

SWEST28 基調講演

# AI時代のキョウソウ戦略 ～ソフトウェア・ディファインドがもたらす価値創造の進化とは～

株式会社 東芝 総合研究所  
デジタルイノベーション技術センター

**小林 良岳**

# 本日のTakeaway

- 上物の価値は下回りが決める。だから土台は仲間と創る
- 社外はオープンソース、社内はインナーソース、  
協調領域も競争領域も加速！
- AIがさらに活動を加速するが大切なのは人！

# 本日のTakeaway

- 頼り頼られる仲間を大切に！

**Think OPEN**

# 自己紹介



- **名 前** : 小林 良岳 (こばやし よしたけ)
- **所 属** : 株式会社 東芝 総合研究所  
デジタルイノベーション技術センター  
副センター長 / シニアフェロー
- **略 歴** :
  - ・ EDS Australia [システムアドミニストレータ]
  - ・ 電気通信大学 [助教]
  - ・ 東芝 [～今に至る]
- **専 門** :
  - ・ オペレーティングシステム (OS)
  - ・ オープンソースソフトウェア (OSS)



## ● 社外活動 : Linux®/OSS関係の活動多め

- TOPPERS Project 理事 (2010-)
- CE Linux Forum Architecture Group Member (2010-)
- Core Embedded Linux Project SC Chair (2015-)
- Civil Infrastructure Platform Project TSC Chair (2016-)
- Open Invention Network Technical Advisory Council (2019-)
- インナーソース Commons Foundation Member (2025-)
- The Linux Foundation Board Member (2026-)
- Embedded Linux Conference Program Committee (2014-)
- Open Source Summit Japan Program Committee (2024-)
- 電子情報通信学会 常任査読委員 (2008-)
- OpenChain Project Board Member (2018-2020)

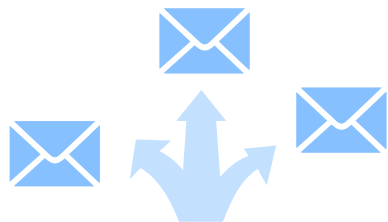
**製品適用 : 技術適用から製品化まで関わった最初の製品 (2010年)**



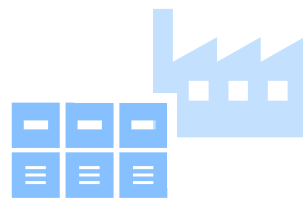
**OSSを用いて作った自動改札機としては国内初！**

# 製品適用：関わった製品例（直接的・間接的）

郵便区分機



産業用  
コントローラ



放送システム



気象レーダ



発電制御  
コントローラ



エレベーター  
群管理システム



マルチファンクション  
プリンター

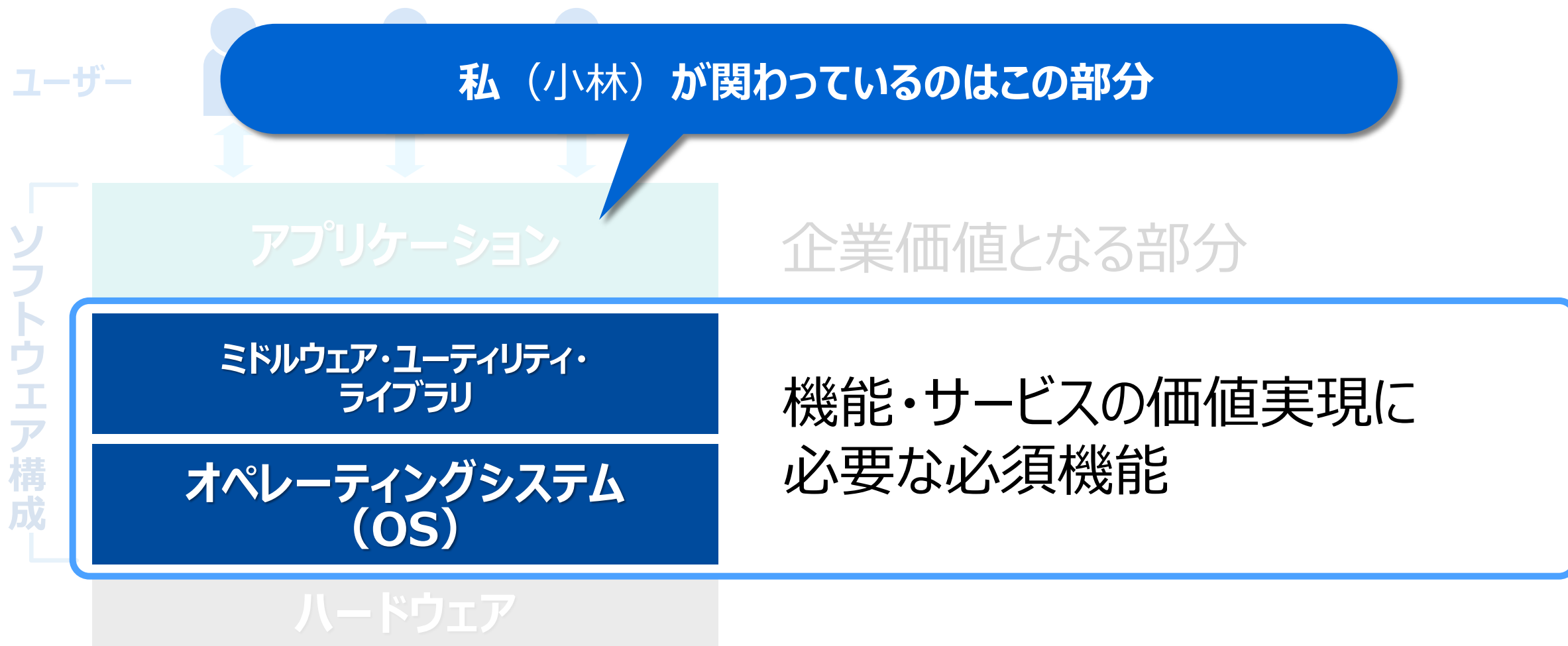


… 他たくさん  
(70以上)

なぜこれだけ多くの製品に関われたの？

# 小林の仕事の中心（OS/OSS = 共通ソフトウェア）

共通利用されるソフトウェアの研究開発を担当することで多事業に貢献



# Contents

- 00** 自己紹介：関わってきた製品と、共通ソフトウェア
- 01** 導入：AIが迫る変革
- 02** 目的地：ソフトウェア・ディファインド
- 03** よりどころ：価値を作る3つの領域
- 04** 進み方：Collaboration
- 05** 結論：価値創造のサイクル

# 1. Introduction

AIが迫る「作り方」と「組織」の変革

# AIが変えるのは「作り方」だけではない

「書く時間」より、「レビューし、判断する時間」が主役に

AI生成コードの  
デバッグは時間がかかる

45.2%

動くコードと欲しいコードの  
ギャップは人が埋める

(改善に向かってはいるが...)

