ICS［点击此处添加 ICS 号］
［点击此处添加中国标准文献分类号］

YD

中 华 人 民 共 和 国 通 信 行 业 标 准

YD/T［×××××］—［××××］
［代替 YD/T］

可信开源社区评估规范 第 1 部分： 总体框架

Assessment criteria of trustworthy open source community--Part 1: General framework

［点击此处添加与国际标准一致性程度的标识］

（征求意见稿）

2021.5.13

［××××］-［××］-［××］发布　　［××××］-［××］-［××］实施

中华人民共和国工业和信息化部 发 布

目　　次

前　言

近些年开源已经成为新兴技术领域的主流技术，一个好的开源社区有助于开源项目营造良好的开源生态并扩大影响力。本标准将从社区治理、社区运营、社区开发、基础设施等角度，树立开源社区应关注的内容及指标，聚焦于如何营造健康可信的开源社区。

本标准/本部分按照 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由中国信息通信研究院征求意见，本标准由中国通信标准化协会提出并归口。

本标准起草单位：中国信息通信研究院，华为技术有限公司，深圳市腾讯计算机系统有限公司，腾讯云计算（北京）有限责任公司，中兴通讯股份有限公司，国际商业机器(中国)投资有限公司，中国移动通信集团有限公司

本标准主要起草人： 俊哲，郭雪，梁辰晔，高琨，陈哲，王勇，张亚军，刘海涛，苏强，项曙明，李响，李雪，邓小华

可信开源社区评估体系

1 范围

本标准针对单个项目的开源社区，从社区治理、社区运营、社区开发、基础设施这 4 个角度，规定了开源社区应关注的内容及指标。

本标准适用于针对开源项目分析社区内的发展情况，衡量社区的活跃度、贡献情况与健康度，帮助开源项目完善社区治理。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 11457　　信息技术 软件工程术语

3 术语和定义

3.1 开源 open source

即开放一类技术或一种产品的源代码，源数据，源资产， 可以是各行业的技术或产品，其范畴涵盖文化、产业、法律、技术等多个社会维度。

3.2 开源社区 open source community

以开源项目的贡献者为主体，在开源项目贡献过程中形成的具有特定文化、组织结构、运行机制的共同体。

3.3 可信 Trustworthy 开源社区

开源社区的可信即保证社区内治理架构和开源项目的可信，包括社区参与者的信息透明性、社区运营的包容性、社区开源项目的可用性、社区开源代码的安全性、社区工具平台的可靠性等。

