Yao Meta Skill 2.0 正式升级方案

主题：从 Meta Skill Factory 升级为 Skill OS

文档版本：1.0

日期：2026-06-12

适用对象：Yao Meta Skill 维护者、贡献者、使用该体系构建团队级 agent skills 的工程团队

审阅范围：公开仓库 README、SKILL.md、核心方法论文件、平台能力矩阵，以及 Anthropic、OpenAI、Agent Skills、VS Code 等公开资料。

> *说明：本方案基于公开仓库和官方公开资料完成。未在本地复跑仓库 CI，仓库中的指标按公开 README 与 reports 描述引用。正式发布前建议固定 commit 后复跑* `make test`*、packaging suite、trigger eval、output eval 与 conformance suite。*

1. 执行摘要

Yao Meta Skill 1.x 已经具备强工程底座：它把 skill 创建从“写一份好说明”推进到“有触发边界、有评测、有治理、有打包、有可移植语义”的工程系统。公开 README 将项目定位为创建、评估、打包和治理可复用 agent skills 的轻量严谨系统，并强调从 workflow、transcript、prompt、notes、runbook 生成 reusable skill packages。[R1]

2.0 的关键升级方向建议定义为：**把 Yao Meta Skill 从 Meta Skill Factory 升级为 Skill OS**。它的目标是让团队可以把重复工作沉淀为可安装、可评测、可分发、可治理、可观测的 agent skill package。

2.0 的主线不应扩写根 `SKILL.md`。根 `SKILL.md` 已明确要求通过 frontmatter description 路由，保持入口精简，把指导放入 `references/`，把逻辑放入 `scripts/`，把证据放入 `reports/`。[R2] 2.0 应延续这个原则，把复杂度放进 Skill IR、Output Eval Lab、Runtime Conformance、Trust Layer、Registry、Review Studio、Skill Atlas 与 Telemetry Loop。

最高优先级建议：先做 Output Eval Lab 和 Skill IR。Trigger 能命中只是入口质量，真正商业价值来自“用了 skill 后输出是否更好、更稳、更省成本”。Skill IR 则让 Yao 的核心语义先形成平台无关资产，再编译到 OpenAI、Claude、Agent Skills、VS Code / Copilot 与 generic targets。

2. 当前整体 Review

2.1 当前强项

维度	当前表现	2.0 价值
方法论	公开方法论已经包含是否应创建 skill、intent dialogue、最小 archetype、boundary design、trigger-first authoring、gate	可以继续作为 2.0 的“技能工程宪法”。

	selection 与 promotion policy。[R3]	
触发评测	README 报告 regression corpus 为 66 个 prompts、21 个 families，并给出 0 false positives、0 false negatives、precision 1.0、recall 1.0 的项目内结果。[R1]	触发层已强，应把输出质量层拉到同等强度。
资源边界	根 `SKILL.md` 保持极简，正文只保留路由、模式、紧凑流程和输出契约。[R2]	2.0 可继续通过 progressive disclosure 控制上下文成本。
治理	README 已强调 owner、lifecycle state、review cadence、maturity score、promotion decisions、regression history 等治理资产。[R1]	可以升级为 team skill portfolio 的管理面。
可移植	当前 matrix 支持 OpenAI、Claude、generic 三类 target，保留 activation、execution、trust、degradation 语义。[R4]	2.0 可升级为 conformance-first 的多端编译体系。
本地可靠性	README 暴露 `make test`、`scripts/yao.py`、packaging、governance 与 resource-boundary checks。[R1]	有条件把 2.0 做成可验收的 CLI 产品。

