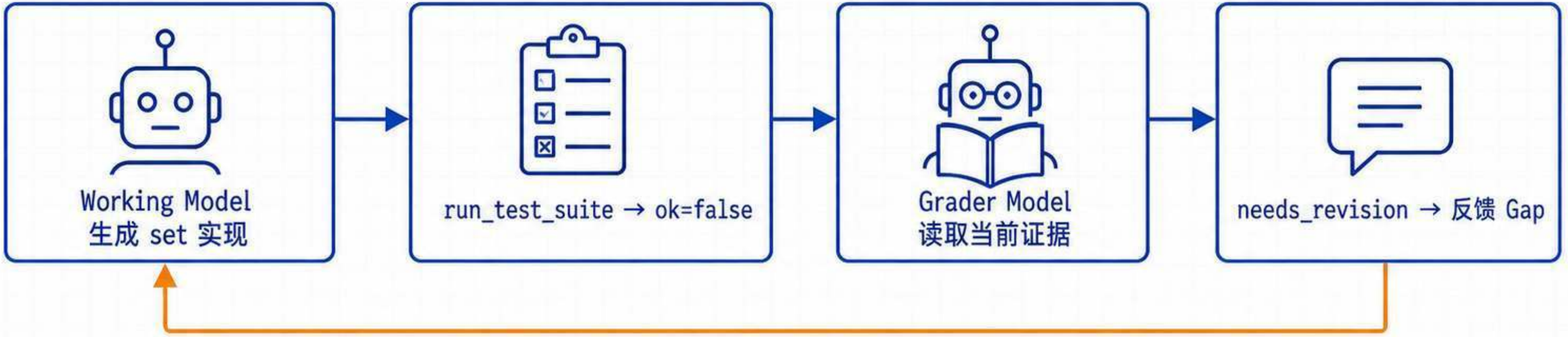



```
config = {
    "configurable": {"thread_id": "ch13-rubric-case"}
}

evaluation_start = len(evaluations)

result = agent.invoke(
    {
        "messages": [HumanMessage(content=task)],
        "rubric": rubric,
    },
    config=config,
)

run_evaluations = evaluations[evaluation_start:]
```

首次调用或新任务 → 显式传入非空 rubric

第一轮失败后，needs_revision 会触发修订

真正回到 Working Model 的是可执行 Gap



代表性运行

```
iteration 0: needs_revision
gap: Unhashable values are supported
      — test_unhashable raised TypeError
```

避免依赖 Hash，改用相等性比较

真实模型也可能第一轮直接通过

修订后，证据必须与当前候选对应

本例重新运行 run_test_suite，确认不可哈希输入已经通过

旧候选 + 旧证据

```
seen = set()
seen.add(value)
```

↓

```
values = [[1], [1]]
test_unhashable → TypeError
```

旧证据不能证明新候选正确

当前候选 + 当前证据

```
def contains(items, target):
    return any(item == target for item in items)

seen = []
if contains(seen, value):
    duplicates.append(value)
```