3.4 开源软件 open source software

允许用户直接访问源代码，通过开源许可协议将其复制、修改、再发布的权利向公众开放的计算机软件。

3.5 开源组件 open source component

是开源软件系统中最小可识别且本身不再包含另外组件的、组件信息可在公共网站获取且可独立分发、开发过程中带有版本号并且可组装的软件实体。

3.6 开源许可证 open source license

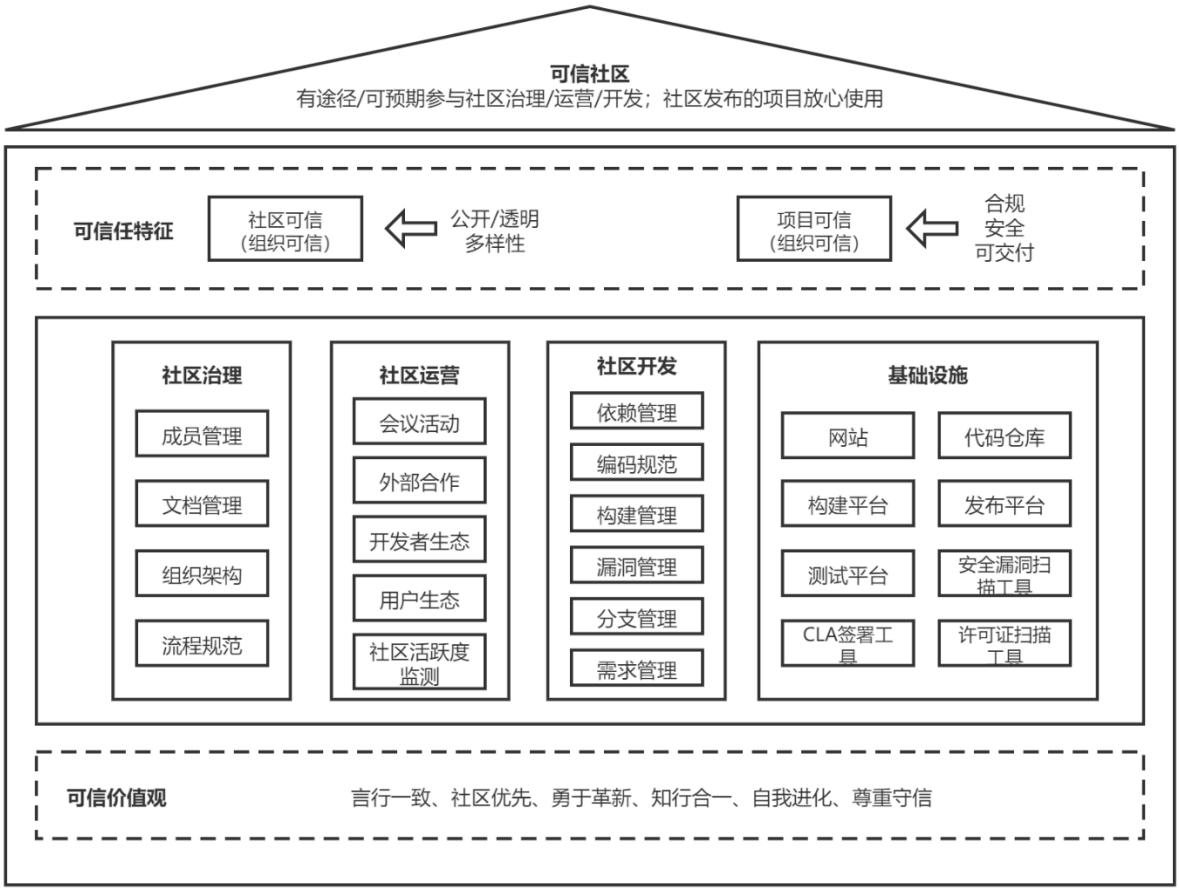
开源软件的版权持有人授予用户可以学习、修改开源软件，并向任何人或为任何目的分发开源软件的权利。

3.7 安全漏洞 vulnerability

计算机信息系统在需求、设计、实现、配置、运行等过程中，有意或无意产生的缺陷。这些缺陷以不同形式存在于计算机信息系统的各个层次和环节之中，一旦被恶意主体所利用，就会对计算机信息系统的安全造成损害，从而影响计算机信息系统的正常运行。

[GB/T 28458-2012，定义 3.10]

4 可信开源社区能力框架



5 社区治理

5.1 成员管理

5.1.1 行为准则（code of conduct）

社区应具有明确、包容的行为准则来规范社区维护者和参与者的行为，帮助不同背景的参与者融入社区，同时应提供违反准则者的约束措施，例如参与者公约1.4或2.0版本。

5.1.2 贡献者协议（CLA）

社区应具有明确的贡献者协议（CLA）来阐述开源项目本身所采用的开源协议，同时应对个人与企业签署进行划分，贡献者协议内容应包括：

a) 关于签署贡献者协议的主体信息；
b) 授予著作权给拥有该开源项目知识产权的组织或个人；
c) 授予专利权给该开源项目；
d) 签署者需保证依法有权授予上述许可；
e) 签署者需保证贡献内容属于原创作品；
f) 签署者在提交非原创作品时需采用的方式；
g) 保证在拥有不准确事实的情况下及时通知签约方。