PRレビュー  
時間の増加

+91%

自分が書いていないコードは理解  
に時間がかかる

コードベースの  
複雑性

+40%

作れる量が増えるほど、  
「何を作らないか」の重みが増す

# 組み込みでは、素朴なAI適用は効きにくい



## 制約が強い領域

### 規格・セキュリティ認証など、その説明責任がついて回る

- リアルタイム性、限られたリソース、止まらないこと
- コーディング規約、ニッチなハードウェア



## 学習データが 少ない世界

### 世に出ているコードが少なく、AIの得意分野から外れる

- 10年～数十年、そのコードを誰が保守するのか

「作り方」と「組織」をどう変えるかが問われる

## 今日の発表の地図：目的地・よりどころ・進み方

AIが、そのすべてを貫く縦糸になったとき、人はどのように振る舞うか？  
AIは縦糸。今日の本論は横糸であるOSSとインナーソース

| 目的地                                                                                                                                                                     | よりどころ                                                                                                                                                                | 進み方                                                                                                                                                                       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>ソフトウェア・ディファインド</p>  <ul style="list-style-type: none"><li>●出荷後も価値を作り続ける状態</li></ul> | <p>価値を作る3つの領域</p>  <ul style="list-style-type: none"><li>●領域ごとに、打つ手を変える</li></ul> | <p>Collaboration</p>  <ul style="list-style-type: none"><li>●外はOSS、内はインナーソース</li></ul> |

# 「キョウソウ」には、3つのタイプがある

同じような言葉でも、向き合い方はちがう



競争・協調・共創 — この3つを行き来する

# 2. Destination

ソフトウェア・ディファインドによって変わるもの変わらないもの

今は前提の確認 — 競争・協調・共創のどれにも共通する「目的地」

| 目的地                                                                                                                        | よりどころ                                                                                                                 | 進み方                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・ディファインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | Collaboration<br><br>●外はOSS、内はインナーソース |

## ソフトの柔軟性と、ハードの信頼性を配分し直すコンセプト



何を目指すのか

実現したい価値や目的に対して、ソフトの柔軟性と、ハードの信頼性という特性をソフトウェアの実装を通して最大限に生かす

- そのために機能分担を見直し、将来に備える



新しい概念ではない

もとはネットワークをどう適切に接続するかという文脈で広まった

ソフトがハードを管理するようになり、改めて注目される



広がる領域

車載（SDV）から、産業機器、家電、インフラ、医療機器まで波及

# なぜソフトウェア・ディファインドなのか？

## 市場・制度・技術、3つの圧力が同時にかかっている

### 市場からの圧力

製品の差は、ハード性能ではなく、ソフトの体験で決まりはじめた

- 買った後も良くなり続けることが、選ばれる条件になった



### 制度からの圧力

CRA（サイバーレジリエンス法）が、出荷後の更新責任を求める

- 長期のセキュリティ対応が、売るための前提条件になる



### 技術からの圧力

クラウドとAIが、機能を素早く更新・追加し続けられるようにした

- 出荷後の機能修正・向上が、例外ではなく前提になる



# 社会インフラに実際に起きていること

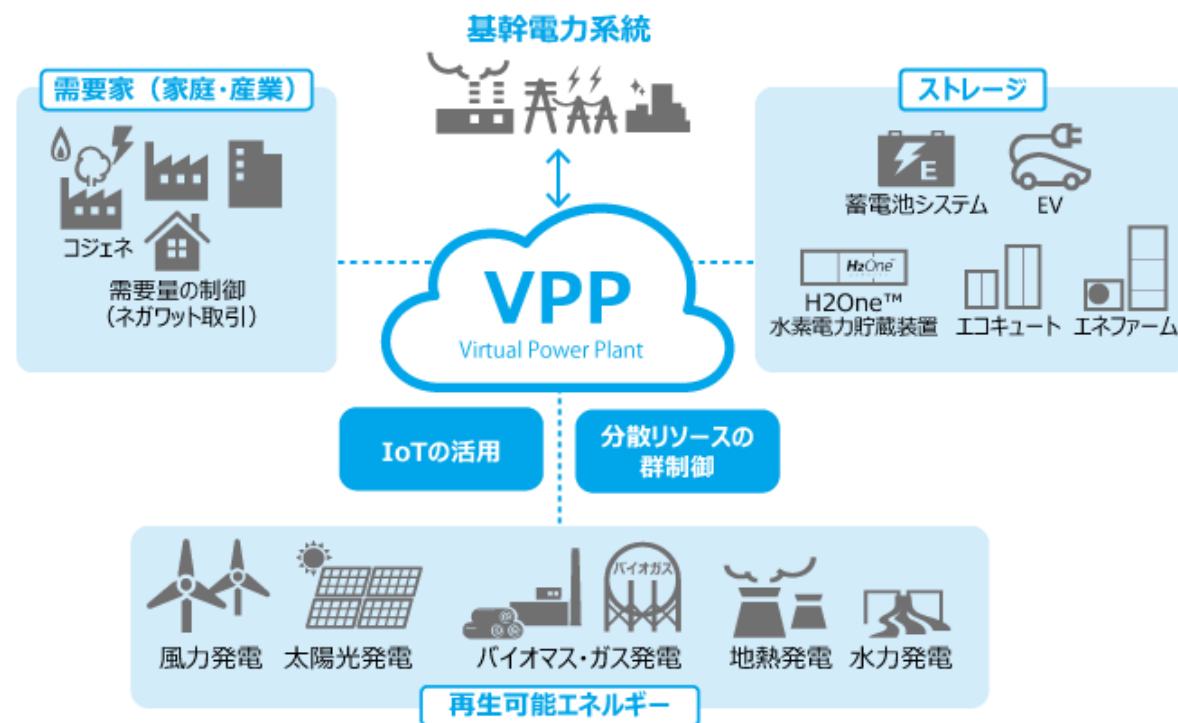
## CPS(\*)の常態化

- クラウド側(Cyber)と機器側(Physical)の機能が連携
- 機器のもつデータをクラウド側処理
- 機器制御の一部をクラウドで実行

## 機器側機能の特化

- 即時性の高い厳密なタイミング制御
- 機器の信頼性や安全性確保
- セキュアな環境の維持

## VPP (バーチャルパワープラント : 仮想発電所)



Ref: [Toshiba Clip](#) | ハードとソフトを、デジタルでつなぐ ～サステナビリティの課題解決へ、東芝の挑戦

外部とコミュニケーションしながら社会全体での最適化を目指している

# 出荷はゴールではなく、スタートになる

「作って終わり」から「動かし続けて価値を伸ばす」へ

## Before 出荷時が品質のピーク

仕様凍結 → 作り込み  
→ 出荷 → 現状維持

- 以後は保守中心で、価値は増えない



## After 出荷後に価値が増えていく

モノ売りから、コト売りへ

- 継続的な機能追加とアップデート（OTA前提の設計）
- 稼働データを次の価値に還元するサイクル
- このサイクルをAIが加速して、はじめて現実的に回りはじめる



# つまり「価値のつくり方を再定義する」ということ

## どこを固め、どこを伸ばすかを、設計で決め直す

### 設計で決めるべきこと



製品のどこは厳密さ・信頼性・高品質を重視するのか



どの部分に柔軟性を持たせ、ソフトで価値を生み出すのか



価値につながるデータを、どう収集するのか

### 身近な例



テスラ：出荷後の更新で、新機能も航続距離などの性能も伸びる



Apple Watch：心電図アプリが心房細動を検知し、改良され続ける

作り続けるには、組織のほうが先に変わる必要がある

「どこを自前で持つか」を決め直し、戦略と組織の両方を見直す必要がある

## 出荷後も価値を伸ばすなら

製品を出し終えても、チームは  
解散できない

- 10年、20年、同じ土台を更新し続ける体制が要る



## ひとつの事業部では抱えきれない

セキュリティ、長期保守、認証対応は、  
どの製品にも等しくのしかかる

- 同じ下回りを、それぞれが別々に抱えるほどの余力はもうない



難所：コト売りになるほど、モノへの要求は厳しくなる

## ソフトウェア・ディファインドは、モノを軽くする話ではない



### よくある誤解

**上モノが主役になるなら、組込みソフトは軽くなるのではないか**

- 実際は逆。信頼できる下回りやハードがあっこそ、上モノが価値を生む



### 土台への要求

**土台に突きつけられる要求は、むしろ厳しくなる**

- ミリ秒単位のリアルタイム性と、止まらない高信頼性
- SLTS（超長期サポート）10年、20年という保守責任
- CRA（サイバーレジリエンス法）が求めるサイバーレジリエンス