2.2 当前主要缺口

缺口	影响	建议补强
输出质量评测仍偏轻	目前已有 baseline compare，但还需要系统化 with-skill / without-skill、v1 / v2、blind A/B、assertion grading、人工 review。	新增 Output Eval Lab，成为 production 及以上模式的发布门槛。
benchmark 可信度需增强	README 的 weighted benchmark 有传播力，但需要公开方法、样本、权重、失败样本、复现脚本。	新增 `reports/benchmark_methodology.md` 和 public benchmark harness。
portability 偏 packaging check	当前 matrix 明确列出尚未实现 runtime-specific behavior transforms、client SDK integration、provider-specific execution logic。[R4]	新增 Runtime Conformance Suite，验证加载、路由、metadata、相对路径、fallback、权限。
安全供应链层不足	skill 可能包含 scripts、assets、references，团队分发后风险提升。	新增 Trust & Security Layer，覆盖依赖锁定、脚本接口、权限声明、secret scan、hash。
多 skill 组合治理不足	单个 skill 能做强，但大型团队会遇到 route collision、stale skill、重复脚本、owner 缺失。	新增 Skill Atlas，形成 catalog、route overlap、dependency graph、no-route opportunity。
Review UX 仍是明显短板	README 自评 onboarding/review 为 6.5，并说明这是最清楚的 UX 改进区。[R1]	升级 Review Studio 2.0，把评测、信任、conformance、release note 统一展示。

2.3 对外部一流实践的吸收

Anthropic Agent Skills 的核心设计是 progressive disclosure：启动时只加载 name 和 description，触发后再读取完整 `SKILL.md`，必要时再读取 references、scripts、resources。[R5] 这与 Yao 当前“lean entrypoint + references/scripts/reports”的原则高度一致。

Anthropic Skill Creator 的领先点在 eval-driven loop：创建真实 test prompts，同时运行 with-skill 与 baseline，记录 timing、tokens、assertion grading、benchmark delta，并通过 viewer 做人类 review。[R6] 这是 Yao 2.0 最应该系统吸收的部分。

OpenAI Codex Skills 的关键分层是：skills 是 reusable workflow 的 authoring format，plugins 是可安装分发单元；Codex 通过 name、description、file path 做渐进加载，并对技能 catalog 的上下文预算设置上限。[R7] 这支持 Yao 2.0 引入 registry、plugin metadata、version pinning 与 distribution contracts。

Agent Skills open standard 已经明确了 `SKILL.md` frontmatter、name 与 description 约束、optional directories、scripts、references、assets、progressive disclosure、validation 等规格。[R8] Yao 2.0 应成为标准兼容的 compiler、validator 与 governance layer。

Agent Skills evaluation guide 强调结构化 eval 的价值：用真实 prompts、expected output、input files、with-skill / without-skill baseline、timing、tokens、assertions、grading、benchmark delta、人类 review 与迭代闭环来判断 skill 是否可靠。[R9]

Agent Skills scripts guide 强调脚本要版本固定、复杂命令沉入 scripts、相对路径引用、自包含依赖、避免交互阻塞、支持 `--help`、清晰错误信息与结构化输出。[R10] 这应进入 Yao 2.0 的 trust report 与 script lint。

VS Code / Copilot 对 Agent Skills 的支持说明 skills 已经成为多客户端开放格式，要求 name 匹配目录、description 最大 1024 字符，并建议审查共享 skills 的安全标准。[R13] 这进一步说明 Yao 2.0 应把 cross-client conformance 和 trust gating 作为一等能力。

3. 2.0 战略定位

3.1 一句话定位

Yao Meta Skill 2.0 是一个 Skill OS：把重复工作转成可安装、可评测、可分发、可治理、可观测的 agent skill package。

3.2 产品边界

2.0 应专注于四件事：

1. **Capture**：从 prompt、workflow、transcript、文档、runbook、团队 SOP 中提炼稳定可复用能力。
2. **Compile**：用 Skill IR 保存平台无关语义，再编译到 OpenAI、Claude、Agent Skills、VS Code / Copilot、generic targets。

3. **Verify**：通过 trigger eval、output eval、runtime conformance、trust/security、context budget、governance gates 验证质量。
4. **Operate**：通过 registry、review studio、telemetry、drift report、release note 支撑团队长期使用。

