woven by TOYOTA

What ELISA's Lighthouse OSS Best Practices Reveal About LLVM's State of Practice

ELISAの「Lighthouse OSSベストプラクティス」から見える LLVMプロジェクトの実務と成熟度

December 11, 2025

2025年12月11日

Wendi Urribarri

Functional Safety Engineer 機能安全エンジニア

Global Safety & Quality LoB グローバル・セーフティ＆クオリティ事業本部

wendi.urribarri@woven.toyota

woven by TOYOTA

What ELISA’s Lighthouse OSS Best Practices Reveal About LLVM’s State of Practice

ELISAの「Lighthouse OSSベストプラクティス」から見える LLVMプロジェクトの実務と成熟度

December 11, 2025

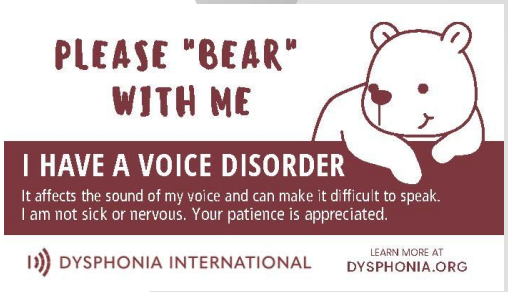
2025年12月11日

Wendi Urribarri

Functional Safety Engineer 機能安全エンジニア

Global Safety & Quality LoB グローバル・セーフティ＆クオリティ事業本部

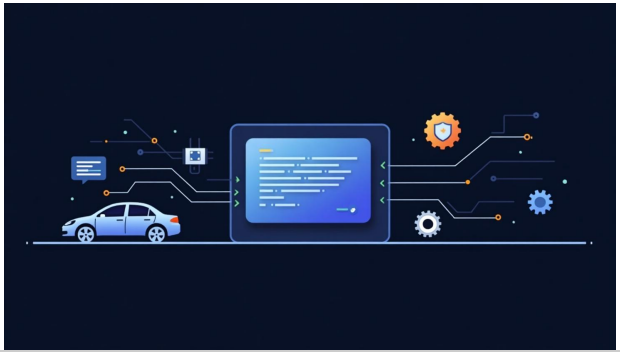
wendi.urribarri@woven.toyota



Why this talk? なぜこの話をするのか？

Enabling Linux in Safety Applications (ELISA)

セーフティクリティカルな　システムにおけるLinuxの活用を推進するコミュニティ（ ELISA）



The LLVM Compiler Infrastructure

LLVMコンパイラインフラストラクチャ

Safety-critical & SDV projects depend on open source software, but we often lack a shared language to talk about OSS quality and maturity

安全関連システムやSDVプロジェクトはOSSに大きく依存していますが、
OSSの「品質」や「成熟度」について語る共通言語がまだ十分にありません。

Why this talk? なぜこの話をするのか？

ELISA Lighthouse OSS SIG

- Best practices checklist
- Evaluate and document established open source development best practices
- Provide an assessment guide for the user to rate the quality of open source projects
- ベストプラクティスのチェックリストを提供する
- 確立されたオープンソース開発のベストプラクティスを評価し、文書化する
- OSSプロジェクトの「品質」を評価するためのガイドを提供する

LLVM Qualification Group

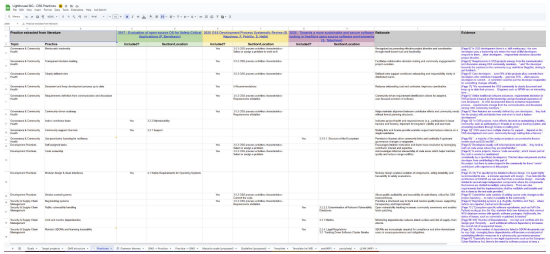
- Enable confidence in the use of software tools (e.g., compilers) and libraries (e.g., runtimes)
- Guidance on processes, methods, and tools for qualification of LLVM components
- コンパイラなどのソフトウェアツールや、ランタイムなどのライブラリを「安心して使える」ようにすることを目指す
- LLVMコンポーネントの認証に関するプロセス・メソッド・ツールのガイダンスを提供する