# 3. Framework

## 価値を作る3つの領域とその戦略

今は地図づくり — 競争・協調・共創を、3つの領域として切り分ける

| 目的地                                                                                                                       | よりどころ                                                                                                                 | 進み方                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・デファインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | Collaboration<br><br>●外はOSS、内はインナーソース |

# 領域ごとに、打っ手は変わる

## Unique Innovation

企業が顧客に提供する価値

## 自社の強みを、ここに注ぎ込む

市場で勝ち抜く上で直接効く、絶対領域

## Delivery Support for the Innovation

価値提供に必須の技術

## 社外のOSSで、仲間と創る

差異化には効かないが、勝ち抜くには必須。ここが下回り発展させる要素も多数ある領域

## Commodity Platform

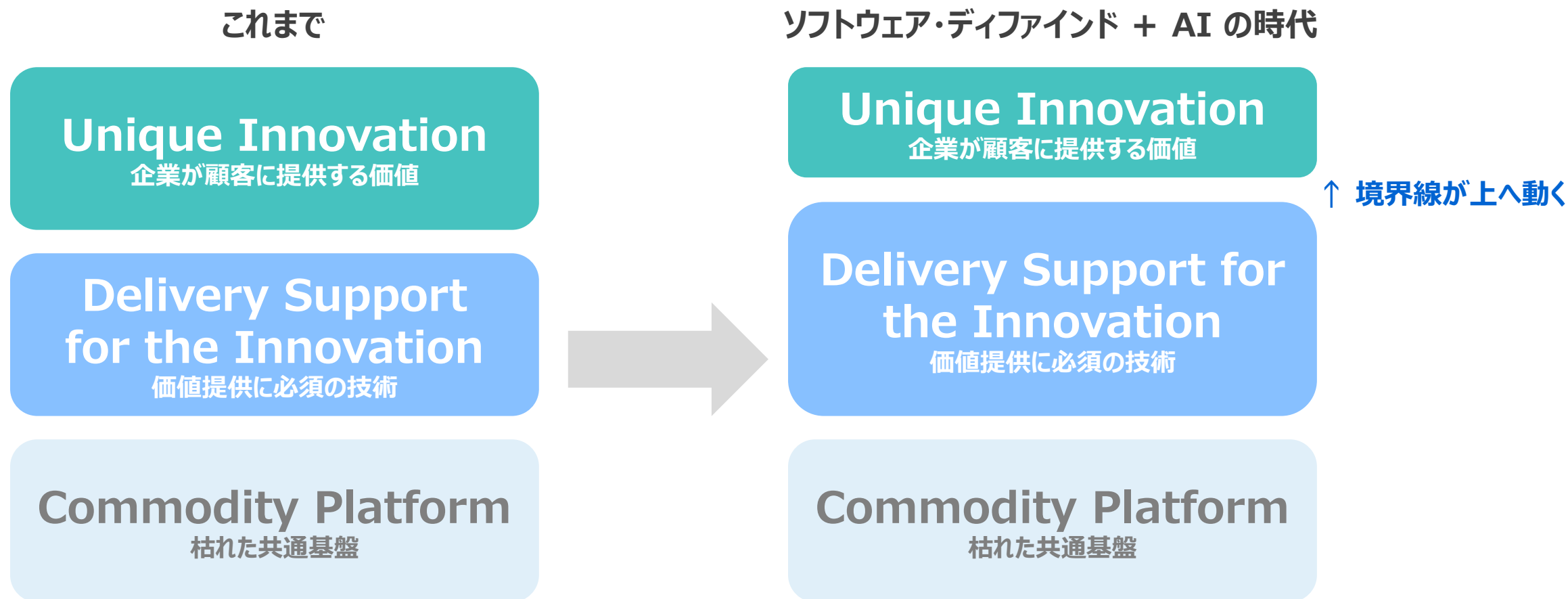
枯れた共通基盤

## 作らない。既にあるものを使う

技術が枯れていて、購入すれば間に合う

ソフトウェア・ディファインドへ向かうほど、真ん中の層が重くなる。  
上物を更新し続けるには、下回りが10年20年動き続けなければならない

# 価値を作る3つの領域と、動き続ける境界線



かつて作り込んでいた領域の共通化が進むと、守るべき絶対領域は薄くなり、支える下回りが厚くなる

## 何が非競争で、何が競争かの決断が難しい 判断の物差し

Q1

顧客が価値を感じて  
いる部分か

Q2

自社の差別化に直結  
しているか

Q3

他社や外部で  
代替できてしまうものか

協調領域

共通課題は、分かち合う

競争領域

ハード制御技術・ドメイン知見・安全設計

共通課題は分かち合い、独自の付加価値でエッジを効かせる

# 立ちはだかる2つの壁 — 内部硬直性と外部硬直性

組織の縦割りと自前主義。この2つの硬直性を打破する鍵が、オープンな開発

## 内部硬直性 — 組織が縦割りで動く



事業部ごとに縦割りで動くため、似たものを重複して作り続けてしまう

- 似た基盤ソフトを、事業部ごとに別々に開発・保守している
- セキュリティ、更新、認証対応といった横断的な課題が、誰の仕事でもない

## 外部硬直性 — 自前主義で社外連携が進まない



社外の力を取り込めず、変化のスピードに追いつけなくなる

- 基盤ソフトを自社だけで維持するコストが、年々積み上がる
- 製品が高機能化するなか、全てを自社開発しては変化に追いつけない

# 4. Collaboration

## 外と中とで異なる戦略

今は「協調」— 共通課題は、外の仲間と一緒に（OSS）

| 目的地                                                                                                                       | よりどころ                                                                                                                 | 進み方                                                                                                                               |
|---------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・デファインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | <b>Collaboration</b><br><br>●外はOSS、内はインナーソース |

# 本章のポイント：外はOSS、内はインナーソース

## 外は世界と共通の土台を支え、内はOSSの作法で縦割りを溶かす

### 外部＝OSSで、非競争領域を世界と作る



競合とも手を組み、共通の土台を一緒に支える

- 産業用途の超長期保守を、競合と共に支える
- それを製品基盤として取り込む

### 内部＝インナーソースで、縦割りを溶かす



社外のやり方を、そのまま社内の開発に持ち込む

- OSSの作法（公開・レビュー・貢献）を社内に持ち込む

# OSSの強み

## 誰でも自由に使える



誰でも自由に使え、変更できる

## 公開されて、進化し続ける



工夫点を公開して受け入れられれば、どんどん進化する

## 関わる人が多いほど、直っていく



多くの開発者が関わり続けるから、不具合が見つかり修正される

閉じた開発のままでは、この速度には追いつけない

# 4-1. 黒歴史からの学び

オープンであるということの本質

| 目的地                                                                                                                        | よりどころ                                                                                                                 | 進み方                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・ディファインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | Collaboration<br><br>●外はOSS、内はインナーソース |

# 東芝のOSS貢献の始まり： CE Linux Forum

最初のきっかけは、パナソニック（当時、松下電器）とソニーが手を組んで、情報家電向けのLinux開発を進めるフォーラムを作ったのが始まり（2003年）



<https://www.celinuxforum.org/>

参加メンバの例

Appointed Member (ステアリング会議での議決権あり)

|     |          |       |         |         |     |    |    |
|-----|----------|-------|---------|---------|-----|----|----|
| IBM | 日立       | シャープ  | Philips | Samsung | NEC | HP | 東芝 |
| LG  | Motorola | NOKIA | 松下      | SONY    |     |    |    |