3.3 2.0 北极星指标

指标	定义	目标
Skill value delta	with-skill 相比 baseline 的输出通过率提升	production skill 应为正向且可解释
Trigger reliability	should-trigger 与 should-not-trigger 的 precision / recall	不低于 1.x 基线
Runtime compatibility	目标平台 conformance pass rate	主要 target 100% 通过结构检查
Trust readiness	脚本、依赖、权限、secret、package hash 检查结果	governed skill 无高风险项
Review efficiency	人类 reviewer 完成首轮审查所需步骤	一个 Review Studio 页面完成主审查
Drift control	过期、冲突、低采用、低质量 skill 的发现率	atlas 与 telemetry 可定位具体对象

4. 2.0 目标架构

```
Raw workflow / prompt / transcript / SOP / repo context
        |
        v
Intent Dialogue + Reference Scan + Qualification Gate
        |
        v
Skill IR: platform-neutral capability contract
        |
        +--> Trigger Eval Lab
        +--> Output Eval Lab
        +--> Trust & Security Check
        +--> Runtime Conformance Suite
        +--> Context Budget + Resource Boundary Check
        |
        v
Compilers / Adapters
        |
        +--> OpenAI / Codex target
        +--> Claude target
        +--> Agent Skills standard target
        +--> VS Code / Copilot target
        +--> Generic target
        |
        v
Registry + Package + Review Studio + Telemetry Loop
```

2.0 的架构重点是：**先把 skill 表达成 Skill IR，再通过编译器生成不同平台包，并用统一评测与治理体系验收。** 这样即使某个平台修改 frontmatter、安装路径、plugin manifest 或 activation policy，Yao 的核心语义仍然稳定。

5. 2.0 核心模块设计

5.1 Skill IR：平台无关中间表示

目标：把 skill 的真实语义从目录和平台细节中抽离出来，成为可测试、可迁移、可编译的核心对象。

建议目录：

```
skill-ir/
  schema.json
  examples/
  migrations/
  README.md
```

最小字段建议：

```
{
  "schema_version": "2.0.0",
  "name": "example-skill",
  "job_to_be_done": "...",
  "trigger_surface": {
    "description": "...",
    "should_trigger": [],
    "should_not_trigger": [],
    "edge_cases": []
  },
  "workflow": {
    "steps": [],
    "decision_points": [],
    "failure_modes": []
  },
  "resources": {
    "references": [],
    "scripts": [],
    "assets": []
  },
  "eval_plan": {
    "trigger": [],
    "output": [],
    "adversarial": [],
    "baseline": "without_skill"
  },
  "risk": {
    "output_risk": "low|medium|high",
    "execution_risk": "low|medium|high",
    "trust_boundary": "personal|team|external"
  },
  "governance": {
    "owner": "",
    "maturity": "scaffold|production|library|governed",
    "review_cadence": "",
    "review_due": ""
  }
}
```

验收标准：

- 现有 root skill 可导出为 Skill IR。
- 2 到 3 个 examples 可从 IR 编译回 `SKILL.md`、`agents/interface.yaml` 与 target metadata。
- CI 检查 schema validity。
- 编译前后 description、job、exclusions、resources、governance 字段语义一致。

5.2 Output Eval Lab：证明 skill 真的提升结果

目标：把“用了 skill 是否更好”变成可复现证据。

建议目录：

```
evals/output/
  schema.json
  cases.jsonl
  rubrics/
  assertions/
  fixtures/
  results/
```

评测类型：

类型	作用	输出
with-skill vs without-skill	证明 skill 增益	delta report
v1 vs v2	证明版本升级有效	upgrade report
assertion grading	验证客观条件	grading.json
blind A/B	降低作者偏见	comparison report
human review	捕捉主观质量	feedback.json
cost / token / time	衡量质量成本	efficiency report
failure taxonomy	沉淀失败原因	failure summary