→ run_test_suite

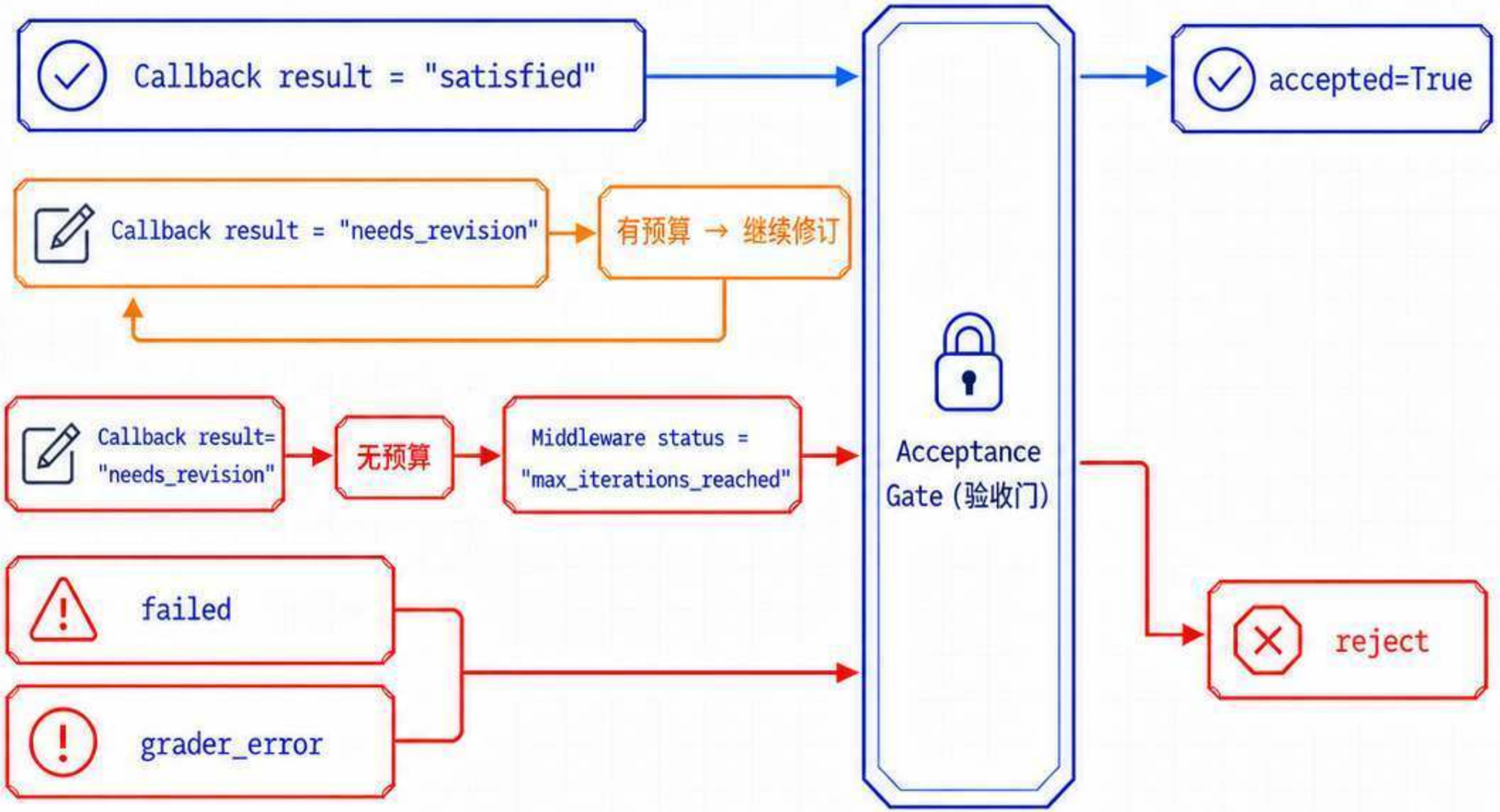
```
{'ok': True, 'failures': []}
```

→ iteration 1: satisfied ✓

Callback 记录本轮 Verdict，验收门决定是否放行

needs_revision 可能耗尽预算；只有 satisfied 才能 accepted=True

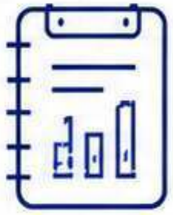
```
accepted = (
    final_evaluation is not None
    and final_evaluation["result"] == "satisfied"
)
```



达到上限时，Callback 中的 needs_revision 不会被改写

观察、状态延续与放行各有独立入口

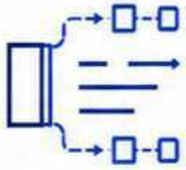
Callback 看结果，事件流看进度，Checkpoint 续状态，验收门做决定



on_evaluation

每轮评分后

日志与指标



stream.custom

评分开始前与结束后

实时进度

Checkpointer

相同 thread_id

延续任务状态

应用验收门

整次调用返回后

只有 satisfied 放行

非空 rubric

没有评分

Evidence Tool

工具未调用

Criterion 与 Gap

一直修订

最终 Verdict

有答案未放行

用 AgentSeek Template 体验 Rubric Middleware

推荐先跑无密钥演示，再连接真实模型观察完整评分回环

安装与启动

01 先安装 AgentSeek

```
$ uv tool install agentseek
```

02 创建 langchain/rubric 模板

```
$ agentseek create langchain/rubric --checkout main
$ cd rubric_lab
```

03 无密钥验证并启动

```
$ cp .env.example .env
$ agentseek task sync
$ agentseek task frontend
$ agentseek task rubric-smoke
$ agentseek dev
```

模板核心内容

1 Guided Demo（引导演示）

无需模型密钥，复现失败候选 → 证据 → 反馈 → 修订 → 通过

失败候选 (Candidate) → 证据 (Evidence) → 反馈 (Feedback) → 修订 (Revision)

2 Live Model（真实模型）

分别配置 Working Model（工作模型）与 Grader Model（评分模型）

- Working Model（工作模型）
- Grader Model（评分模型）

3 Evidence Loop（证据回环）

Evidence Tool（证据工具）绑定当前候选，评分结果可追溯

Evidence Tool（证据工具） → 绑定当前候选（当前版本） → 评分结果（可追溯）

4 Acceptance Gate（验收门禁）

satisfied + 当前候选的 Evidence 通过，才显示 Accepted

受限子进程不是 Isolated Sandbox ； Live Model 凭据只放在服务端 .env