Project Incubation Process Workflow:

Prepared by:
Christian Taylor
Head of Open Source Office, Intersect
&
Terence “Tex” McCutcheon
Open Source Program Manager, Intersect

Date:
February 5, 2025

Organization:
Open Source Committee
Intersect Member Based Organization
Cardano Ecosystem

Abstract:

The incubation life cycle described below is written by Intersect's Open Source Office, is endorsed by Intersect's Technical Steering Committee and is implemented by Intersect's Open Source Committee, these groups henceforth will be identified as OSO, TSC, and OSC respectively.

This framework emphasizes sustainable growth, community engagement, and adherence to governance practices. By establishing clear milestones for early project development, technical infrastructure, and community formation, the Incubation Framework provides essential resources and guidance to help projects attract contributors, build solid foundations, and align with Cardano's long-term open-source goals

Projects looking for support should fall within the **Incubation** phase and have a trajectory to move through each phase with the exception of **Archive** unless the *Project* fails to succeed. After successful completion of the metrics set forth in the **Incubation** phase, *Projects* will move to **Growth**, followed by **Maturity and Maintenance**. If a *Project* experiences or begins to display points of failure in either the **Growth** or **Maturity** phases it will proceed to the **Decline** phase where there is an opportunity to return to the **Growth** stage upon achieving the requirements metrics and reporting. Failure to adapt the project within the guidelines of the **Decline** phase, a *Project* then transitions to **Archive** and prepares to public archive within an established timeframe.

Executive Summary

The Project Incubation Framework, from Intersect's Open Source Office, serves as a guide for new and emerging projects within Cardano's open-source ecosystem, supporting their journey from initial concept to a sustainable, community-backed development. Recognizing the unique needs of early-stage open-source projects, the Incubation Framework emphasizes building a robust infrastructure, developing community engagement channels, and establishing governance alignment with Intersect's Open Source Committee (OSC) standards.

The Incubation phase is critical for establishing the core elements of a project's success. This framework provides a structured pathway with key areas of focus:

1. **Stage Definition**: The Incubation phase acts as a sandbox environment where new projects set up essential infrastructure, outline their goals, and test their ideas. By providing a clear roadmap and an early technical setup, projects are equipped to establish a stable base for future development.
2. **Criteria for Entry**: Projects eligible for Incubation are typically newly created or recently forked, with a limited codebase, minimal documentation, and a small contributor base. Projects must show alignment with Cardano's open-source governance standards and a clear vision and roadmap for development that adds to or improves features/products in the ecosystem.
3. **Goals**: Incubation focuses on building a solid technical foundation, engaging the community, and creating comprehensive documentation. Projects are guided to establish essential infrastructure, define a clear vision, set up processes for feedback collection, and foster both growth and engagement.
4. **Actions**:
    - **Project Vision and Roadmap**: Host stakeholder workshops and perform a SWOT analysis to establish a strategic vision, identify potential challenges, and set achievable milestones.
    - **Community Formation**: Develop early community engagement via various channels (e.g., Discord, GitHub discussions) and host introductory events to build an initial user and contributor base.
    - **Technical Foundation**: Establish a repository that includes the core code and connections to Core-Cardano thereto, with version control, CI/CD pipelines, and basic Security practices. Documentation, such as a README and contributor guidelines, are written to make the project accessible to new contributors. Details: OSC Governance Policy.
    - **Initial Marketing and Outreach**: Launch initial marketing efforts to raise awareness, including blog posts, videos, and participation in relevant

open-source communities and tech events.

5. **Monitoring and Evaluation**: Throughout Incubation, metrics are tracked to gauge progress, community engagement, and technical stability. Monthly reviews provide a feedback loop to refine strategies based on user and contributor input, ensuring the project stays aligned with its objectives. Key metrics include active contributor count, code commit frequency, community engagement levels, and roadmap milestone completion.

The Project Incubation Framework fosters the growth of sustainable open-source projects that align with Cardano's mission of community empowerment and technical excellence. Through strategic planning, resource allocation, and structured monitoring, the Incubation phase provides a robust foundation that enables projects to transition smoothly to the next stages of growth, ultimately enhancing Cardano's ecosystem and supporting the broader open-source movement.