建议 release gate：production 及以上模式必须至少包含 3 个 output eval cases；library 与 governed 模式至少包含 5 个，并覆盖一个边界案例、一个 near-neighbor 案例、一个真实文件或真实输出约束案例。

验收标准：

- `scripts/run_output_eval.py` 能读取 cases、执行 baseline、输出 benchmark delta。
- `reports/output_quality_scorecard.md` 自动生成。
- 至少一个 example skill 展示 with-skill pass rate 高于 baseline。
- 每个失败项绑定 failure taxonomy 和下一步修复建议。

5.3 Runtime Conformance Suite：从“能打包”升级到“能被客户端正确消费”

目标：确保每个 target 的输出包可被对应客户端识别、加载、降级和审查。

建议目录：

```
runtime/
  conformance/
    schema.json
    openai_cases.json
```

```
    claude_cases.json
    agentskills_cases.json
    vscode_cases.json
    generic_cases.json
```

检查项：

5. 文件结构是否满足目标平台路径约定。
6. frontmatter 字段是否满足 name、description、compatibility、metadata 约束。
7. description 是否在 1024 字符以内，并包含能力与使用场景。
8. 相对路径引用是否可达。
9. scripts、references、assets 是否被正确声明。
10. 手动激活与自动激活语义是否可表达。
11. 无法表达的平台特性是否写入 degradation note。
12. package hash、version、license、owner 是否完整。

验收标准：

- OpenAI、Claude、Agent Skills、generic 至少四类 target 通过结构与 metadata conformance。
- VS Code / Copilot target 可作为 stretch goal，纳入 name-directory match、description length、project/user scope 检查。
- 对 unsupported feature 输出清晰 fallback，而非静默丢失。

5.4 Trust & Security Layer：脚本和包的供应链安全

目标：让团队能够安全审查和安装 skills，尤其是带 scripts 的 package。

建议目录：

```
security/
  trust_policy.md
  script_policy.md
  dependency_policy.md
  secret_policy.md
  package_integrity.md
```

检查项：

类型	检查	建议门槛
脚本接口	是否支持 `--help`、非交互、清晰错误信息	production 必须通过
依赖	是否 pin 版本，是否存在 lock 或 inline metadata	library 及以上必须通过
权限	是否声明网络、文件写入、shell、外部工具	governed 必须通过
secret	是否出现 token、key、credential、私有路径	任何模式不得有高风险泄漏
package	是否生成 SHA256、manifest、版	team distribution 必须通过

	本、license	
remote code	是否远程拉取或执行代码	默认高风险，需人工批准

验收标准：

- `scripts/trust_check.py` 输出 `reports/security_trust_report.md`。
- 带 scripts 的 example skill 至少通过 `--help`、non-interactive、dependency pin、secret scan。
- 高风险项阻断 governed release。

5.5 Skill Atlas：管理多 skill 生态

目标：让 Yao 能管理整个团队的 skill portfolio，发现冲突、重复、过期和新机会。

建议目录：

```
skill_atlas/
  catalog.json
  route_overlap_matrix.csv
  dependency_graph.json
  stale_skills.json
  no_route_opportunities.json
  owner_review_gaps.json
```

核心能力：

- catalog：列出所有 skill 的 name、description、owner、maturity、targets、review due。
- route overlap：检测 description 和 trigger cases 的相似度冲突。
- dependency graph：找出共享 scripts、references、assets 的依赖关系。
- no-route opportunities：从失败或未命中 prompts 发现新 skill 候选。
- stale report：发现 review overdue、低采用、无 owner、长期失败的 skill。

验收标准：

- 可读取一个 workspace 下多个 skill。
- 生成 HTML overview 和 JSON artifacts。
- 至少能报告 route collision、stale skill、missing owner 三类问题。