Associated Member (ステアリング会議での発言権あり)

|     |         |         |        |            |         |         |     |
|-----|---------|---------|--------|------------|---------|---------|-----|
| ST  | TI      | THOMSON | Intel  | Sanyo      | JVC     | 三菱      | ARM |
| CAC | KENWOOD | LINEO   | TimSys | Montavista | Phoenix | Softier |     |

Support Member (活動参加のみ)

# CELPの活動当初の技術課題

- 組み込み機器でLinuxを用いるために解決すべき技術課題
  - System size
  - Boot time
  - Power management
  - Realtime
  - Security
- 手段
  - Linuxの標準機能に取り込む
  - 課題を提案して仲間を集めて活動に繋ぐ
  - Wikiを立ち上げ技術情報を積極的に公開

# CELF/CEWG/CELP20年の歴史

設立

2003/07



CE Linux Forum

The Linux Foundationと合流&改名

2011/01



CE Workgroup

改名

2017/06



Core Embedded Linux Project

PJ終了

2023/07

技術活動

CELF Specification V1.0  
(2004/05)

OSSコミュニティへのコントリビューション

Open Project (2008~2018)

Elinux wiki (2006/12~)

Japan Technical Jamboree (2004/10~)

Embedded Linux Conference (2005/01~)

Episode II

イベント

参画

東芝参画 (2003/7~)

AG / SCメンバ (2003/07~)

SC運営 (2014~)

残務整理

小林  
参画

# CELF黒歴史 (私のバイブル)

- 2004/05 : CELF Embedded Linux Specification を出版

<https://lkml.org/lkml/2004/5/17/149>



**From** Christoph Hellwig <>  
**Subject** Re: ANNOUNCE: CE Linux Forum - Specification draft  
On Mon, May 17, 2004 at 12:05:36PM -0700, Tim Bird wrote:

「あのねえ、こんな仕様書を書けば誰かがタダで実装してくれると思っているの？ だとしたらトンデモない間違いだよ。こんなのが書けるんだったらもう何かコードを書いているんでしょ。それをまずコミュニティに持って来なよ！仕様書なんてどうでも良いから」

# 一方で弊社のCELF参画中の黒歴史：OSSの作者に利用許可を求めた

## NTP（時刻同期のためのソフトウェア）の作者からの返信（2009年6月）

From: "Poul-Henning Kamp"

Subject: Re: Confirmation: your copyright about NTP

To: "\*\*\*\*\*@toshiba.co.jp"

Citat:

Dear NTP Project Members. We are considering using your program "NTP" in our products. Before going any further, however, we would like to confirm the following so that we are sure to fully respect your rights. You are of the copyrights in the program. You have distributed the program und or without modification provided that any cond confirming the above, that would be most appri program. Thank you in advance for your kindn

Let me get this straight: You, a major worldwide m reference implementation on the internet, containir trust the contents of the license file to be legally bir with before, to people you have no way of authenti return mail you may or may not get, will cover you a result of you using the NTP reference implementa competent legal assistance. You could have done w who trust the license file and happily uses the NTP researching this point, you sent this pointless email license status of the code I may, or may not, have me, you will have to send me a registered letter, m the questions you want answered, and a suitable to sensible monetary donation to the NTP project would work, as long as I do not need to ask my lawyer to answer your questions. Until such time, I will place you in legal jeopardy with respect to your use of the NTP reference implementation, with my statement above, and my signature below; sign: Somebody who may or may not be, Poul-Henning Kamp Who may or may not have written parts of the NTP reference implemenation, which may or may not be covered by the license restrictions you quoted.

「ちょっと確認させてよ、ライセンスって知ってる？あなたたちのような弁護士何人も抱えている会社は何言っているの？IBMもCiscoも理解して使ってるよ。」

Ref: <http://www.version2.dk/blog/i-am-not-nice-man-11355>

- コントリビューションあってこそ、コミュニティは活性化する
- コミュニティを尊重する
- 何かを得るには自ら動く
- 早めに聞いて、早くフィードバックをもらう



理想は言えども（当時の）日本の技術者は外に出ていかない問題

# 海外のカンファレンスでの発表（2010年～2020年）

- 1.Evaluation of Data Reliability on Linux File Systems (ELC 2010, Apr. 2010)
- 2.Linux Kernel Acceleration for Long-term Testing (ELC-E, Oct. 2010)
- 3.Moving Forward: Overcoming Compatibility Issues (ELC, Apr. 2011)
- 4.Automated Regression Test Environment For Multiple Kernels (LinuxCon EU, Oct. 2011)
- 5.Ineffective and effective way to find out latency bottlenecks by Ftrace (ELC, Feb. 2012)
- 6.Improvement of Scheduling Granularity for Deadline Scheduler (ELC-E, Oct. 2012)
- 7.Deadline Miss Detection with SCHED\_DEADLINE (ELC, Feb. 2013)
- 8.An Essay on the Design of the Linux Kernel (ELC, Apr. 2013)
- 9.Expectations for the Future of Linux (ELC, Apr. 2013)
- 10.Using the Linux Kernel to Improve Performance (ELC, Apr. 2013)
- 11.Collaboration between the Linux Kernel and the Hardware (ELC, Apr. 2013)
- 12.Using the Linux Kernel to Improve Performance (ELC, Apr. 2013)
- 13.Poky: A Linux-based Embedded Linux Distribution (ELC, Apr. 2013)
- 14.BoFS: A File System for Embedded Linux (ELC, Apr. 2013)
- 15.Apply the Linux Kernel to Improve Performance (ELC, Apr. 2013)
- 16.CE W: A Linux-based Embedded Linux Distribution (ELC, Apr. 2013)
- 17.Introduction to the Linux Kernel (ELC, Apr. 2013)
- 18.The Linux Kernel: A History of the Linux Kernel (ELC, Apr. 2013)
- 19.Introduction to the Linux Kernel (ELC, Apr. 2013)
- 20.Time to Move Forward: Overcoming Compatibility Issues (ELC, Apr. 2013)
- 21.Civil Infrastructure Platform: Industrial Grade SLTS Kernel and Base-layer Development (Open Source Summit Japan 2017, Oct. 2017)
- 22.Debian: A Linux-based Embedded Linux Distribution (ELC, Apr. 2013)
- 23.Civil Infrastructure Platform: Two Year Experience of Industrial Grade Open Source Base-Layer Development (Open Source Summit Japan 2018, Oct. 2018)
- 24.Civil Infrastructure Platform: Industrial Grade SLTS Kernel and Base-layer Development (Open Source Summit Japan 2018, Oct. 2018)
- 25.Civilization runs on Debian (DebConf 2018, July 2018)
- 26.Civil Infrastructure Platform: Two Year Experience of Industrial Grade Open Source Base-Layer Development (Open Source Summit Japan 2018, Oct. 2018)
- 27.How to Make Smart Cities Stay Smart with Open Source Projects (Smart City Event 2019, January 31, 2019)
- 28.Open Source Projects to Live Long and Prosper: Linux for Smart Infrastructure and Industry (Open Source Summit Europe 2019, October 2019)
- 29.State of CIP Project (CIP mini-summit Europe 2019, October 2019)
- 30.CIP Installed: Sustainable Software Stacks in Long-living Products (Open Source Summit North America, June 29, 2020)
- 31.State of Civil Infrastructure Platform (CIP mini-summit Europe 2020, October 2020)
- 32.The InnerSource Learning Path (InnerSource Commons APAC 2020 Virtual Summit, December 2020)

年2～3回は何かしらの発表をした！  
実装も公開した！

信頼性・性能

社会インフラ

OSSプロジェクト

# CELPの活動 (イベント : Japan Technical Jamboree)



## • 日本の組込み開発者の持つアイデアや問題意識を コミュニティに届けるきっかけ作り

- 2004/10 から 毎年約4回のペースで開催
- 開催回数は 80回 (2023/5)
- 企業・大学等から開発者が結集
  - ベテランはもとより初心者も参加
- Linuxコミュニティには  
ばたいた技術や開発者多数