Eclipse SDV

- Build open, modular software stacks for software-defined vehicles using open source
- Bring car makers and tech companies together to co-develop open software platform for future vehicles
- オープンソースを活用して、ソフトウェア定義車両（SDV）のためのモジュール型のソフトウェアスタックを構築する
- 自動車メーカーとテック企業をつなぎ、将来の車両向けオープンソフトウェアプラットフォームを共同開発する

ELISA Lighthouse OSS Best Practices checklist

ELISA Lighthouse OSS ベストプラクティス・チェックリスト



What the checklist is

Structured list of OSS “best practices” observed in literature about OSS development processes

OSS開発プロセスに関する文献などで観察された
OSSの「ベストプラクティス」を体系的に整理したリスト

Organized into areas:

- Governance & Community Health
- Development Practices
- Security & Supply Chain Management
- QA & Engineering Discipline
- Documentation
- Sustainability & Long-term Viability

以下のような分野ごとに整理：

- ガバナンスとコミュニティの健全性
- 開発プラクティス
- セキュリティとサプライチェーン管理
- 品質保証（QA）とエンジニアリング規律
- ドキュメンテーション
- 持続可能性と長期的な継続性

What it is *not*

Not a certification
Not a pass/fail scorecard
認証制度ではありません
合否判定のスコアカードでもありません

It’s a **conversation tool**:

- *What do we do today?*
- *Where are the gaps or tensions?*

これは「対話のためのツール」です：

- *いま、私たちは何をやっているのか？*
- *どこにギャップやモヤモヤがあるのか？*

Status & limitations of this checklist このチェックリストの現状と限界

01 It’s a work in progress
まだ進行中である

- The Lighthouse OSS Best practices checklist is still being refined in ELISA
- Items may change as we analyze more projects and feedback
- Lighthouse OSSベストプラクティス・チェックリストは、ELISAの中で現在もブラッシュアップ中です
- さらに多くのプロジェクトやフィードバックを分析することで、項目が変わっていく可能性があります

02 Maturity scale under discussion
成熟度スケールは検討・試行中である

- Current Limited / Good / Strong scale is experimental
- We’re still exploring how to separate *process robustness* from *evidence confidence*
- 「Limited / Good / Strong」の成熟度スケールは、まだ試行中の案にすぎません
- プロセスの強さ と、エビデンスへの信頼度 をどう分けて評価するかを議論している段階です

03 The LLVM assessment is an experiment
LLVMに対する評価は、ひとつの実験的ケーススタディである

- Based on public information + my interpretation
- It’s meant to start conversations, not to produce a “final score”
- LLVMに対する評価は、公開情報と私自身の解釈に基づく、ひとつの試み（実験）です
- 最終的な「スコア」をつけることが目的ではなく、議論を始めるきっかけにしたいと考えています

Maturity lens used for LLVM LLVMに適用した成熟度レンズ

Simple 3-level lens シンプルな 3段階の成熟度レンズ

Strong
practice is clearly present, documented, and used; evidence is “easy to find”

実務として明確に定着しており、文書化され、実際に使われている。エビデンスも比較的「見つけやすい」状態。

Good
practice exists, but with limitations (in coverage, consistency, or clarity)

実務として存在しているが、カバレッジ・一貫性・分かりやすさなどに制約がある。

Limited
we see signs or ingredients, but no systematic project-wide practice

いくつかの兆候や要素は見えるが、プロジェクト全体として体系的な実務にはなっていない。

Inputs from LLVM LLVMからのインプット

Public documentation and policies
Dev Policy, governance docs, etc.