5.6 Registry & Distribution：从目录包到可安装产品

目标：让 Yao 生成可安装、可版本固定、可升级、可审查的 skill distribution。

建议目录：

```
registry/
  index.schema.json
  package.schema.json
  packages/
  examples/
```

package metadata 建议：

```
{
  "name": "example-skill",
  "version": "2.0.0",
  "description": "...",
  "targets": ["openai", "claude", "agentskills", "generic"],
  "maturity": "production",
  "owner": "Yao Team",
  "review_cadence": "quarterly",
  "trust_level": "team",
  "checksums": {
    "package_sha256": "..."
  },
  "compatibility": {
    "openai": "pass",
    "claude": "pass",
    "agentskills": "pass",
    "generic": "pass"
  }
}
```

建议 CLI：

```
python3 scripts/yao.py dist my-skill --targets openai claude agentskills generic
python3 scripts/yao.py registry add my-skill
python3 scripts/yao.py registry audit
python3 scripts/yao.py upgrade-check my-skill
```

验收标准：

- 输出 zip、manifest、hash、target folders、release notes。
- 支持 version bump、upgrade diff、breaking change note。
- Registry audit 能发现缺失版本、hash、owner、review cadence、license。

5.7 Review Studio 2.0：把审查体验产品化

目标：把现在的 HTML overview 与 side-by-side viewer 升级成一个完整审查面。

页面模块：

模块	内容
Intent Canvas	job、inputs、outputs、exclusions、constraints、success standard
Trigger Lab	should-trigger、should-not-trigger、edge cases、route conflicts
Output Lab	with-skill / baseline、v1 / v2、assertions、human feedback
Context Budget	SKILL.md、references、scripts 的预算与质量密度
Runtime Matrix	OpenAI、Claude、Agent Skills、generic、VS Code target pass/fail
Trust Report	scripts、dependencies、permissions、secrets、package integrity

Release Notes	版本差异、breaking changes、migration guide、known limitations

验收标准：

- 一个 HTML 页面能完成 production skill 首轮审查。
- Reviewer 可看到证据路径和失败原因。
- 所有红色 blocker 都可跳转到具体 report 或 source file。

5.8 Telemetry & Drift Loop：从一次性生成到持续运营

目标：把真实使用反馈变成下一轮 skill 迭代信号。

事件 schema 建议：

```
{
  "event": "skill_activation",
  "skill": "example-skill",
  "version": "2.0.0",
  "activation_type": "implicit|explicit",
  "outcome": "accepted|edited|rejected|missed",
  "failure_type": "wrong_trigger|under_trigger|bad_output|missing_resource|script_error",
  "timestamp": "2026-06-12T00:00:00Z"
}
```

隐私原则：默认 local-first、可关闭、仅记录必要 metadata、避免记录完整用户内容。需要跨团队聚合时先脱敏。

验收标准：

- 能生成 adoption、missed-trigger、bad-output、script-error、review-overdue 报告。
- Telemetry 与 Skill Atlas 联动，自动提出 stale skill 与 next iteration candidates。

6. 建议新增目录结构

```
yao-meta-skill/
  SKILL.md
  manifest.json
  agents/
    interface.yaml
    openai.yaml
    claude.yaml


  skill-ir/
    schema.json
    examples/
    migrations/


  runtime/
    conformance/
    openai/
    claude/
    agentskills/
    vscode/
    generic/
```

```
evals/
  trigger/
  output/
  adversarial/
  confusion/
  conformance/
  security/


skill_atlas/
  catalog.json
  route_overlap_matrix.csv
  dependency_graph.json


registry/
  index.schema.json
  package.schema.json
  examples/


security/
  trust_policy.md
  script_policy.md
  dependency_policy.md
  secret_policy.md


reports/
  output_quality_scorecard.md
  conformance_matrix.md
  security_trust_report.md
  skill_atlas.html
  benchmark_methodology.md
  adoption_drift_report.md


scripts/
  yao.py
  compile_skill.py
  run_output_eval.py
  run_conformance_suite.py
  build_skill_atlas.py
  trust_check.py
  package_plugin.py
  verify_package.py
  registry_audit.py
  collect_feedback.py


docs/
  migration-v2.md
  skill-os-architecture.md
  registry-guide.md
  runtime-targets.md
```

