三份契约 · 一页版

产品定义 / 架构设计 / 质量控制

配套文章《工程方法论没死，落地法得重做》（公众号「小火慢炖出滋味」· 周一 · 中小企业那些事）

填法只有一条：每项一句话。写不出数字的项，就是还没想清楚，别交给智能体。

Progent Labs | progentlabs.com | 公众号「小火慢炖出滋味」
v1.0 | 2026-11-01 | 免费件，可转发

产品定义契约 · 一页版

配套《工程方法论没死，落地法得重做》 | 公众号「小火慢炖出滋味」

填法只有一条：每项一句话。写不出数字的项，就是还没想清楚，别交给智能体。

契约正文

项 | 填 写 内 容

要做什么 | F1【动词开头，一句话】； F2【】； F3【】（不超过三项）

不做什么 | N1【不接的需求，说明归谁接或不做】； N2【】

给谁用 | 角色【谁在用】； 场景【什么时候用】； 频次【每天 / 每周几次】

做到什么程度 | 性能【响应时间 / 并发数 / 数据量】； 安全【权限要求 / 加密要求】； 兼容【系统与版本范围】

约束 | 必须复用【已有系统 / 接口】； 必须遵守【行业或公司规范】； 不许动的数据【】

验收 | 谁来验【角色】； 按什么标准【可判定的条件】； 什么情况退回重做【】

交出去之前自检

□ 每一项都能被第三方读懂，不需要我在旁边解释

□ 「不做什么」写了具体请求，不是一句「按需支持」

□ 硬指标都是数字，没有「尽快」「尽量」「高可用」这类形容词

□ 验收标准可以当场判真假，不是「效果良好」

□ 三项功能之外的需求，都写进了「不做什么」

常见坑

· 只写「要做什么」，不写「不做什么」——智能体会自己补边界，补一次返一次工。

· 「做到什么程度」写成愿景——它会把「快」理解成任意一种快。

· 验收交给「看着差不多」——这是后面所有返工的源头。

架构契约 · 一页版

配套《工程方法论没死，落地法得重做》 | 公众号「小火慢炖出滋味」

与产品定义契约配套：那份说做什么，这份说怎么搭。分布式、多端系统，通信一项不能空。

契约正文

项 | 填 写 内 容

模块边界 | M1 职责【一句话】 / 不负责【】； M2 职责【】 / 不负责【】（模块之间不许共享内部数据）

数据模型 | 核心实体【】； 主键与唯一键【】； 实体关系【一对多 / 多对多】； 数据归属【哪个模块是唯一写入方】

通信契约 | 协议【HTTP / gRPC / 消息队列，含版本】； 密钥管理【生成、轮换、存放位置】； 加密与防篡改【算法 / 校验方式】； 两端场景【发起方做什么、接收方做什么、异常怎么回】

技术约束 | 选用【技术选型】及理由【】； 禁用【不许引入的库或服务】

兼容与升级 | 版本策略【向前兼容范围】； 回滚方式【】

评审 | 谁审【角色】； 审什么【模块边界 / 数据模型 / 通信三项】； 什么情况驳回【】

交出去之前自检

□ 每个模块都写了「不负责什么」，不只是「负责什么」

□ 每个数据只有一个写入方，没有两处都能改

□ 通信写到了字段级：谁发起、谁接收、失败怎么回

□ 密钥的轮换和存放写清了，不是「用的时候再说」

□ 分布式或跨端的地方，两端场景都写了，不是只写一端

常见坑

· 通信只写「用 HTTP」——联调时两端对报文字段的理解会分叉，人和智能体都猜。

· 数据模型只画实体不写归属——后面必然出现两处写同一张表。

· 模块边界靠口头约定——智能体会跨模块改，改完局部对、整体崩。

质量契约 · 一页版

配套《工程方法论没死，落地法得重做》 | 公众号「小火慢炖出滋味」

质量不是最后测出来的，是从任务拆解那天起就立好的闸门。这份契约管三件事：用例从哪来、闸门看什么、谁签字放行。

契约正文

项 | 填 写 内 容

用例来源 | 每条用例标注它盖的是哪条契约：例 T01 ← F1；T02 ← M1；T03 ← 通信异常【】

四道闸门 | 自测【谁执行 / 看什么】；集成【谁执行 / 看什么】；系统【谁执行 / 看什么（含新老数据、升级、压力、安全）】；发布【谁执行 / 看什么】

准出指标 | 缺陷水位【未关闭 P0 / P1 数】；用例覆盖【核心功能覆盖比例】；回归通过率【%】；性能达标【响应时间上限】

缺陷流转 | 发现【谁提】→ 修【谁改】→ 回归【谁验】→ 关闭【谁确认】；同一模块重复出现【几次】即回查架构契约

放行权限 | 谁签字【角色，不是智能体】；一票否决的情形【列出】；谁有权紧急放行【】

回流 | 超出产品定义范围的回【产品定义契约】；范围内的回【开发】；谁判定【】

交出去之前自检

□ 每条用例都能回溯到产品定义或架构契约的某一条，没有凭空的用例

□ 四道闸门都写了「谁执行」，不是只有「要测」

□ 准出指标是数字，不是「质量达标」

□ 放行是人的签字，智能体最多提交待放行

□ 缺陷反复出现在同一模块时，有回查架构的动作

常见坑

· 把「测试通过」当准出——测试通过只说明用例跑完了，不说明边界守住了。

· 让智能体自己放行——它没有担责能力，出了事找不到人。

· 用例只覆盖功能，不覆盖通信异常与升级——线上出事多半在这两处。