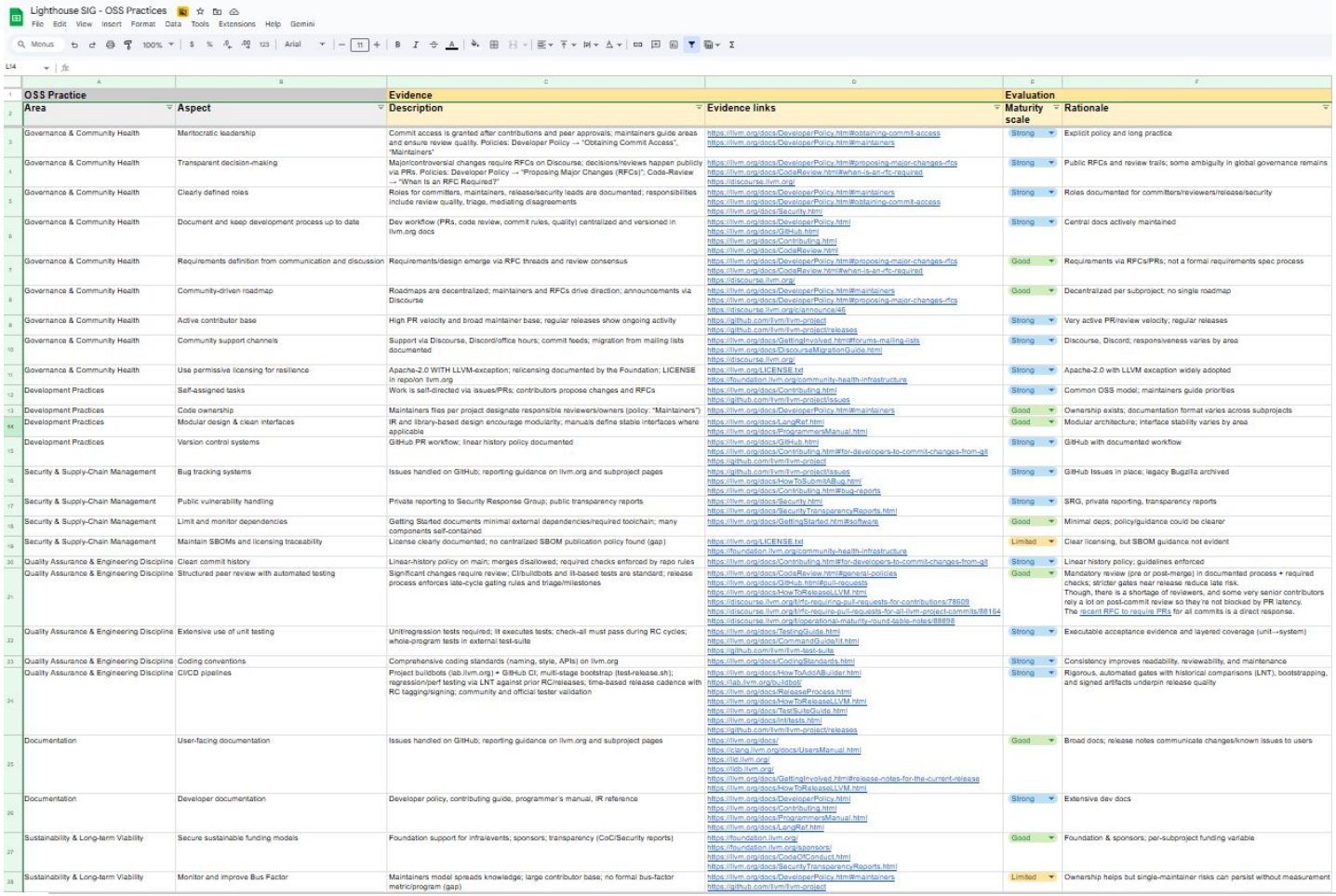
公開ドキュメントとポリシー（開発ポリシー、ガバナンス文書など）

Public infrastructure
GitHub repo, CI configuration, issue tracker, security policy

公開インフラ（GitHubリポジトリ、CI設定、Issueトラッカー、セキュリティポリシーなど）

Community artifacts
RFCs, Discourse threads, release notes, etc.