7. 根 SKILL.md 的 2.0 改法

原则：根 `SKILL.md` 继续保持轻，不把 2.0 的全部细节写进去。只新增 6 条高价值路由规则，并把细节放进 references。

建议加入：

```
## 2.0 Routing Additions


For production, library, governed, or team-distributed skills:
1. Create or update Skill IR before platform-specific packaging.
```

```
2. Run both trigger eval and output eval.
3. Check Skill Atlas for route overlap, stale skills, dependency conflicts,
   and no-route opportunities when a skill library is present.
4. Validate runtime conformance for requested targets.
5. Produce a trust/security report when scripts, external files, network
   access, or team distribution are involved.
6. Emit registry/package metadata when reuse, installation, or upgrade is
   part of the goal.
```

建议新增 references：

```
references/skill-ir-method.md
references/output-eval-method.md
references/runtime-conformance-method.md
references/skill-atlas-method.md
references/trust-security-method.md
references/distribution-registry-method.md
references/review-studio-method.md
references/telemetry-drift-method.md
```

8. 2.0 评分类目与发布门槛

8.1 新评分模型

维度	权重	评价重点
Trigger reliability	15	should-trigger、should-not-trigger、edge、confusion
Output effectiveness	20	with/baseline delta、v1/v2 delta、rubric、assertions
Context efficiency	10	root budget、progressive disclosure、resource split
Runtime conformance	10	OpenAI、Claude、Agent Skills、VS Code、generic
Portability & distribution	10	packag e、plugin、registry、version、upgrade
Security & trust	10	script s、dependencies、permissions、secrets 、hash
Governance & drift	10	owne r、maturity、review、deprecation、telem etry
UX & adoption	10	quickstart、review studio、examples、team handoff
Evidence reproducibility	5	methodology、fixtures、failure cases、external review

8.2 分层发布门槛

Tier	最低门槛
Scaffold	structure validation、context budget、basic trigger smoke test
Production	scaffold + trigger eval + output eval + resource boundary check
Library	production + Skill Atlas scan + runtime conformance + package metadata
Governed	library + trust report + owner/review cadence + promotion evidence
Public world-class	governed + public benchmark methodology + reproducible results + failure disclosure

9. 90 天实施路线图

以 2026-06-12 为起点，建议拆成 6 个阶段。

阶段	日期	目标	主要交付物
Phase 0	06-12 至 06-18	冻结 1.x 基线	tag、v1 baseline report、benchmark methodology draft
Phase 1	06-19 至 07-05	Output Eval Lab	schema、runner、scorecard、example with/baseline report
Phase 2	07-06 至 07-25	Skill IR + Compiler	IR schema、examples、compile_skill.py、migration guide
Phase 3	07-26 至 08-10	Runtime + Security	conformance suite、trust_check.py、security reports
Phase 4	08-11 至 08-28	Skill Atlas + Registry	atlas generator、registry schema、package metadata、audit command
Phase 5	08-29 至 09-10	Review Studio + release	unified HTML studio、release checklist、public v2 docs
Buffer	09-11 至 09-12	发布修正	docs polish、failure cases、demo walkthrough

10. 建议优先落地的 12 个 PR

PR	标题	验收标准
1	`reports/benchmark_methodology.md`	明确权重、样本、对标对象、失败披露、复现命令
2	Output eval schema	`evals/output/schema.json` 通过 CI 校验
3	Output eval runner	生成 with-skill / baseline delta report
4	Output quality scorecard	自动输出 `reports/output_quality_scorecard.md`
5	Skill IR v0	root skill 与至少两个 examples 可导出 IR
6	Compiler refactor	OpenAI、Claude、generic 从 IR 生成
7	Agent Skills conformance	增加 standard validator 与 conformance cases
8	Trust check	检查脚本 `--help`、non-interactive、dependency pin、secret risk
9	Skill Atlas generator	输出 catalog、route overlap、dependency graph、stale report
10	Registry package format	输出 package metadata、hash、version、compatibility matrix
11	Review Studio 2.0	一个 HTML 页面整合 trigge r、output、conformance、trust、release notes
12	Migration v2 docs	提供 1.x 到 2.0 的迁移指南和 breaking changes