# CELPの活動 (イベント : Embedded Linux Conference)

HOP

Jamboree

STEP

ELC US/EUR

JUMP

Community

## • 組み込みLinuxに焦点を絞った開発者会議

- 春に米国、秋に欧州で開催
- 毎回500人を超える参加者
- 50を超える技術セッション
- Technical Showcaseでデモを展示

Core Embedded  
Linux Project

## CELPの活動スコアカード（技術開発）～2014年まで

- 組み込み機器でLinuxを用いるために解決すべき技術課題
  - System size - done
  - Boot time - done
  - Power management - done
  - Realtime - in progress → done (2024年11月)
  - Security - in progress
- 手段
  - Linuxの標準機能に取り込む
  - 課題を提案して仲間を集めて活動に繋ぐ
  - Wikiを立ち上げ技術情報を積極的に公開



## 個人的CELP活動まとめ

- 課題はそれなりにあった……
- CELPが関わった**技術は多くの製品に搭載**
- **CELPとしても個人としてもOSSコミュニティと繋がり  
を構築**
  - 意識せずとも頼れる背後霊ネットワーク
  - （何人かは守護霊になってくれたのだろうか……）



オープンに活動したからこそできた頼り頼られる人脈が宝

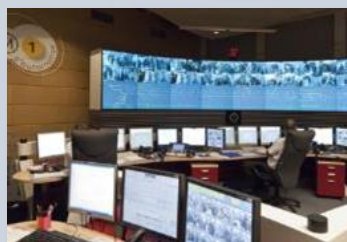
# 4-2. オープンな活動からの出会い

## 社会課題の解決に向けたオープン戦略

| 目的地                                                                                                                        | よりどころ                                                                                                                 | 進み方                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・デフィアインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | Collaboration<br><br>●外はOSS、内はインナーソース |

# Linuxは既に社会インフラの一部として欠かせない存在

## 交通



運行管理



車両制御



自動改札機

## エネルギー



発電所監視制御



タービン制御(I/O)



タービン制御

## 他



ビル管理



放送



ヘルスケア

## 産業機器



産業オートメーション



CNC



産業ネットワーク



CIVIL  
INFRASTRUCTURE  
PLATFORM  
INDUSTRIAL GRADE LINUX

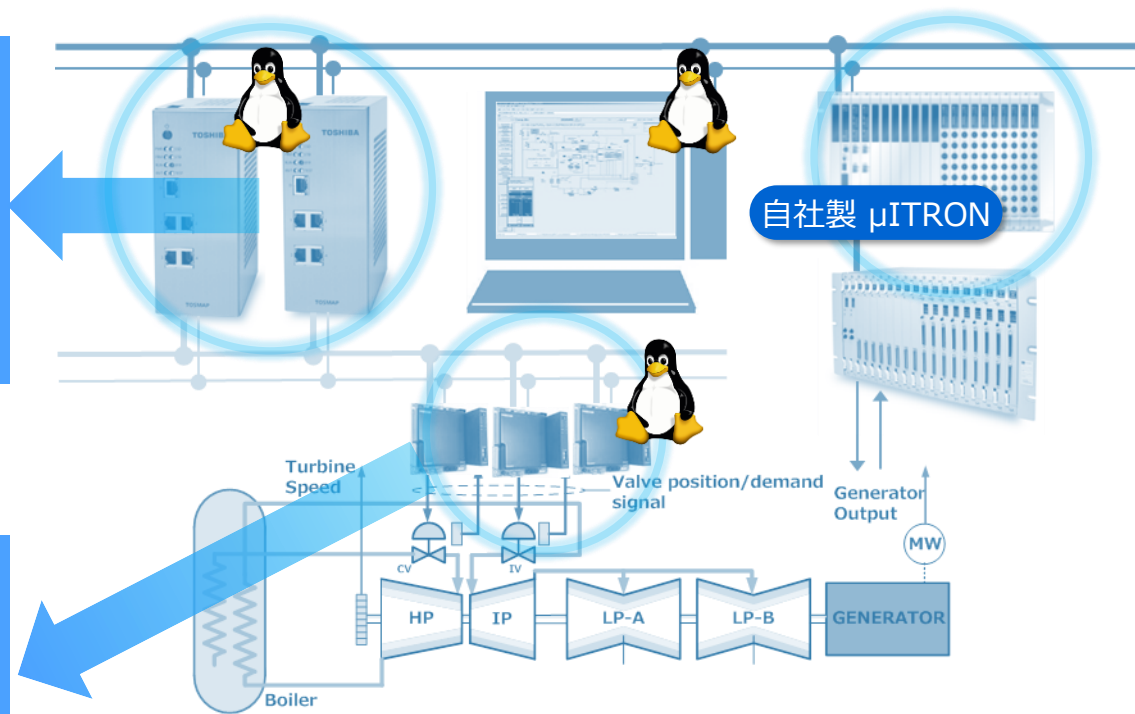
## 事例：発電所向けコントローラ

# Linux®/OSSが発電所のような社会を支えるシステムで利用されている

# DCSコントローラ

- ・ボイラーの制御
- ・プラント全体のモニタリング
- ・タービンの起動と停止

## 100ミリ秒以上で制御。



## タービン向け コントロールモジュール

- ・バルブ制御
- ・周波数検出
- ・速度検出

## 5～30ミリ秒で制御。

Ref: Power Plants run on Linux CIP, CIP mini-Summit 2019

## 目指したい姿

- IoTを社会インフラシステムにも適用する
- 製品品質と寿命を確実なものにする
- 何万台ものつながったシステムのセキュリティを保つ



## 解決すべき課題

### 産業レベル 品質

- 信頼性
- 機能安全
- リアルタイム性

### 持続可能性

- 長期製品ライフサイクル
- 互換性維持
- 標準化対応

### セキュリティ

- セキュリティ、脆弱性管理
- ファームウェア更新
- リグレッションリスク最小化

# 課題をオープンにしたら、同じ提案が別のところから出ていた！

- CEWG内で開始した社会インフラ向けLinux PJ (2014/9)
- Siemens から Open Industry Platform の提案 (2014/12)



## 2015/2 F2Fで打合せ

**LINUX FOUNDATION**  
COLLABORATIVE PROJECTS

Open Industry  
Formation

1

Platform

Communication

ALLSEEN  
ALLIANCE  
hosted by LINUX  
FOUNDATION

OPEN INTERCONNECT  
CONSORTIUM

Industrial Internet  
CONSORTIUM

PI  
PROFINET - PROFINET

Consumer

Industry

**LINUX FOUNDATION**  
COLLABORATIVE PROJECTS

Open Industry Alliance (OIA)

This document describes the "Standard Distribution for Embedded Systems" for the CE Workgroup.

**Description**

This project consists of helping Linux Foundation member companies (particularly those associated with the CE Workgroup) to share a Standard Distribution for Embedded Systems (SDES).

below:

for the CE Workgroup.

particularly those associated  
structure systems.

for variety of social infrastructure  
the system is a control system which runs

has systems. In this document, the definition  
on the following systems. (See also appendix B for detail.)

- Transport infrastructure
- Energy infrastructure
- Water management infrastructure
- Communications infrastructure
- Earth monitoring and measurement systems

This definition might also cover a part of IoT related technologies (See also appendix B). However this project focuses on infrastructure systems for IoT.