コミュニティの成果物（RFC、Discourseスレッド、リリースノートなど）



Lighthouse SIG - OSS Practices

OSS Practice Area	Aspect	Evidence Description	Evidence links	Evaluation Maturity scale	Rationale
Governance & Community Health	Meritocratic leadership	Commit access is granted after contributions and peer approvals; maintainers guide areas and ensure review quality. Policies: Developer Policy → "Obtaining Commit Access", "Maintainers"	https://llvm.org/docs/DeveloperPolicy.html#obtaining-commit-access https://llvm.org/docs/DeveloperPolicy.html#maintainers	Strong	Explicit policy and long practice
Governance & Community Health	Transparent decision-making	Major/controversial changes require RFCs on Discourse; decisions/reviews happen publicly via PRs. Policies: Developer Policy → "Proposing Major Changes (RFCs)"; Code-Review → "When is an RFC Required?"	https://llvm.org/docs/DeveloperPolicy.html#proposing-major-changes-rfcs https://llvm.org/docs/CodeReview.html#when-is-an-rfc-required https://discourse.llvm.org/	Strong	Public RFCs and review trails; some ambiguity in global governance remains
Governance & Community Health	Clearly defined roles	Roles for committers, maintainers, release/security leads are documented; responsibilities include review quality, triage, mediating disagreements	https://llvm.org/docs/DeveloperPolicy.html#maintainers https://llvm.org/docs/DeveloperPolicy.html#obtaining-commit-access https://llvm.org/docs/Security.html	Strong	Roles documented for committers/reviewers/release/security
Governance & Community Health	Document and keep development process up to date	Dev workflow (PRs, code review, commit rules, quality) centralized and versioned in llvm.org docs	https://llvm.org/docs/DeveloperPolicy.html https://llvm.org/docs/GitHub.html https://llvm.org/docs/Contributing.html https://llvm.org/docs/CodeReview.html	Strong	Central docs actively maintained
Governance & Community Health	Requirements definition from communication and discussion	Requirements/design emerge via RFC threads and review consensus	https://llvm.org/docs/DeveloperPolicy.html#proposing-major-changes-rfcs https://llvm.org/docs/CodeReview.html#when-is-an-rfc-required https://discourse.llvm.org/	Good	Requirements via RFCs/PRs; not a formal requirements spec process
Governance & Community Health	Community-driven roadmap	Roadmaps are decentralized; maintainers and RFCs drive direction; announcements via Discourse	https://llvm.org/docs/DeveloperPolicy.html#maintainers https://llvm.org/docs/DeveloperPolicy.html#proposing-major-changes-rfcs https://discourse.llvm.org/c/announce/46	Good	Decentralized per subproject; no single roadmap
Governance & Community Health	Active contributor base	High PR velocity and broad maintainer base; regular releases show ongoing activity	https://github.com/llvm/llvm-project https://github.com/llvm/llvm-project/releases	Strong	Very active PR/review velocity; regular releases
Governance & Community Health	Community support channels	Support via Discourse, Discord/office hours; commit feeds; migration from mailing lists documented	https://llvm.org/docs/GettingInvolved.html#forums-mailing-lists https://llvm.org/docs/DiscourseMigrationGuide.html https://discourse.llvm.org/	Strong	Discourse, Discord; responsiveness varies by area
Governance & Community Health	Use permissive licensing for resilience	Apache-2.0 WITH LLVM-exception; relicensing documented by the Foundation; LICENSE in repo/on llvm.org	https://llvm.org/LICENSE.txt https://foundation.llvm.org/community-health-infrastructure	Strong	Apache-2.0 with LLVM exception widely adopted
Development Practices	Self-assigned tasks	Work is self-directed via issues/PRs; contributors propose changes and RFCs	https://llvm.org/docs/Contributing.html https://github.com/llvm/llvm-project/issues	Strong	Common OSS model; maintainers guide priorities
Development Practices	Code ownership	Maintainers files per project designate responsible reviewers/owners (policy: "Maintainers")	https://llvm.org/docs/DeveloperPolicy.html#maintainers	Good	Ownership exists; documentation format varies across subprojects
Development Practices	Modular design & clean interfaces	IR and library-based design encourage modularity; manuals define stable interfaces where applicable	https://llvm.org/docs/LangRef.html https://llvm.org/docs/ProgrammersManual.html	Good	Modular architecture; interface stability varies by area
Development Practices	Version control systems	GitHub PR workflow; linear history policy documented	https://llvm.org/docs/GitHub.html https://llvm.org/docs/Contributing.html#for-developers-to-commit-changes-from-git https://github.com/llvm/llvm-project	Strong	GitHub with documented workflow
Security & Supply-Chain Management	Bug tracking systems	Issues handled on GitHub; reporting guidance on llvm.org and subproject pages	https://github.com/llvm/llvm-project/issues https://llvm.org/docs/HowToSubmitABug.html https://llvm.org/docs/Contributing.html#bug-reports	Strong	GitHub Issues in place; legacy Bugzilla archived