5.1.3 贡献者类型

5.1.3.1 组织多样性

项目主要贡献者应来自不同国家或组织，社区组成多元化，对所有成员应一视同仁。

5.1.3.2 贡献多样性

社区内贡献者应具备多样的贡献方式来保证社区的健康度，除了提交代码外，还可以包括社区管理、安全漏洞提交、项目布道、用户支撑、市场运营等工作。

5.2 文档管理

社区内的重要文档应完备不遗漏，并且确保不同的用户都能理解使用。文档可用性包括结构、内容、专业名词的可读性等问题。

5.2.1 最终用户文档

社区应具有高可用性的最终用户文档来提高开源软件的可靠性与易用性。

5.2.2 开发者指南

社区应具有高可用的开发者指南来帮助贡献者参与项目，内容可包括：

a) 如何获取源代码；
b) 代码组织和目录结构；
c) 如何设置构建系统；
d) 如何进行构建和单元测试；
e) 开源项目的贡献规则。

5.2.3 变更日志

社区应具有详尽的变更日志来帮助贡献者与用户沟通版本差异。

5.3 组织结构

开源社区应具备明确的开源组织架构与分工，其职责可包括：

a) 项目管理：负责社区战略运营、规章制度、项目发展方向、下属组织设立、关键成员提名等工作；
b) 代码审查（Committer）：负责审核代码的质量和安全性、pull request评审、更新维护版本等工作；
c) 法律合规审查：负责知识产权审查、开源许可证的合规使用等工作；
d) 安全管理：负责接收、调查和披露社区相关的安全漏洞等工作。

5.4 流程规范

开源社区应具有清晰的社区流程规范，包括社区决策制度、投票机制、审批制度、反馈机制、辅导机制、发布机制等，社区内应通过文档共享等形式，确保社区成员明确相关制度的存在并依照制度规定执行。所有的关键性讨论都应该以书面的形式记录在主要的沟通渠道上，包括对社区与项目产生影响的线下讨论与私人讨论。

5.4.1 决策制度

社区应明确规定社区内各项工作的决策机制，并维护一个有决策权的贡献者公开列表。同时，决策人群应保持多样性，多元化的管理团队可以更容易同意与理解不同背景人群提出的问题与需求。

5.4.2 投票机制

社区应明确规定项目发展路径、关键成员选举的投票机制，并且投票结果应有明确的文档记录并公开透明。

5.4.3 审批制度

开源项目发布前应按照流程规定至少从开源许可证与法律合规、代码安全、代码质量等方面进行审核，审批通过并确认所使用的开源许可证后方可发布。

5.4.4 反馈机制

社区应建立公共沟通渠道保证社区公开透明，确保用户遇到问题时能够有途径获得社区的帮助；且对参与者提出的请求及时进行回复反馈，包括但不限于使用电子邮件、社区留言等形式。

5.4.5 辅导机制

社区应具有明确的辅导机制来帮助成员在社区内成长，从而确保整个社区的健康。通过辅导机制，社区可以邀请不同的贡献者，并创造包容的环境，让贡献者成长并回馈社区。

5.4.6 发布机制

社区应具有明确的发布机制来确保产品安全发布，内容可包括：
a) 代码发布应包含发行管理者的签名；
b) 代码发布时应具有开源许可证规定的License声明；
c) 版本发布是否严格遵循了投票流程与规则，例如充足的投票时间（大于72小时）；
d) 版本有清晰的RC，GA等类型区分；
e) 使用语义版本控制格式为每个发布提供唯一版本。

6 社区运营

6.1 会议活动

社区应定期举办线上线下的会议活动来推动内部/外部贡献者积极参与社区贡献。

6.1.1 活动分级

社区内活动应进行阶梯型分级，针对不同参与者的工作内容和技术等级，开展不同的会议活动；不同背景的演讲者有助于提供不同的观点和更广阔的视角，提升参与者的活跃度。

6.2 外部合作

社区应进行各类市场活动和项目布道，与不同的组织进行合作，可包括：

a) 国内外高校：社区与高校合作举办活动，进行人才培养；
b) 公司企业：社区与企业进行共建，产出企业案例；
c) 其他开源社区：社区与其他开源项目进行端对端合作构建生态，并共同举办社区活动。