Social infrastructure systems are required to have significantly high quality. There are many requirements for the infrastructure systems as shown below:

# 2016年4月 Civil Infrastructure Platform (CIP) 始動！



**CIVIL**  
**INFRASTRUCTURE**  
**PLATFORM**

INDUSTRIAL GRADE LINUX

## Platinum Members

**RENESAS**

**SIEMENS**

**TOSHIBA**

## Gold Members

 **TEXAS INSTRUMENTS**

## Silver Members

  
cybertrust

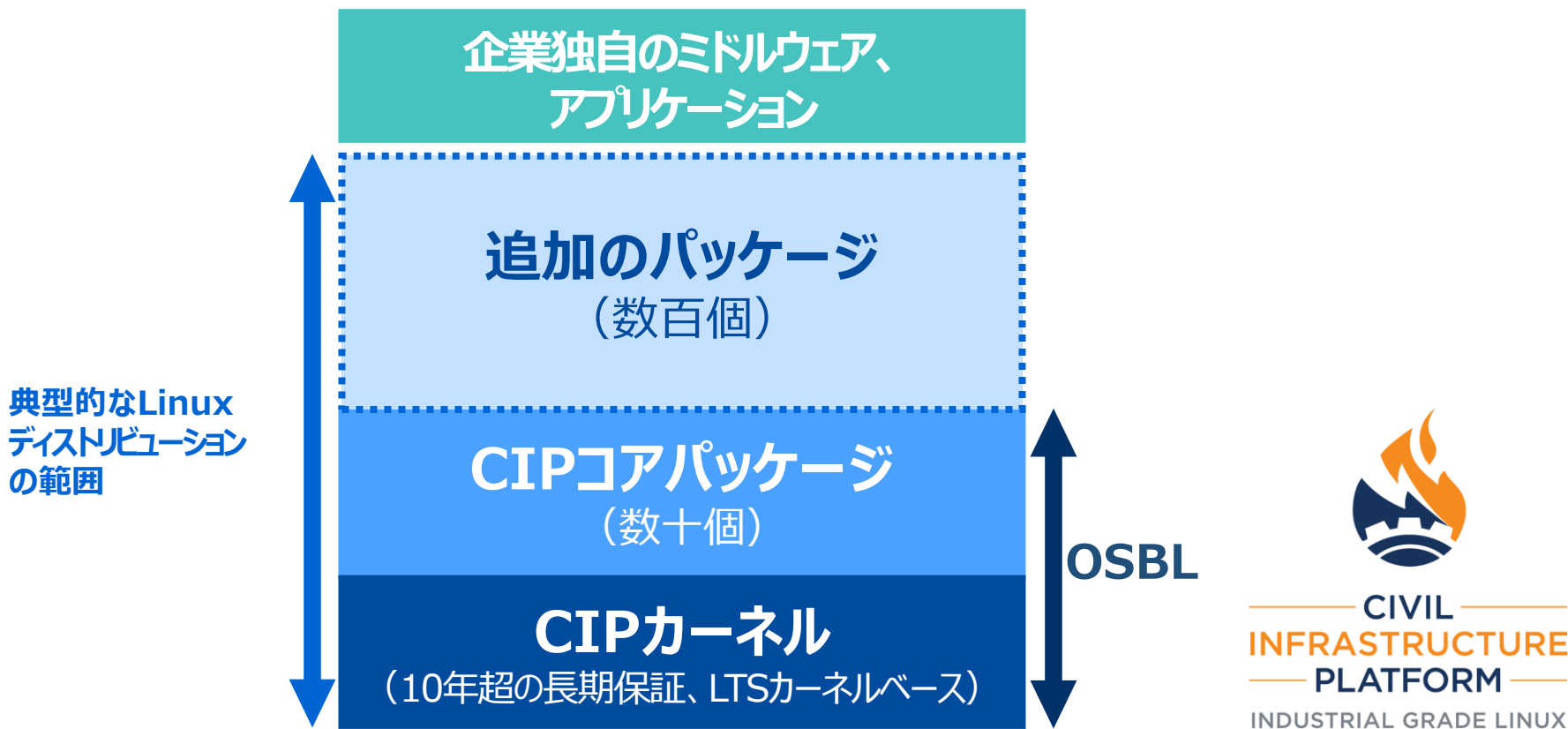
**HITACHI**

 **LINUTRONIX**  
LINUX FOR INDUSTRY

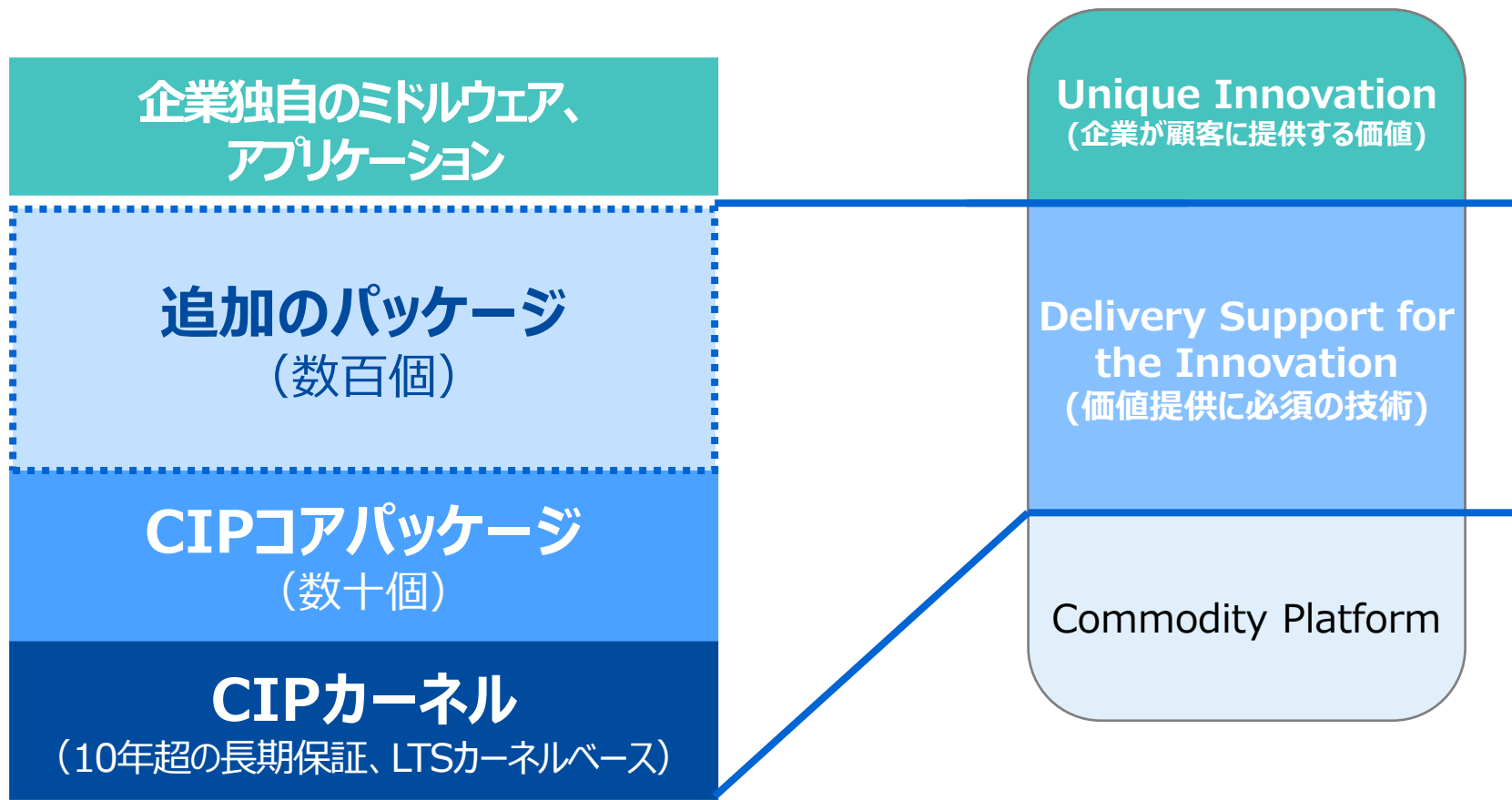
**MOXA**

## 課題解決に向けた姿：CIPが目指す“Open Source Base Layer (OSBL)”

- 社会インフラのような高い信頼性や長期保守が求められる製品向けの共通OSS集
- Linuxディストリビューションや関連するOSSと共に利用

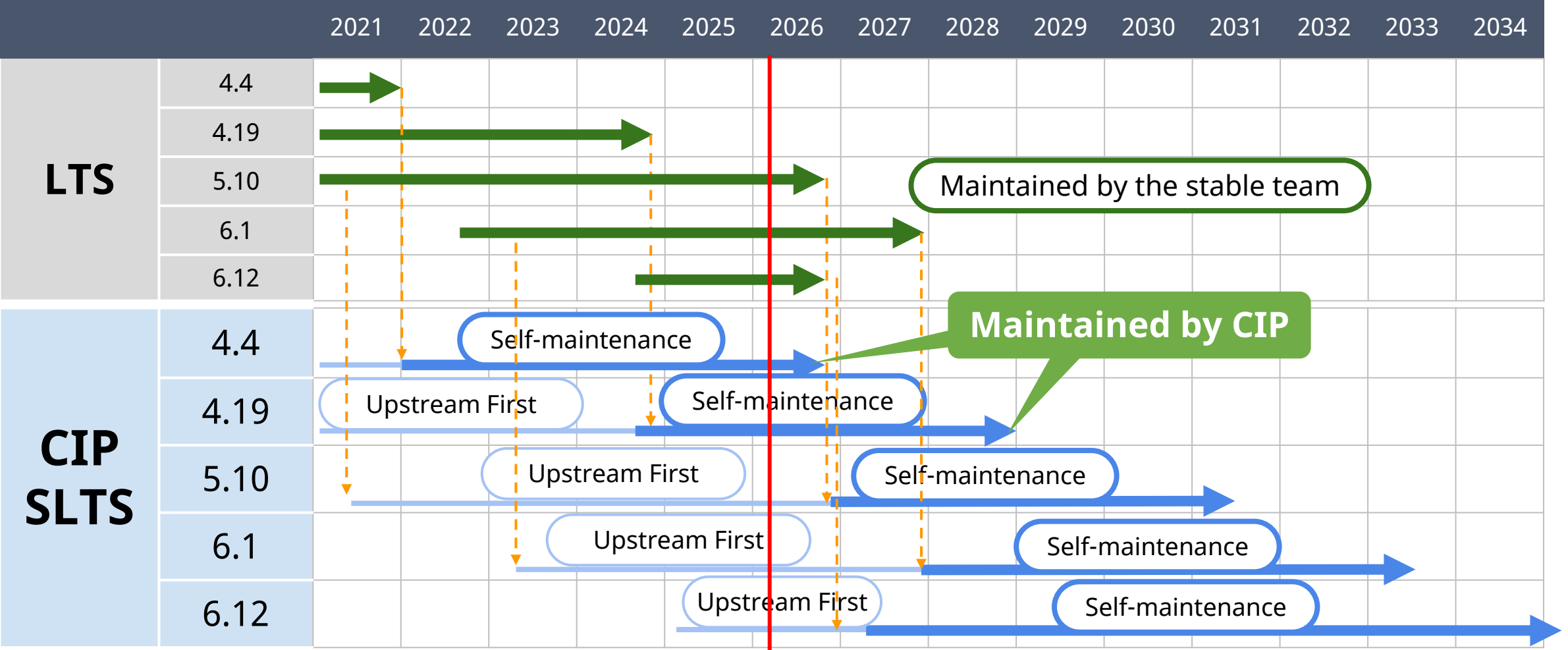


# 価値を作る3つの領域とCIP OSBL



**Ref:** How to Contribute to the Linux Kernel, and Why it Makes Economic Sense  
(James Bottomley, Novell; LinuxCon Japan 2009)