Incubation Phase

- **Definition**
  - **Purpose**: The Incubation phase serves as the foundation-building stage where projects: establish their essential structure, define their purpose, and begin building the core infrastructure needed for future stages. The primary objective is to establish a solid base for community engagement and technical development. This process acts as a sandbox to test new ideas, form a foundation, and coalesce a following from the community behind them.
  - **Key Characteristics**:
    - Early-stage or minimal codebase with the initial functionality in place.
    - Limited documentation focused on the basics, such as a README, initial setup instructions, and license information.
    - Small contributor base, typically between 1–3 individuals.
    - Minimal to no formalized community, but some engagement or discussion channels may be introduced.
  - **Requirements for Entry**:
    - Project must be newly created or recently forked, showing alignment with
      - Intersect's Open Source Strategy
      - Intersect's Open Source Committee (OSC)
      - Intersect's Technical Steering Committee (TSC)
      - Intersect's Open Source Office (OSO)
    - Minimal codebase is available, with some foundational elements developed.
      - Less than 5 active builds
      - Adheres to Intersect OSC open source Governance
    - Initial concept is formed, ready to be structured for growth and community engagement.
      - Clear baseline Technical Documentation
      - Roadmap for intended Development
      - Value Proposition(s) to the Cardano Community

- **Criteria**
  - **Community Readiness**:
    - Identification of a target user and contributor base, with early consideration of community management and engagement strategies.
    - Plans to create basic communication channels, such as a Discord server, Matrix channel, or to utilize GitHub discussions.
  - **Technical Foundation**:
    - Foundational technical setup, including repository structure, initial CI/CD configurations, and version control established in line with open-source best practices.
    - Basic functional code or proof of concept to demonstrate the project's core value or potential.
  - **Documentation**:
    - Initial documentation that clearly describes the project's purpose, installation instructions, and code of conduct, following in adherence with Intersect's Governance Policy.
    - Contributing guidelines to make it easy for early contributors to understand the code structure and expectations.
  - **Commitment to Roadmap**:
    - High-level roadmap drafted to outline milestones for short-term and long-term objectives.
    - Defined project vision statement to provide clarity on the project's direction and purpose.

- **Goals**
  - **Community Development**:
    - Engage an initial base of contributors and users by establishing basic community guidelines and creating introductory channels.
    - Engage within the Intersect Discord and or working groups to stimulate community engagement to the project.
    - Increase visibility within the Cardano ecosystem and begin to build relationships with other open-source contributors.
  - **Technical Setup**:
    - Set up foundational elements of the codebase to align with OSC Governance including repository structure, CI/CD pipelines, and version control configurations.
    - Establish initial security practices and documentation templates to encourage consistency and scalability.

- The project will be hosted/migrated under the Intersect Github and a team structure for access will be provided in adherence with the OSC Governance policy working with the project maintainers.

➢ **Project Roadmap and Vision**:
- Create a well-defined vision statement and develop a roadmap with achievable milestones and projected timelines.
- Ensure clarity around project goals to foster alignment among early contributors.

➢ **Feedback Mechanisms**:
- Develop mechanisms to gather feedback from contributors and users, such as monthly community calls, feedback forms, and dedicated forums for suggestions.
- Schedule periodic check-ins with initial contributors to gather input on the development process and areas for improvement.

❖ **Actions**

➢ **Project Vision and Roadmap**:
- Host stakeholder workshops to align on the project's vision, values, and objectives.
- Conduct a SWOT analysis to identify the project's strengths, weaknesses, opportunities, and threats, refining the roadmap as necessary.

➢ **Community Formation**:
- Create and set up community engagement channels, such as Discord, GitHub discussions, or mailing lists.
- Organize introductory webinars, Q&A sessions, or workshops to help early contributors understand the project and ways to get involved.

➢ **Technical Foundation**:
- Set up an accessible and well-organized code repository with initial CI/CD tools, version control, and basic security framework.
- Develop initial documentation, including a README file, a "Getting Started" guide, and templates for reporting issues and submitting contributions.

➢ **Initial Marketing and Outreach**:
- Implement basic marketing strategies to attract contributors, such as creating blog posts, introductory videos, and social media content.
- Actively participate in relevant open-source communities and tech

events to attract early interest and support for the project.