Security & Supply-Chain Management	Public vulnerability handling	Private reporting to Security Response Group; public transparency reports	https://llvm.org/docs/Security.html https://llvm.org/docs/SecurityTransparencyReports.html	Strong	SRG, private reporting, transparency reports
Security & Supply-Chain Management	Limit and monitor dependencies	Getting Started documents minimal external dependencies/required toolchain; many components self-contained	https://llvm.org/docs/GettingStarted.html#software	Good	Minimal deps; policy/guidance could be clearer
Security & Supply-Chain Management	Maintain SBOMs and licensing traceability	License clearly documented; no centralized SBOM publication policy found (gap)	https://llvm.org/LICENSE.txt https://foundation.llvm.org/community-health-infrastructure	Limited	Clear licensing, but SBOM guidance not evident
Quality Assurance & Engineering Discipline	Clean commit history	Linear-history policy on main; merges disallowed; required checks enforced by repo rules	https://llvm.org/docs/Contributing.html#for-developers-to-commit-changes-from-git	Strong	Linear history policy; guidelines enforced
Quality Assurance & Engineering Discipline	Structured peer review with automated testing	Significant changes require review; CI/buildbots and lit-based tests are standard; release process enforces late-cycle gating rules and triage/milestones	https://llvm.org/docs/CodeReview.html#general-policies https://llvm.org/docs/GitHub.html#pull-requests https://llvm.org/docs/HowToReleaseLLVM.html https://discourse.llvm.org/t/rfc-requiring-pull-requests-for-contributions/78609 https://discourse.llvm.org/t/rfc-require-pull-requests-for-all-llvm-project-commits/68164 https://discourse.llvm.org/t/operationalize-maturity-round-table-notes/68856	Good	Mandatory review (pre or post-merge) in documented process + required checks; stricter gates near release reduce late risk. Though, there is a shortage of reviewers, and some very senior contributors rely a lot on post-commit review so they're not blocked by PR latency. The recent RFC to require PRs for all commits is a direct response.
Quality Assurance & Engineering Discipline	Extensive use of unit testing	Unit/regression tests required; lit executes tests; check-all must pass during RC cycles; whole-program tests in external test-suite	https://llvm.org/docs/TestingGuide.html https://llvm.org/docs/CommandGuide/lit.html https://github.com/llvm/llvm-test-suite	Strong	Executable acceptance evidence and layered coverage (unit→system)
Quality Assurance & Engineering Discipline	Coding conventions	Comprehensive coding standards (naming, style, APIs) on llvm.org	https://llvm.org/docs/CodingStandards.html	Strong	Consistency improves readability, reviewability, and maintenance
Quality Assurance & Engineering Discipline	CI/CD pipelines	Project buildbots (lab.llvm.org) + GitHub CI; multi-stage bootstrap (test-release.sh); regression/perf testing via LNT against prior RC/releases; time-based release cadence with RC tagging/signing; community and official tester validation	https://llvm.org/docs/HowToAddABuilder.html https://lab.llvm.org/buildbot/ https://llvm.org/docs/ReleaseProcess.html https://llvm.org/docs/HowToReleaseLLVM.html https://llvm.org/docs/TestSuiteGuide.html https://llvm.org/docs/lit/tests.html https://github.com/llvm/llvm-project/releases	Strong	Rigorous, automated gates with historical comparisons (LNT), bootstrapping, and signed artifacts underpin release quality
Documentation	User-facing documentation	Issues handled on GitHub; reporting guidance on llvm.org and subproject pages	https://llvm.org/docs/ https://clang.llvm.org/docs/UsersManual.html https://lld.llvm.org/ https://lldb.llvm.org/ https://llvm.org/docs/GettingInvolved.html#release-notes-for-the-current-release https://llvm.org/docs/HowToReleaseLLVM.html	Good	Broad docs; release notes communicate changes/known issues to users
Documentation	Developer documentation	Developer policy, contributing guide, programmer's manual, IR reference	https://llvm.org/docs/DeveloperPolicy.html https://llvm.org/docs/Contributing.html https://llvm.org/docs/ProgrammersManual.html https://llvm.org/docs/LangRef.html	Strong	Extensive dev docs
Sustainability & Long-term Viability	Secure sustainable funding models	Foundation support for infra/events; sponsors; transparency (CoC/Security reports)	https://foundation.llvm.org/ https://foundation.llvm.org/sponsors/ https://llvm.org/docs/CodeOfConduct.html https://llvm.org/docs/SecurityTransparencyReports.html	Good	Foundation & sponsors; per-subproject funding variable
Sustainability & Long-term Viability	Monitor and improve Bus Factor	Maintainers model spreads knowledge; large contributor base; no formal bus-factor metric/program (gap)	https://llvm.org/docs/DeveloperPolicy.html#maintainers https://github.com/llvm/llvm-project	Limited	Ownership helps but single-maintainer risks can persist without measurement