11. 关键文件建议

11.1 `reports/benchmark_methodology.md`

必须说明：

- benchmark 类型：self-eval、internal blind eval、external review。
- 样本来源：公开 fixture、真实匿名案例、失败库、near-neighbor prompts。
- 评价维度：trigger、output、context、runtime、trust、governance、UX。
- 对标边界：不同平台 runtime 能力不一致时不做直接强比较。
- 失败披露：每次 release 至少公开典型失败样本和修复方向。

- 复现方式：固定 commit、命令、model、fixtures、expected artifacts。

11.2 `references/output-eval-method.md`

必须包含：

- 如何写真实 prompts。
- 如何设置 expected output。
- 何时写 assertions。
- 如何做 with-skill / without-skill。
- 如何做 v1 / v2。
- 如何看 pass rate、time、token、delta。
- 如何避免过拟合到少数 eval cases。

11.3 `references/trust-security-method.md`

必须包含：

- 什么情况下必须做 trust report。
- scripts 的最小接口标准。
- dependency pin 与 lock policy。
- allowed-tools / permissions 的声明策略。
- secret scan 与 package hash。
- 高风险项的人工批准规则。

11.4 `docs/migration-v2.md`

必须包含：

- 1.x skill 如何生成 Skill IR。
- 旧 adapter 如何迁移到 compiler。
- 旧 baseline compare 如何迁移到 Output Eval Lab。
- 旧 governance report 如何接入 Registry 与 Skill Atlas。
- 兼容策略和废弃时间表。

12. 风险与缓解

风险	表现	缓解
过度工程化	Scaffold skill 被迫携带大量报告	gate selection 保持按风险启用，低风险只做最小检查
Eval 过拟合	几个样本变好，真实任务无提升	增加 blind、adversarial、near-neighbor、failure library、holdout
Benchmark 可信度不足	weighted score 被质疑	公布 methodology、fixtures、失败样本、

		复现命令
平台规格变化	adapters 很快落后	使用 Skill IR + conformance tests + target compatibility matrix
脚本安全风险	团队安装后执行未知脚本	trust_check、权限声明、secret scan、hash、人工批准
UX 复杂	Reviewer 看报告负担过重	Review Studio 聚合红绿灯、证据路径、blocking issues
Telemetry 敏感	用户担心内容泄露	local-first、metadata-only、可关闭、脱敏聚合

13. 立即可执行的 30 天计划

第 1 周：冻结与公开基线

13. 打 tag，记录当前版本与 commit。
14. 生成 `reports/v1_baseline.md`。
15. 新增 `reports/benchmark_methodology.md` 草案。
16. 把 README 当前 benchmark 标注为 project-level self-eval。
17. 建立 v2 milestone 和 PR labels：`v2-output-eval`、`v2-ir`、`v2-conformance`、`v2-trust`、`v2-registry`。

第 2 周：Output Eval Lab v0

18. 新增 `evals/output/schema.json`。
19. 新增 3 个 output eval cases。
20. 新增 `scripts/run_output_eval.py`。
21. 新增 `reports/output_quality_scorecard.md`。
22. 更新 `scripts/yao.py validate`，让 production 模式提示 output eval 状态。

第 3 周：Skill IR v0

23. 新增 `skill-ir/schema.json`。
24. 从 root skill 生成 `skill-ir/examples/yao-meta-skill.json`。
25. 新增 `scripts/compile_skill.py`。
26. 让 OpenAI、Claude、generic package 从 IR 读取核心字段。
27. 新增 `docs/migration-v2.md` 初稿。