- **Monitoring and Evaluation**
  - **Success Metrics**:
    - Track the number of active contributors and frequency of code commits to gauge development progress, provided by Intersect Open Source Office.
    - Measure levels of community engagement, such as participation in Q&A sessions or number of interactions in discussion channels.
    - Track the completion of roadmap milestones to ensure that the project is moving forward as planned.
  - **Regular Reviews**:
    - Schedule monthly reviews to evaluate progress and make adjustments as needed based on feedback from contributors and users.
    - Use insights from these reviews to refine the roadmap and action items, adapting to the needs of the growing community.
  - **Specific Criteria Reviewed:**
    - Contributor Count and Retention
      - Metric: Number of consistent contributors (e.g., those contributing monthly over a 3-6 month period).
      - Purpose: Shows that the project is attracting and retaining contributors, a key sign of community engagement and sustainability.
    - Basic Testing Coverage
      - Metric: Percentage of code covered by basic tests (e.g., unit tests) with a goal of at least 50% coverage for key functionalities.
      - Purpose: Establishes that the project has essential tests to ensure functionality, which is critical as it scales.
    - Issue Response Time
      - Metric: Average time to respond to new issues, ideally within 48 hours.
      - Purpose: Indicates a responsive and active project, reassuring new contributors and users that their feedback will be heard and addressed.
    - Completion of Core Features
      - Metric: Percentage of initial core features implemented (e.g., 80% or higher completion of planned MVP features).

        - Purpose: Confirms that the project has a stable, usable foundation, making it appealing for further adoption and growth.
    - Community Engagement Score
        - Metric: Simple metric based on interactions like comments on issues, PRs, or basic feedback surveys.
        - Purpose: Reflects community interest and satisfaction, showing that the project is meeting user needs and has momentum.
- ➢ Transition to Growth Phase

Growth Phase

- **Definition**
  - **Purpose**: The Growth phase is focused on scaling and expanding the project's impact, both technically and within the community. This stage is designed to stabilize infrastructure, broaden the contributor base, and deepen community engagement. The objective is to enhance project quality and establish robust processes that can sustain increased demand and interest.
  - **Key Characteristics**:
    - Increase in the number of contributors and community members, signaling momentum.
    - Expanded codebase with more features and greater functionality.
    - Comprehensive and structured documentation that supports scaling and usability.
    - Formalized community channels with active engagement and participation.
  - **Requirements for Entry**:
    - Established core technical infrastructure from the Incubation phase, ready to support greater complexity and scaling.
    - Sufficient contributor base to enable continuous development, ideally more than five consistent contributors.
    - Proven increase of community interest and/or user base growth.
    - Adherence to Intersect's governance practices through the OSC, TSC, and OSO.
    - Comprehensive roadmap showing the detailed trajectory for the next development phases.

- **Criteria**
  - **Community and Contributor Engagement**:
    - Evidence of increasing contributors and active community members, with an emphasis on fostering diverse and skilled contributions.
    - Plans to implement governance practices that ensure structured, inclusive decision-making.
  - **Technical Maturity**:

- Codebase expansion with additional features, improved stability, and increased functionality.
- Adoption of more comprehensive testing and quality assurance (QA) practices, such as unit, integration, and automated tests.

➢ **Documentation**:
- Thorough documentation covering all primary features, setup, contribution guidelines, and troubleshooting.
- Expanded user documentation and guidelines to facilitate ease of adoption by new contributors.

➢ **Project Governance and Roadmap Commitment**:
- Clearly defined project roadmap with updates reflecting growth milestones.
- Governance structures that enable efficient, transparent decision-making aligned with Intersect's governance policies.

❖ Goals

➢ **Scalability and Stability**:
- Ensure infrastructure can handle increased usage and complexity, with an emphasis on stability, uptime, and performance optimization.
- Adopt processes that maintain a high standard of code quality and reliability.

➢ **Community Development**:
- Continue engaging contributors, expanding diversity within the community, and supporting a positive, inclusive environment.
- Develop recognition programs and community management tools to incentivize contributions and strengthen community ties.

➢ **Enhanced Documentation and Support**:
- Maintain and expand documentation to support new features and anticipated user needs.
- Implement improved support channels and response times for user inquiries.

➢ **Governance and Structured Decision-Making**:
- Establish governance practices that include community voting, regular meetings, and clear documentation of decisions and rationales.
- Facilitate smooth project operations and empower contributors to actively participate in decision-making.