6.3 开发者生态

6.3.1 培训认证

社区应具有建立匹配社区的开发者培训认证体系，增加社区贡献者的储备厚度，更加系统性的培养人才。

6.3.2 降低贡献者门槛

社区应通过降低贡献者门槛来吸引更多的贡献者参与，内容可包括：

a) 对Issue进行标准化处理，降低贡献者理解Issue的门槛；
b) 保持架构的稳定性，不轻易修改软件架构；
c) 构建良好的接口体系；
d) 保留足够的系统性能，让参与者来帮助提升；
e) 本地化运营：针对不同国家的人群进行定制化运营。

6.3.3 利益驱动

社区应通过利益驱动的方式鼓励参与者积极贡献，内容可包括：

a) 通过各类活动挑战赛获得奖金；
b) 通过各行业的用户案例来提升贡献者对项目的认同感与荣誉感；
c) 通过社区贡献获得工作机会；
d) 通过清晰的项目技术栈发展规划，让贡献者可以预期自身的技术成长。

6.4 用户生态

6.4.1 用户最佳实践

社区应在运营过程中寻找潜在行业用户，通过在生产环境下获得反馈，优势场景研究和输出，最终产出用户最佳实践并进行广泛传播。

6.4.2 产品易用性

社区应持续提升产品易用性，例如给用户提供好用的操作流程与API，产品可以持续稳定升级等。

6.4.3 降低使用门槛

社区应通过降低使用门槛来吸引更多的使用者，内容可包括：

a) 社区网站的建设与可用性；
b) 项目文档的完整性与正确性；
c) 及时的社区沟通反馈与支持；
d) 商业运营：针对衍生和商业化的产品服务进行一致性、兼容性与安全性产品认证。

6.5 社区活跃度监测

开源社区应持续监测社区的活跃健康度，指标可包括：

a) 项目状态：Pull request数、Commit数、Issue提交数与评论数、Star与Fork数等；
b) 贡献者趋势：贡献者活跃时间、特定时间内贡献者的增加或减少量，特定时间段内有多少贡献者停止贡献的指标等；
c) 用户数量；
d) 社区文章阅读量；
e) 用户案例趋势；
f) 企业/组织劳动力投入；
g) 活动分析：社区内会议活动的参与度。

7 社区开发

7.1 依赖管理

7.1.1 开源软件选型

社区内基于需求和漏洞修复引入的上游社区开源软件都应满足技术生态、合法合规（如出口管制）、网络安全和生命周期要求，供对外开源项目配套使用。

7.1.2 依赖软件使用（可选）

社区内依赖软件的使用应满足：

a) 将开源项目自研代码与上游开源代码隔离，开源软件独立目录（repo）存放。
b) 确保被集成的所有开源软件及其依赖软件满足开源许可证的合规要求（许可证兼容性判断等），且正确履行开源使用声明义务。

7.1.3 开源软件维护（漏洞修复）

社区应对标上游开源社区，按照以下方式及时修复开源漏洞：

a) 升级维护模式：同步上游社区，及时获取新版本解决当前维护版本存在的安全漏洞；
b) 补丁维护模式：从上游社区获得漏洞补丁，修复当前维护版本的安全漏洞。

7.2 编码规范

7.2.1 合入管理

7.2.1.1 权限机制

社区内的代码合入应遵循：

a) 所有合入主干的代码，必须通过代码检视和代码审查者（Committer）审核；
b) 代码提交权限与代码合入权限分离，避免恶意代码植入，提高代码质量；
c) 建立peer review机制，以保障代码审查（Committer）机制的健康高效运作。

7.2.1.2 代码合入门禁机制（可选）

社区内的代码合入门禁应包括：
a) 门禁检查内容、阈值和规则的建立；
b) 门禁需规范化运营。

7.2.2 编码管理

7.2.2.1 IDE 级代码检查

IDE应内嵌代码检查插件，方便开发人员及时检查。

7.2.2.2 开发者测试

社区内应开展开发者测试，内容可包括：
a) 开展开发者测试设计，并对设计输出结果进行评审；
b) 开发者测试代码应参照业务代码进行管理；
c) 开发者测试工程上应设置门禁；
d) 持续提升开发者测试代码质量与效果；
e) 开展模糊（FUZZ）测试。