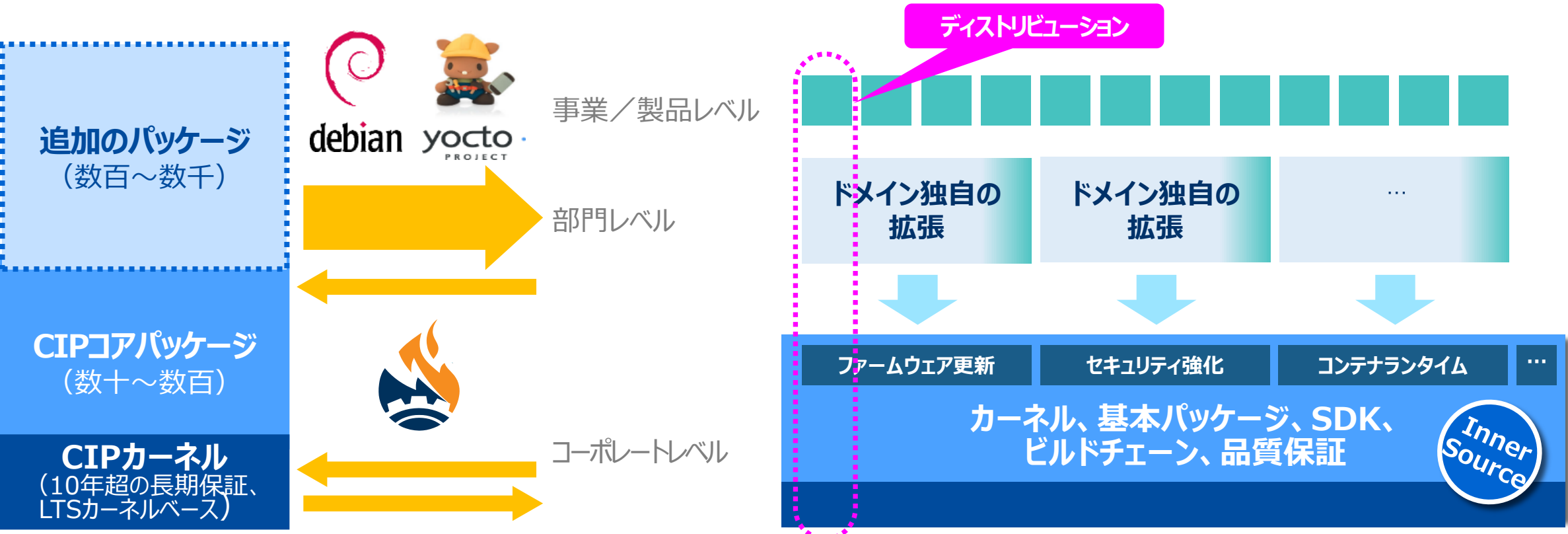
# 最低10年のメンテナンスマイルストーン



We are here

# CIPを企業に適用する

レイヤー構造を持つインダストリアル製品向けLinuxディストリビューションを  
CIPのOSBLを用いて作成



個々の製品に対する、ライセンス確認、セキュリティ問題チェック、ソフトウェア保守、アプリケーション対応・テストなどで  
**最大70%もの工数削減が可能**

# 数字で見るCIPの活動

## CONTRIBUTIONS

**1000+/month**

LTS Patch  
reviews

## CVE CHECKS

**2000+ fixes**

Counts only for  
6.1.y-cip  
in 2024

## SLTS RELEASES

**582 releases**

August 2026  
v4.4(114, rt62)  
v4.19(135,rt49)  
v5.10(76, rt26)  
v6.1(59, rt31)  
v6.12(30)

## REFERENCE HW

**11 boards**

4 architectures  
X86\_64  
Armhf  
Arm64  
Risc V

## TEAM

**5 Key Developers**

Jan Kiszka  
Pavel Machek  
Nobuhiro Iwamatsu  
Ulrich Hecht  
Masami Ichikawa  
(and more contributors.)