❖ **Actions**

- ➢ **Scaling Infrastructure**:
  - Enhance the technical infrastructure to handle greater loads, improve CI/CD processes, and ensure robust security frameworks.
  - Implement monitoring tools to track performance and usage metrics, allowing proactive adjustments as demand scales.
- ➢ **Community Engagement and Recognition**:
  - Develop community recognition programs (e.g., badges, contributor of the month awards) to motivate contributors.
  - Organize regular community events, such as contributor meet-ups, hackathons, and workshops, to strengthen engagement.
- ➢ **Advanced Documentation and Support**:
  - Expand documentation to include new features, advanced usage guides, and troubleshooting FAQs.
  - Set up efficient support channels and track response times, aiming to respond to community inquiries within a standard time frame.
- ➢ **Governance Implementation**:
  - Establish clear governance protocols that include transparent decision-making processes, voting systems, and regular governance meetings.
  - Document all decisions in alignment with Intersect's OSC governance policies to ensure transparency and accountability.
  -

❖ **Monitoring and Evaluation**

- ➢ **Metrics**:
  - **Contributor Growth**: Track the number of active contributors and new contributions to assess engagement and growth.
  - **Code Quality and Coverage**: Track code quality metrics such as test coverage, with a target for at least 70% coverage of critical functions.
  - **Community Activity**: Measure interactions in forums, issue comments, PRs, and overall community engagement scores.
  - **Uptime and Performance**: Monitor system performance, uptime, and response times to ensure stability and scalability.
- ➢ **Regular Reviews**:
  - Conduct quarterly reviews to evaluate the effectiveness of scaling efforts and community engagement strategies.
  - Adjust the roadmap as needed based on metrics and feedback,

ensuring alignment with project goals and community needs.

➢ Specific Criteria Reviewed:

- **Contributor Engagement and Growth Rate**
  - **Metric**: Number of active contributors and the rate of new contributors joining each month.
  - **Reason**: Strong contributor engagement signals that the project has garnered community interest and can sustain development through a broad and active contributor base, which is essential for scaling.
- **Code Quality and Testing Coverage**
  - **Metric**: Percentage of code covered by unit and integration tests, along with code quality indicators (e.g., low technical debt).
  - **Reason**: Ensuring that the codebase is robust and thoroughly tested establishes confidence in the project's stability and scalability as it moves to the growth phase.
- **Response and Resolution Time for Issues and Pull Requests**
  - **Metric**: Average time to respond to new issues and resolve pull requests.
  - **Reason**: Quick responsiveness indicates that the core team and community are committed and capable of maintaining momentum. Projects with efficient support and issue resolution are better positioned to attract and retain users and contributors.
- **Feature Completeness and Roadmap Alignment**
  - **Metric**: Percentage of roadmap goals and key feature milestones achieved during the incubation period.
  - **Reason**: A project that has successfully met initial roadmap goals demonstrates clear direction and viability, which are critical to proving readiness for further growth and external adoption.
- **Community Feedback and Satisfaction**
  - **Metric**: Qualitative and quantitative community feedback on project usability, relevance, and satisfaction (e.g., through surveys or GitHub feedback).
  - **Reason**: Positive community feedback signals that the project is meeting user needs and has practical relevance, which is essential for a smooth transition to broader adoption in the growth phase.

Maturity and Maintenance Phase

- **Definition**
  - **Purpose**:
    - **Maturity**: At this stage, the project reaches a stable, well-established state with a mature codebase, a reliable release schedule, and a broad user base. The focus is on sustaining quality, addressing minor improvements, and enhancing user experience. Maturity is about reinforcing the project as a dependable solution within its domain.
    - **Maintenance**: Following Maturity, the project enters Maintenance, where the primary goals are continued support, minor updates, and bug fixes rather than new feature development. Maintenance ensures the project's ongoing usability, reliability, and compatibility within the ecosystem without active expansion.
  - **Key Characteristics**:
    - **Maturity**:
      - Large, active community with diverse contributors and strong industry recognition.
      - Stable codebase with regular, predictable release cycles.
      - Highly refined documentation and robust testing framework.
    - **Maintenance**:
      - Continued user support and bug fixes, with minimal feature additions.
      - Ongoing community engagement to ensure user satisfaction and retention.
      - Long-term viability planning and succession plans for leadership roles.
  - **Requirements for Entry**:
    - Achieved stability in codebase and governance, with processes in place for conflict resolution and contributor succession.
    - High community adoption and a strong reputation within the industry.
    - Clear documentation and technical infrastructure that support both user onboarding and sustained contributions.

- **Criteria**
  - **Community Engagement and Cohesion**:

- **Maturity**: Active, engaged community with programs to recognize contributors, support diversity, and encourage collaboration.
- **Maintenance**: Retain community interest through regular check-ins, user support channels, and recognition of continued contributions.

➢ **Technical Stability and Quality**:
- **Maturity**: Feature-complete and stable codebase with minimal need for major additions or rework.
- **Maintenance**: Infrastructure geared toward longevity, with a focus on bug fixes, updates for compatibility, and small refinements.