Strong

Good

Limited

Other checklists

Other OSS best practices checklists exist (e.g., OpenSSF Best Practices)

LLVM has already been evaluated once (2024) against the OpenSSF checklist

OpenSSF Best Practices など、他のOSSベストプラクティスのチェックリストも存在する

LLVMは既にOpenSSF のチェックリストに対する評価が行われている

Lighthouse SIG - OSS Practices - LLVM (WIP)

LLVM “state of practice” at a glance LLVMの「現状の実務」から見える所見

Area	Observations
Governance & community health ガバナンスとコミュニティの健全性	Many **strong** practices (roles, decision-making, process docs, open RFCs) 役割・意思決定・プロセス文書RFC公開など、効果的なプラクティスが多い
Development practices 開発プラクティス	Mix of **strong** and **good** (modern Git/PR workflow, modular architecture, some API stability variation) モダンなGit/PRワークフローやモジュールアーキテクチャなど、強みと弱みが混在(API安定性はエリアによってばらつきあり)
QA & engineering discipline QA & テスト／レビューに関わる部分	**Strong** testing & CI foundations (multi-layer tests, automated gates, clear expectations) but peer review is **uneven** in practice 強力なテスト基盤やCI基盤（多層テスト、自動化ゲート、明確な期待値）が存在するが、実際のピアレビューではばらつきがある
Security & supply chain セキュリティとサプライチェーン管理	**Strong** bug tracking and vulnerability handling; SBOM and dependency monitoring is **limited** in today’s public evidence 強力な不具合追跡と脆弱性対応SBOMと依存関係ガバナンスは現在の公開情報では限定的である
Documentation ドキュメンテーション	**Strong** contributor / reference docs; user docs generally **good**. Both have gaps, especially around fast-changing internals 貢献者向け/リファレンスドキュメントは充実しており、ユーザー向けドキュメントも概ね良好です。ただし、特に内部構造が急速に変化する部分では、両者にギャップが見られる
Sustainability & long-term viability 持続可能性と長期的継続性	Funding and foundation support **good**; bus factor **limited** in some key areas (high reliance on a few experts) 資金やFoundationによる支援は良好だが、重要な領域で少数のエキスパートに依存する「バス係数」の問題がある

Governance & community health ガバナンスとコミュニティの健全性

A strong foundation 強い基盤

Strengths 強み

- Meritocratic leadership
- Transparent decision-making
- Clearly defined roles
- Process documentation up to date
  - メリトクラシーに基づくリーダーシップ
  - 透明性の高い意思決定プロセス
  - 役割が明確に定義されている
  - プロセス文書が整備・更新されている

Tensions 課題・モヤモヤ

- Global governance can still be hard to explain from outside
- Project size means not everyone experiences governance the same way
- Sub-projects may feel slightly different in responsiveness and visibility
  - 外から見ると、全体のガバナンス構造が分かりにくい部分がある
  - プロジェクトが非常に大きいため、ガバナンスの経験も人によって異なる
  - サブプロジェクトのレスポンスや可視性はやや異なるように感じられる場合がある