7.2.2.3 代码重构

社区内代码重构应满足：
a) 明确的重构策略与计划；
b) 以社区需求为基础进行重构；
c) 社区内重构能力与工具应满足需求；
d) 社区内重构能力可持续积累与传递。

7.3 构建管理

7.3.1 构建环境

社区内保证构建环境的可信，包括：
a) 社区内构建环境变更应无手动操作，统一通过修改环境代码进行变更。
b) 社区应使用构建镜像自动生成，具备环境自动化搭建能力，操作过程无人工干预。
c) 社区内构建环境应代码化，并纳入配置管理。

7.3.2 构建执行过程

社区内保证构建执行过程的可信，包括：
a) 社区内构建工程框架本身代码化使用主流高阶构建框架（如Cmake,go module,Maven,Gradle,Ninja,OBS,Yocto等），并纳入社区的代码仓管理；
b) 构建工程CI调度使用代码化描述；
c) 构建任务启动后，构建过程无人工干预。

7.3.3 构建结果

社区内构建结果应在相同源码与相同环境的条件下，应进行多次比较，且结果一致。

7.3.4 构建数据源

社区应保证构建数据源可信，包括：

a) 基础来源可信：使用业界主流版本的基础镜像，源头可追溯到社区版本或官方版本,且遵循社区开源软件的管理要求，与其保持一致；
b) 构建工具来源可信：使用业界主流版本构建工具，且遵循社区开源软件的管理要求，与其保持一致；
c) 开源软件来源可信：使用业界主流版本的开源软件，源头可追溯到社区版本或官方版本,且遵循社区开源软件的管理要求，与其保持一致。

7.4 漏洞管理

7.4.1 漏洞修复方案管理

7.4.1.1 CVE 申请（可选）

社区应遵循开源社区CVE申请策略，对符合条件的漏洞申请CVE编号。

7.4.1.2 漏洞公告编写与评审

社区应根据漏洞影响分析报告，组织编写漏洞公告，并评审。

7.4.1.3 漏洞修复方案

社区内漏洞修复方案应具备：

a) 社区具备版本生命周期内通过补丁修补漏洞的能力；
b) 确保修复方案正确修复漏洞；
c) 制定版本发布漏洞修复期限，确保漏洞修复进度符合版本发布周期；
d) 社区制定策略，决策哪些漏洞可以通过版本升级方式修复，通常是影响性较低的漏洞。

7.4.1.4 漏洞修补验证

社区应通过黑盒、白盒、软件包成分扫描分析验证版本漏洞修复情况。

7.4.1.5 漏洞与补丁发布

社区应有明确的漏洞与补丁发布机制，包括：

a) 确认需要受限披露的漏洞，都提前通知到上下游利益相关方（下游相关方的定义和选择由社区决策，并不是所有都需要通知），且有保护机制，保证利益相关方不会提前披露漏洞信息；
b) 补丁发布后预留时间给上下游修复，再发布漏洞公告，避免因漏洞公告发布导致漏洞被利用。

7.4.2 漏洞感知

社区内漏洞感知应包括：

a) 上游社区漏洞感知：确保社区使用的所有上游社区软件漏洞披露渠道都得到监控，且漏洞获取及时，无遗漏；
b) 内部上报漏洞感知：为社区内部提供合适的漏洞上报途径；

c） 外部上报漏洞感知： 为外部研究者提供合适的漏洞上报途径， 确保外部研究者上报的漏洞及时响应.