➢ **Documentation and Support**:
- **Maturity**: Comprehensive documentation, including advanced guides, troubleshooting, and contribution guidelines.
- **Maintenance**: Keep documentation current, with revisions as necessary to reflect updates and maintain relevance for users.

➢ **Governance and Succession Planning**:
- **Maturity**: Solid governance practices, including conflict resolution and community-based decision-making.
- **Maintenance**: Succession planning for key roles, including leadership and technical positions, to ensure continuity and stability.

❖ **Goals**

➢ **Sustainability and Quality Control**:
- **Maturity**: Maintain code quality and project stability, refining processes to ensure reliability and minimize regressions.
- **Maintenance**: Ensure long-term functionality through consistent bug fixes, minor updates, and routine maintenance tasks.

➢ **Community and Contributor Retention**:
- **Maturity**: Implement contributor recognition programs, annual events, and learning opportunities to keep the community engaged and motivated.
- **Maintenance**: Retain core contributors by continuing engagement programs and fostering community loyalty through targeted interactions.

➢ **Conflict Resolution and Succession**:
- **Maturity**: Develop and maintain conflict resolution processes, ensuring a cohesive community experience.
- **Maintenance**: Implement succession planning, focusing on mentoring and shadowing to prepare future leaders.

- **Ongoing Improvement and Adaptation**:
  - **Maturity**: Continuously improve project features, integrating feedback to meet evolving needs while maintaining a stable core.
  - **Maintenance**: Adapt to necessary changes that may arise from dependency updates, community needs, or new platform requirements.

- **Actions**
  - **Continuous Improvement**:
    - **Maturity**: Regularly review and integrate feedback from users and contributors, ensuring the project remains competitive and relevant.
    - **Maintenance**: Implement minor updates and ensure compatibility with dependencies or ecosystem changes.
  - **Community Engagement and Support**:
    - **Maturity**: Host annual community events, provide awards or badges for outstanding contributors, and promote ongoing training and development.
    - **Maintenance**: Keep support channels open, address community inquiries promptly, and hold periodic community reviews.
  - **Documentation and Knowledge Management**:
    - **Maturity**: Expand documentation to cover advanced use cases, implementation examples, and best practices.
    - **Maintenance**: Regularly update documentation to reflect minor updates, ensuring it remains current and accurate.
  - **Succession Planning and Knowledge Transfer**:
    - **Maturity**: Identify key contributors and begin mentoring programs to ensure knowledge transfer and leadership continuity.
    - **Maintenance**: Develop succession plans for critical roles, document key processes, and conduct regular knowledge-sharing sessions.

- **Monitoring and Evaluation**
  - **Metrics**:
    - **Contributor Retention**: Track the number of active contributors over time, aiming for stable or increasing contributor retention.
    - **Community Satisfaction**: Measure user satisfaction through surveys and feedback, aiming to maintain high engagement and

satisfaction levels.
    - **Stability and Uptime**: Monitor system stability, aiming for minimal downtime and consistent performance.
    - **Documentation Quality and Completeness**: Assess documentation relevance and completeness, adjusting as needed for both current and new users.
- **Regular Reviews**:
    - **Maturity**: Conduct bi-annual strategic reviews to evaluate the project's alignment with long-term goals, stability, and community needs.
    - **Maintenance**: Perform annual reviews focused on usability, compatibility, and retention, ensuring that the project remains relevant and accessible
- **Specific Criteria Reviewed:**
- **Mature Projects**
    - **Broad and Sustained Contributor Network**
        - **Metric**: High number of contributors sustained over time, with diverse roles (e.g., core maintainers, casual contributors, technical writers).
        - **Purpose**: A mature project should demonstrate a broad, decentralized contributor base, indicating that the project can sustain itself and evolve without dependency on a small core group.
    - **Extensive Testing and Continuous Integration (CI) Robustness**
        - **Metric**: 90%+ test coverage across unit, integration, system, and security tests, with fully automated CI/CD pipelines that trigger on each change.
        - **Purpose**: High testing and CI/CD adoption indicates the project's resilience, enabling it to handle large-scale deployments and ensuring reliability as it continues to grow.
    - **Adoption and Usage in Real-World Applications**
        - **Metric**: Number of organizations and key stakeholders using the project in production environments, with case studies or testimonials demonstrating value.
        - **Purpose**: Mature projects should have strong external adoption, validated by organizations relying on them in real-world applications. This signals relevance and credibility in the ecosystem.
    - **Sustainable Funding or Sponsorship Base**
        - **Metric**: Stable financial support from grants, donations, or

sponsorships, covering essential maintenance costs and possibly growth initiatives.
- **Purpose**: Financial sustainability is crucial for mature projects, as it ensures they can support continued development, cover infrastructure costs, and maintain contributors' engagement over the long term.