第 4 周：Conformance 与 Trust v0

28. 新增 `runtime/conformance/schema.json`。

29. 新增 `scripts/run_conformance_suite.py`。
30. 新增 `scripts/trust_check.py`。
31. 输出 `reports/conformance_matrix.md` 和 `reports/security_trust_report.md`。
32. 在 CI 中加入 v2 smoke checks。

14. 对 README 的建议叙事

建议 2.0 README 开头改成更产品化的结构：

```
# Yao Meta Skill 2.0

Turn repeated work into governed, evaluated, portable agent skills.

Yao helps teams:
1. Capture real workflows.
2. Compile them into portable skill packages.
3. Evaluate trigger and output quality.
4. Validate runtime compatibility.
5. Package and distribute skills.
6. Govern drift, ownership, trust, and reuse.
```

建议首页架构图：

```
Workflow / Prompt / Transcript / SOP
        |
        v
Intent Dialogue
        |
        v
Skill IR
        |
        v
Trigger Eval + Output Eval + Trust Check + Conformance
        |
        v
OpenAI / Claude / Agent Skills / VS Code / Generic
        |
        v
Registry + Review Studio + Governance + Telemetry
```

15. 最终建议

Yao Meta Skill 2.0 的竞争力不应只来自“生成更好的 `SKILL.md`”。官方 creator 和社区模板会持续进步，单点写作能力很快会变成基础能力。

真正的护城河应集中在五个方面：

33. **评测护城河**：trigger eval + output eval + baseline + blind + adversarial + failure taxonomy。
34. **治理护城河**：owner、maturity、review、promotion、deprecation、drift。
35. **跨平台护城河**：Skill IR + compiler + runtime conformance。
36. **团队采用护城河**：registry、review studio、trust report、upgrade path。
37. **证据护城河**：每个升级有 scorecard，每次 release 有 methodology，每个失败有归因。

2.0 的第一原则可以写成：

> *每个新增结构都必须让 skill 更可靠、更可迁移、更可审查，或更容易被团队长期复用。无法证明收益的结构，应留在 next iteration，而非进入根入口。*

16. 参考资料

[R1] Yao Meta Skill README.

https://raw.githubusercontent.com/yaojingang/yao-meta-skill/main/README.md

[R2] Yao Meta Skill SKILL.md.

https://raw.githubusercontent.com/yaojingang/yao-meta-skill/main/SKILL.md

[R3] Yao Meta Skill Skill Engineering Method.

https://raw.githubusercontent.com/yaojingang/yao-meta-skill/main/references/skill-engineering-method.md

[R4] Yao Meta Skill Platform Capability Matrix.

https://raw.githubusercontent.com/yaojingang/yao-meta-skill/main/references/platform-capability-matrix.md

[R5] Anthropic, Equipping agents for the real world with Agent Skills.

https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills

[R6] Anthropic Skill Creator SKILL.md.

https://raw.githubusercontent.com/anthropics/skills/main/skills/skill-creator/SKILL.md

[R7] OpenAI Developers, Agent Skills in Codex.

https://developers.openai.com/codex/skills

[R8] Agent Skills Specification.

https://agentskills.io/specification

[R9] Agent Skills, Evaluating skill output quality.

https://agentskills.io/skill-creation/evaluating-skills

[R10] Agent Skills, Using scripts in skills.

https://agentskills.io/skill-creation/using-scripts

[R11] Agent Skills, How to add skills support to your agent.

https://agentskills.io/client-implementation/adding-skills-support

[R12] OpenAI Academy, Plugins and skills.

https://openai.com/academy/codex-plugins-and-skills/

[R13] Visual Studio Code Docs, Use Agent Skills in VS Code.

https://code.visualstudio.com/docs/agent-customization/agent-skills