7.4.3 漏洞可追溯

社区内漏洞可追溯应包括：

a） 上游社区软件可追溯：社区软件版本和所依赖的上游社区软件版本双向可追溯；

b） 漏洞影响版本全量可视：构建漏洞影响视图，该视图包含所有受影响的软件及其版本信息；

c） 版本涉及漏洞全量可视：构建版本漏洞视图，该视图包含影响软件对应版本的所有漏洞。

7.4.4 漏洞验证

社区内漏洞验证应包括：

a） 漏洞影响分析：基于漏洞严重等级评估标准，对漏洞进行评定，且全面排查受影响版本，确保无遗漏；

b） 漏洞有效性分析：通过漏洞报告分析漏洞有效性。

7.5 分支版本与工作流管理

社区内应具备完善的分支版本与工作流管理，内容可包括：

a） 社区的开发需要根据自己的项目规模和项目维护的策略选择自己项目的工作流；

b） 下游的开发可能在局部采取不同于上游的工作流；

c） bug修复遵循上游优先的原则。先在上游修复；

d） 分支应及时合入主干，以减少合并风险；

e） 合入到主干分支之前都要进行充分的测试；

f） 维护分支上的开发理论上都属于修复和backport，不应该有高版本没有的特性。

7.6 需求管理

社区内需求管理应具有端对端可追溯的能力，在分解分配、开发、测试验证过程的数据对象和关系都应被记录，可支撑相关过程记录与现场还原。

8 基础设施

开源社区应使用自动化工具平台来协助相关人员进行规范的开源管理，工具应具有明确的权限管理与安全机制。

8.1 网站

开源产品应具有一个稳定的网站，本标准中所提到的所有信息都应该清晰展示，可包括产品使用场景、项目源代码获取方式、邮件列表、反馈方式、贡献方式等。

8.2 代码仓库（可选）

社区对在开发中需要修改的开源代码应建立源码仓库存储，同时设置开源软件白名单，对需要引入的开源代码进行开源许可证、开源漏洞等方面审查，通过后方可使用。

8.3 构建平台

社区应对各分支代码分别进行管理，保障各分支均明确负责人员，对每个分支进行同样的管理流程和测试步骤，标记并记录每个分支与主版本差异部分，同时构建平台应

a) 启用（并修复）编译器警告和类 lint 的检查；

b) 执行静态分析工具并发现修复可被攻击利用的问题漏洞。

8.4 发布平台

社区应对软件交付物进行管理，对社区提供给用户的交付物进行开源软件识别和评估，保障交付物符合用户需求和开源合规要求。

8.5 许可证扫描工具（可选）

社区应运用工具识别代码中的开源组件及开源许可证，确保开源许可证合规使用，如有问题应及时作出相应处理。

8.6 安全漏洞扫描工具（可选）

社区应运用安全漏洞扫描工具发现代码漏洞等安全问题，并及时作出相应处理。

8.7 贡献者许可协议（CLA）签署工具

社区应运用贡献者许可协议（CLA）签署工具，外部代码贡献者如参与项目贡献应首先签署贡献者许可协议（CLA）；CLA签署工具应与开源代码托管平台的账号进行绑定，以此来识别贡献者的个人信息。

8.7.1 多语言支持（可选）

CLA签署工具可支持多种语言签署，来帮助不同国家的贡献者签署贡献者协议。

8.7.2 CLA 签署机器人（可选）

当贡献者在代码托管平台提交PR时，CLA签署机器人可检查作者是否已经签署CLA，如果已经签署则打上已签署标签，否则提供CLA签署地址。

8.8 测试平台

社区内测试平台应进行开发者测试，拥有覆盖大部分代码/功能的自动化测试套件，并自动拉取新代码运行测试，应用动态检查，包括但不限于执行内存/行为分析，模糊测试，web安全扫描等。

8.8.1 组合测试（可选）

测试平台可对代码进行硬件/操作系统/软件的组合测试。

8.8.2 Bug 自动定位（可选）

测试平台可拥有自动定位bug责任人的功能，平台自动找出引发故障的commit，并通过邮件发送bug报告给该commit的贡献者。

8.8.3 自主验证（可选）

测试平台可给开发者提供调试验证的环境，进行相关场景复现。