- **Documented Governance and Decision-Making Processes**
  - **Metric**: Clear, documented governance model (e.g., voting systems, elected maintainers, transparent issue prioritization) with regular community participation in decision-making.
  - **Purpose**: A mature project should have a structured governance model that allows for fair and efficient decision-making, giving stakeholders confidence in the project's longevity and integrity.

➢ **Maintenance Projects:**

- **Issue Response and Resolution Time**
  - **Metric**: Median response time for issues within 24-48 hours, with resolution for critical bugs typically within 1-2 weeks.
  - **Purpose**: Even in maintenance mode, timely responses to user-reported issues indicate the project remains dependable and well-supported.
- **Frequency of Security Updates and Patches**
  - **Metric**: Time taken to address and patch security vulnerabilities, ideally within days for high-severity issues.
  - **Purpose**: A mature project in maintenance mode must prioritize security to protect users and maintain trust, even without new feature development.
- **Codebase and Dependency Health**
  - **Metric**: Regular updates to core dependencies (e.g., third-party libraries, frameworks) and a low percentage of outdated or vulnerable dependencies.
  - **Purpose**: Ensuring dependencies are up-to-date and secure minimizes potential vulnerabilities and compatibility issues over time.
- **Community Support and Engagement Level**
  - **Metric**: Consistent community interaction, such as answering support questions or maintaining documentation, with satisfaction ratings from users where possible.
  - **Purpose**: Maintaining a healthy level of engagement through

community support forums, documentation updates, and occasional discussions helps users feel supported without expecting new features.

- **Documentation Completeness and Clarity**
  - **Metric**: Documentation is fully up-to-date, with clear and accessible instructions for common issues, troubleshooting, and integration.
  - **Purpose**: Comprehensive documentation is crucial in maintenance mode, as it reduces reliance on developers for support, empowering users to resolve common issues independently.

Decline Phase

- **Definition**
  - **Purpose**: The Decline phase identifies and responds to signs of decreasing activity, engagement, or relevance in a project. The focus is on understanding the causes of decline, exploring possible revitalization avenues, and planning for the project's future—whether that means reinvigorating it or preparing for an organized closure.
  - **Key Characteristics**:
    - Noticeable reduction in contributor activity, community engagement, or user interest.
    - Slowdown in updates, new features, or critical improvements.
    - Challenges in maintaining the project's technical relevance or aligning with ecosystem needs.

- **Assessment and Diagnosis**
  - **Goal**: Conduct a comprehensive review to diagnose the root causes of decline. This step helps determine whether the project can be revitalized or if it's heading toward closure.
  - **Key Actions**:
    - **Stakeholder Feedback**: Gather input from contributors, users, and community members to understand perceptions, needs, and challenges.
    - **Competitive Analysis**: Evaluate whether alternative projects or technologies have overtaken the project in value or functionality.
    - **Internal Review**: Assess internal project aspects, such as technical debt, governance issues, or lack of resources, that may be driving decline.

- **Decision Pathways: Revitalize or Prepare for Closure**
  - **Revitalization Pathway**:
    - If the project has potential for rejuvenation, take specific steps to attract new interest, pivot as necessary, and stabilize community involvement.
      - **Revitalization Actions**:
        - **Rebranding and Marketing**: Introduce new branding, marketing campaigns, or outreach efforts to attract

renewed interest.
- **Feature Realignment**: Modify or refocus core features based on current user demands or emerging trends within the ecosystem.
- **Increased Community Engagement**: Launch community re-engagement strategies, such as hackathons, contributor recognition programs, or targeted workshops.
- **Project Pivot**: If necessary, shift the project's focus or application to better meet the needs of a new audience or niche within the ecosystem.

- **Closure Pathway**:
  - If revitalization is deemed unfeasible, plan for an organized and respectful project sunset, focusing on legacy preservation and smooth user transition.
    - **Closure Actions**:
      - **Communication of Closure Plans**: Transparently communicate the closure decision to all stakeholders, explaining reasons and timelines.
      - **Transition Support**: Provide users and contributors with guidance on alternative projects or tools they might transition to, or encourage the community to fork the project if they wish to continue it independently.
      - **Legacy Preservation**: Archive project assets (code, documentation, community contributions) in accessible repositories to ensure ongoing availability and value to the ecosystem.
      - **Celebrate Contributions**: Recognize and celebrate the contributions of the community, highlighting key achievements and milestones to honor the project's legacy.