Takeaways まとめ

- Governance is not a blocker, it's a strong base to build assurance narratives on
- The community is where requirements are
  - ガバナンス自体は「ボトルネック」ではなく、保証ストーリーを組み立てるための強い土台になっている
  - 要求や期待はコミュニティの中にある、という前提で考える必要がある

Development practices 開発プラクティス

Modern and mostly predictable モダンで、おおむね予測可能

Strengths 強み

- Self-assigned tasks
- Version control & workflow
- Modular design & clean interfaces
  - 自律的にタスクをアサインする文化
  - バージョン管理とワークフローが整備されている（Git / PR ベースなど）
  - モジュール設計とクリーンなインタフェース設計

Tensions 課題・モヤモヤ

- Interface stability varies by area; some components are “stable”, others more experimental
- Ownership is documented, but the format and depth vary across sub-projects
  - インタフェースの安定度は領域によって異なり、「安定」している部分と実験的な部分が混在している
  - オーナーシップは文書化されているものの、その書き方や詳細さはサブプロジェクトごとにばらつきがある

Takeaways まとめ

- There is a predictable, documented way to contribute and consume
- You still need to identify which pieces are stable enough for your use case
  - コントリビュート／利用するための「手順」は文書化され、予測しやすい
  - とはいえ、自分のユースケースにとって「どの部分が十分に安定しているか」は見極めが必要

QA & engineering discipline QAとエンジニアリングの規律

Strong foundations, some visible cracks 強い土台、見えてきたひび

Strengths 強み

- Clean commit history
- Automated, extensive testing
- Coding conventions
- CI/CD pipelines
  - クリーンなコミット履歴（履歴方針の徹底）
  - 自動化された広範なテスト
  - コーディング規約の整備
  - CI/CDパイプラインの活用

Tensions 課題・モヤモヤ

- Reviewer bandwidth is limited
- Post-merge review is used heavily, especially by very experienced contributors
  - レビュアーのリソースが限られている
  - 特に経験豊富なコントリビュータほど、マージ後レビューに頼ることが多い

Takeaways まとめ

- LLVM is already doing many things that safety standards like
- But, we cannot assume every change received robust peer review
  - LLVMは、機能安全の観点から好ましいプラクティスをすでに多く取り入れている一方で、レビューの運用には改善の余地があることも見えてきた
  - でも、すべての変更が十分にしっかりとしたピアレビューを受けているとは仮定できません

Security & supply chain セキュリティとサプライチェーン

Mixed pictures 強弱混在の評価

Strengths 強み

- Bug tracking
- Public vulnerability handling
- License traceability
  - バグトラッキングの仕組み
  - 公開された脆弱性対応プロセス
  - ライセンス情報の追跡可能性

Tensions 課題・モヤモヤ

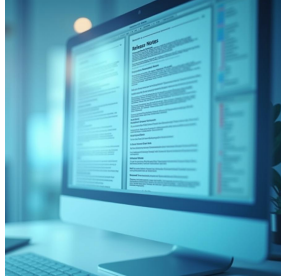
- No automated dependencies monitoring
- No SBOMs
  - 依存関係を自動でモニタリングする仕組みは整っていない
  - プロジェクト全体としてのSBOM（ソフトウェア部品表）は存在しない

Takeaways まとめ

- Security vulnerability response is mature
- If you require SBOMs, you will need additional internal work to fill that gap
  - セキュリティインシデント対応そのものは比較的成熟している
  - SBOMが必須の組織では、LLVM側の情報だけでは足りず、自社側での補完作業が必要になる

Documentation & sustainability ドキュメンテーションと持続可能性

Useful today, uncertain tomorrow 今日は十分、明日は不透明



Strengths 強み

- Developer documentation
- User documentation
- Funding model
- 開発者向けドキュメント
- ユーザー向けドキュメント
- ファンディングモデル（LLVM Foundation など）の存在