- **Monitoring and Evaluation During Decline**
  - **Goal**: Maintain ongoing evaluation to guide decision-making throughout the Decline phase. Whether revitalization or closure is pursued, tracking specific metrics helps adjust actions accordingly.
  - **Key Metrics**:
    - **Community Activity Trends**: Track community engagement

metrics, such as discussion channel activity, contributor counts, and pull request frequency, to gauge shifts in interest.

- **Feedback and Sentiment**: Collect user and contributor feedback regularly to assess their needs and perceptions of the project's direction.
- **Milestone Completion**: For revitalization efforts, monitor progress on new goals (e.g., rebranding, feature pivots) to evaluate the effectiveness of the strategy.
- **Exit Preparedness**: For closure, ensure documentation and archiving completeness, gathering feedback on transition support for further adjustments if needed.
- ---

❖ **Outcome Scenarios**

➢ **Successful Revitalization**:

- If the revitalization actions result in renewed engagement, the project can shift out of the Decline phase and re-establish itself as a growing, sustainable entity. A brief return to the Growth phase may be necessary to rebuild momentum.

➢ **Organized Sunset**:

- In cases where closure is confirmed, the project formally enters the Abandonment phase, concluding with archived resources, documented accomplishments, and final communication to the community.

➢ **Dormancy Option**:

- If community interest or resources are expected to return in the future, a dormancy strategy may be employed. The project remains accessible but without active development or support, allowing it to be revisited if conditions improve.

❖ **Specific Criteria Reviewed:**

➢ **Decline in Active Contributors**

- **Metric**: A decreasing trend in the number of active contributors (e.g., a drop of 30% or more over six months).
- **Indicator**: Fewer active contributors suggest reduced community interest or engagement, a common sign of declining project health.

➢ **Increase in Issue Backlog and Unresolved Pull Requests**

- **Metric**: Growing number of open issues and pull requests, with an increasing average age (e.g., issues unresolved for 3+ months).
- **Indicator**: A backlog of issues and PRs that remain open for extended periods indicates a lack of maintenance and

responsiveness.

➢ **Reduction in New User Adoption and Usage**

- **Metric**: Decrease in the rate of new downloads, installations, or unique users over time (e.g., a 20% drop year-over-year).
- **Indicator**: Lower adoption rates suggest reduced relevance or satisfaction with the project, often leading to further decline in support and funding.

➢ Drop in Community Engagement
- **Metric**: Decreasing interactions on community platforms, such as forum posts, GitHub comments, or support queries.
- **Indicator**: Reduced community discussion and support signal that the user base may be disengaging, often because of a shift to alternatives or diminishing interest.

➢ Stagnation or Regression in Code Quality and Technical Debt
- **Metric**: Increase in code issues, unresolved technical debt, or lack of code updates (e.g., dependencies left outdated or security patches missed).
- **Indicator**: A lack of active development leads to a gradual decline in code quality and security, as dependencies age and new vulnerabilities arise.

➢ Decrease in Financial or Sponsorship Support
- **Metric**: Reduction in funding, grants, or donations (e.g., loss of major sponsors or a significant decrease in financial contributions).
- **Indicator**: Financial challenges are a key sign of decline, as they make it harder to fund maintenance, support, and community engagement activities.

➢ Increase in Forks and Migration to Alternatives
- **Metric**: Reduction in funding, grants, or donations (e.g., loss of major sponsors or a significant decrease in financial contributions).
- **Indicator**: Financial challenges are a key sign of decline, as they make it harder to fund maintenance, support, and community engagement activities.

Archive Phase

- **Stage Definition**
  - **Purpose**: The Abandonment phase represents the official end of active development and support. The focus is on closing out the project in a manner that respects the community's investment, preserves the project's achievements, and minimizes disruption for users.
  - **Key Characteristics**:
    - No further development, feature additions, or active maintenance is planned.
    - Minimal to no ongoing community engagement, but a clear communication plan for the transition.
    - Archiving of project resources for accessibility and legacy preservation.

- **Closure Planning and Communication**
  - **Goal**: Transparently communicate the decision to sunset the project, with clear timelines and reasons to help the community understand and prepare.
  - **Key Actions**:
    - **Announcement of Abandonment**: Formally announce the decision to abandon the project through community channels, including an explanation of the rationale, timeline for the closure, and next steps.
    - **Stakeholder Meetings**: Host final community and stakeholder meetings to discuss the project's end and address any final questions or concerns.
    - **Transition Guidelines**: Provide guidance on recommended alternatives or steps for users to transition to similar projects or tools within the ecosystem.

- **Legacy and Resource Preservation**
  - **Goal**: Preserve valuable project assets for historical reference or potential future use, ensuring that the knowledge and work are accessible even after the project's end.
  - **Key Actions**:
    - **Code and Documentation Archiving**: Archive all code,

documentation, and related assets in a publicly accessible repository (e.g., GitHub, IPFS, or a dedicated archive platform).
- **Resource Organization**: Organize files, documentation, and notable contributions in a clear structure to make the project's history easy to access and reference.
- **Educational and Community Contributions**: If applicable, consider offering project resources to educational institutions or open-source advocacy groups for ongoing learning or research.

❖ **Community Recognition and Celebration**
- ➢ **Goal**: Acknowledge the contributions of all involved, celebrating the achievements and impact the project has made within the community and ecosystem.
- ➢ **Key Actions**:
  - **Final Retrospective Event**: Host a final community retrospective or webinar to reflect on the project's journey, successes, and challenges, giving contributors a chance to share their experiences and insights.
  - **Recognition of Key Contributors**: Highlight and thank significant contributors, such as maintainers, long-term community members, and early supporters, with acknowledgments in the project's final communication and documentation.
  - **Commemorative Content**: Create a blog post, timeline, or video documenting the project's milestones and achievements to honor its impact within the Cardano ecosystem.

❖ **Monitoring and Final Review**
- ➢ **Goal**: Ensure that all closure-related actions are complete and that resources remain accessible for future reference.
- ➢ **Key Actions**:
  - **Post-Abandonment Checks**: Conduct a final review to confirm that all documentation, archiving, and community communications are finalized and available as planned.
  - **Legacy Access Reviews**: Periodically check that archived resources (e.g., code repositories, documentation) remain accessible and in good condition, ensuring they meet the needs of potential users or researchers.

- ❖ **Outcome**
  - ➢ **Completion of Project Lifecycle**:
    - The Abandonment Phase marks the official end of the project lifecycle. With all resources archived, the project's legacy is preserved, allowing the community to honor its contributions while freeing resources for new initiatives.
  - ➢ **Community and Ecosystem Impact**:
    - By completing this phase transparently and respectfully, the community's trust and goodwill are maintained, setting a positive example for future projects within the ecosystem.
- ❖ **Specific Criteria Reviewed:**
  - ➢ **Zero or Near-Zero Contributor Activity**
    - **Metric**: No new commits, pull requests, or code contributions over a defined period (e.g., six months or more).
    - **Indicator**: Lack of contributor activity is a primary signal of abandonment, suggesting no one is maintaining or improving the project.
  - ➢ No Issue Resolution or Triage
    - **Metric**: No issues closed, triaged, or responded to within several months or longer.
    - **Indicator**: When issues and bugs remain unaddressed, it's clear that there is no support structure left for the project.
  - ➢ Complete Decline in Community Interaction
    - **Metric**: No new comments, discussions, or support questions being addressed on GitHub, forums, or community channels.
    - **Indicator**: A complete lack of community interaction indicates that both the user base and maintainers have disengaged from the project.
  - ➢ Unmaintained Dependencies and Security Vulnerabilities
    - **Metric**: Multiple outdated dependencies and known security vulnerabilities without patches or updates.
    - **Indicator**: An abandoned project typically stops receiving security updates, making it risky and outdated for users.
  - ➢ Significant Drop or Complete Cessation in Downloads and Usage
    - **Metric**: Sharp decline or near-zero downloads, installations, or user activity over time.
    - **Indicator**: A loss in active installations and user engagement shows the project is no longer being used or trusted in production environments.

- No Financial Support or Sponsorships
  - **Metric**: Complete cessation of funding or sponsorships, with no incoming financial support.
  - **Indicator**: An absence of financial support typically follows disengagement from maintainers, reinforcing the project's unsustainability.
- **Increasing Forks and Migration to Alternatives**
  - **Metric**: Users forking the project without returning contributions upstream or moving entirely to alternative projects.
  - **Indicator**: Forks and migrations demonstrate that remaining users are abandoning the original project in favor of creating independent versions or using more active alternatives.