Tensions 課題・モヤモヤ

Bus factor: ownership helps but some areas still depend on a very small number of experts

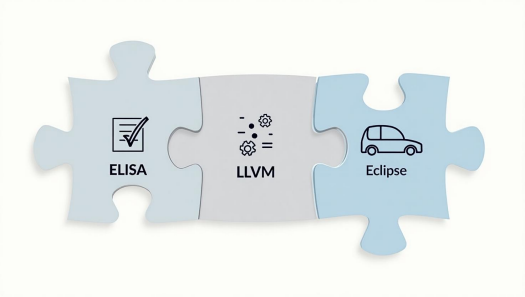
バス係数の問題：オーナーシップは明確だが、一部の重要領域がごく少数のエキスパートに依存している

Takeaways まとめ

- From a process point of view, LLVM looks strong
- But if your product lifecycle is 10-15+ years, you must ask: "*Are the critical components we rely on backed by more than 1–2 maintainers?*", "*What if those people change jobs?*"
- プロセス面だけを見ればLLVMはかなり強い
- しかし、製品ライフサイクルが10〜15年以上にわたる場合、「依存している重要コンポーネントは、本当は1〜2人以上で支えられているのか？」「もしその人たちがいなくなったらどうなるのか？」という問いを避けることはできません。

How ELISA, LLVM & Eclipse SDV can share practices

ELISA・LLVM・Eclipse SDVがプラクティスを共有するには



Instead of reinventing in isolation...

それぞれがバラバラに「車輪の再発明」をするのではなく...

- **ELISA Lighthouse OSS SIG**: provides a good checklist of best practices and a growing body of OSS case studies
- **LLVM community**: provides a real, high-impact project and strong existing practices
- **Eclipse SDV**: represents SDV platforms and industrial users who need assurance narratives

- ELISA Lighthouse OSS SIG：ベストプラクティスのチェックリストと、OSSケーススタディの蓄積を提供
- LLVMコミュニティ：実際に広く使われている大規模プロジェクトと、既に存在する強いプラクティスを提供
- SDVプラットフォームや産業ユーザーの視点から、どのような保証ストーリーが必要かを提示

Opportunities

機会

- Shared maturity vocabulary (e.g., Limited, Good, Strong)
- Small set of evidence patterns that can be reused across LLVM-based projects
- Experiments around bus-factor reduction or SBOM practices in LLVM
- Use the LLVM Qualification WG as a bridge to discuss findings with the LLVM community and propose concrete improvements

- Limited / Good / Strong など、共通の成熟度ボキャブラリを共有する
- LLVMベースのプロジェクト間で再利用できる、小さな「エビデンス・パターン集」を作る
- LLVMでのバス係数改善やSBOM導入の実験を、共同の取り組みとして進める
- LLVM Qualification WG を橋渡しとして利用し、LLVM コミュニティと調査結果を議論し、具体的な改善を提案します

Closing & Invitation

結びとコミュニティへ

Key messages
メインメッセージ

- LLVM already demonstrates many best practices ELISA tracks
- Tensions remain in QA (peer review), supply-chain transparency (SBOMs, dependency tracking), and sustainability and bus factor

- LLVMは、ELISAがチェックしている多くのベストプラクティスを すでに実践している
- 一方で、QA（特にピアレビューの運用）、サプライチェーンの透明性§BOM・依存関係トラッキング）、持続可能性とバス係数には、まだ課題が残っている

Invitation
コミュニティ参加のお誘い

If you:

- Contribute to LLVM or Linux
- Work on SDV platforms at Eclipse
- Or care about open source in safety-critical contexts

...let's **compare notes and share practices**, not build separate checklists and audits for every community

もし皆さんが

- LLVMにコントリビュートしている
- Eclipse SDVのプラットフォーム開発に関わっている
- もしくは、安全関連領域でのOSS活用に関心がある

のであれば、コミュニティごとに別々のチェックリストや監査を作るのではなく、一緒に「気づき」と「プラクティス」を共有していければうれしいです。

Thank you

My email: **wendi.urribarri@woven.toyota**

My LinkedIn profile: **www.linkedin.com/in/uwendi**