# EU Cyber Resilience Act に向けたCIPの取り組み

LF Research発行のレポートにも載っています！

## Reference:

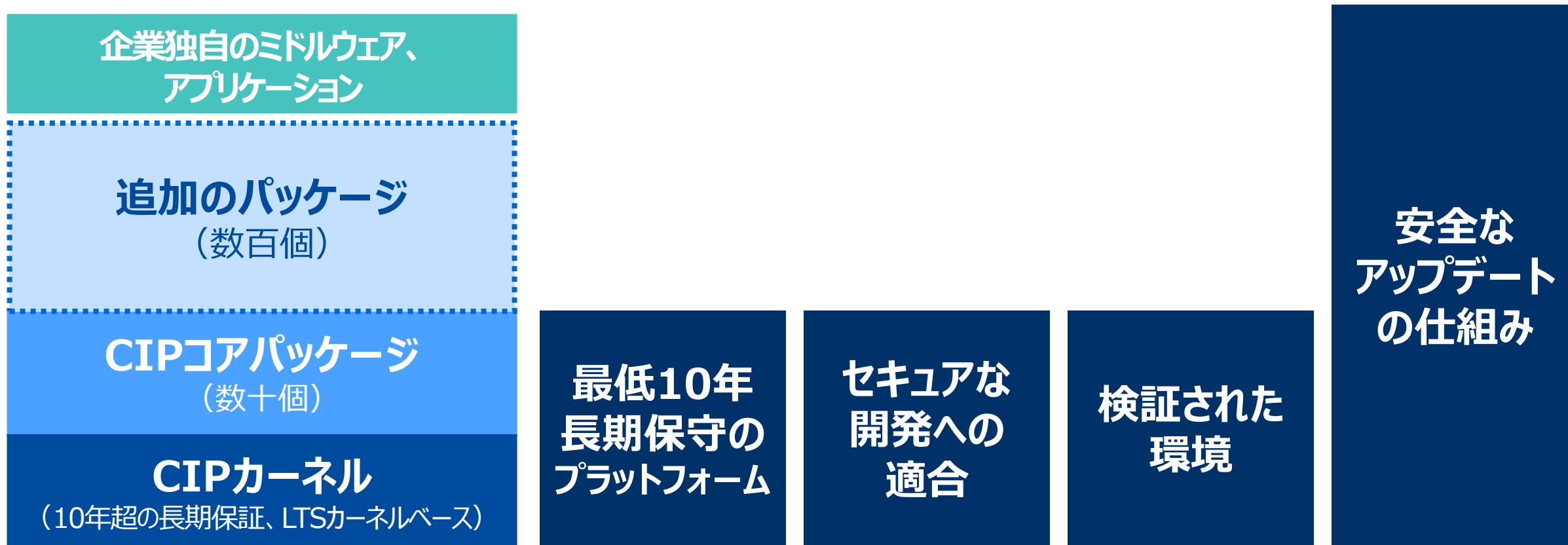
Mirko Boehm, Hilary Carter, and Cailean Osborne,  
"Pathways to Cybersecurity Best Practices in Open Source:  
How the Civil Infrastructure Platform, Yocto Project, and Zephyr Project  
are Closing the Gap to Meeting the Requirements of the Cyber  
Resilience Act,"  
foreword by Miriam Seyffarth, The Linux Foundation, March 2025.

<https://www.linuxfoundation.org/research/cra-compliance-best-practices>



CIP が EU CRA に向けたベストプラクティスの一つとして紹介

# CIP Open Source Base Layer の現在



当初目的を達成しつつ、プロジェクトの長期持続性確保にむけて活動継続

# オープンにしたからこそ生まれた「思わぬ出会い」

## 外部硬直性を打破すると、思わぬ出会いが生まれる

### 閉じたまま進めていたら、どうなっていたか



自社のなかだけで、答えを探し続けることになる

- 同じ課題を、別々の場所で二重に解き続けていた
- 相手がいることにさえ、気づかないままだった

### オープンにしたから起きたこと



同じ方向を向く相手と、外でつながれるようになる

- 同じ志を持つ仲間が見つかり、一緒に立ち上げる相手になった
- 1社では届かない規模と期間の活動が、成立した

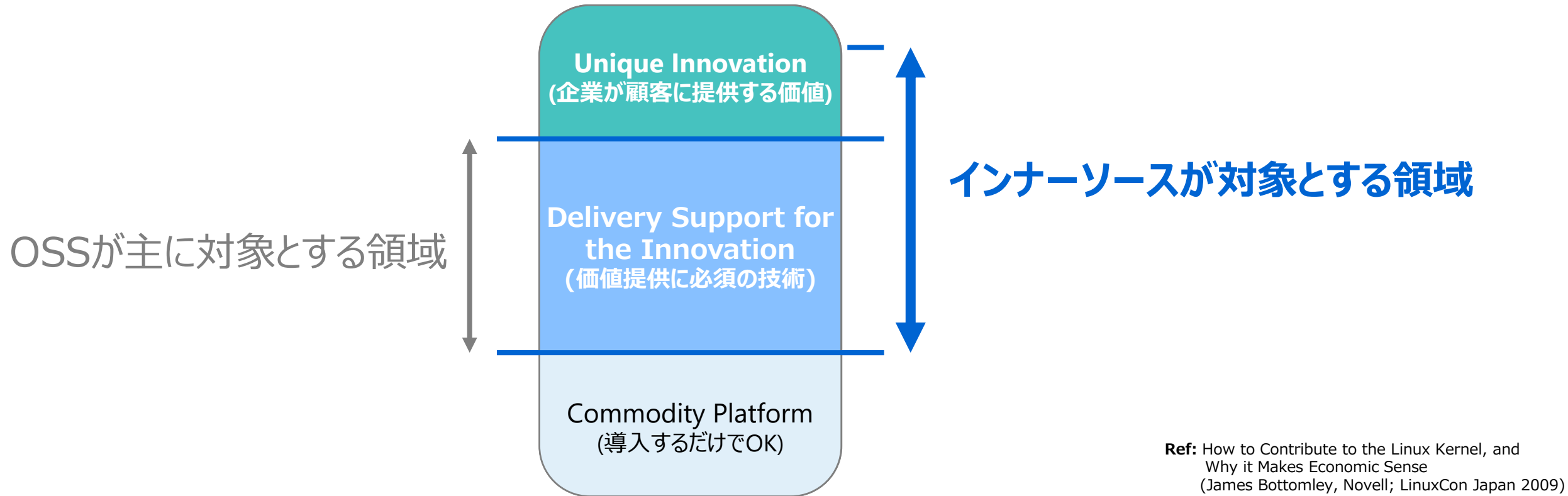
# 4-3. 今こそ社内もオープン化

## 今だからこそ始めるインナーソース

今は「共創」— 社内に開き、競争領域まで一緒に速くする（インナーソース）

| 目的地                                                                                                                       | よりどころ                                                                                                                 | 進み方                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・デファインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | Collaboration<br><br>●外はOSS、内はインナーソース |

# 「共通課題」はどこまで広げられるか（OSSとインナーソースの守備範囲）



**会社内なら外に出せない「共通課題」も解決できる！**

## こんな経験はありませんか？

- 一緒にやってるプロジェクトで他チームを待っていたら遅れた・・・
- バグを見つけて、通知したのに何時まで経っても直ってこない・・・
- 似たような機能を作ってしまい、両方メンテナンスしないといけない・・・

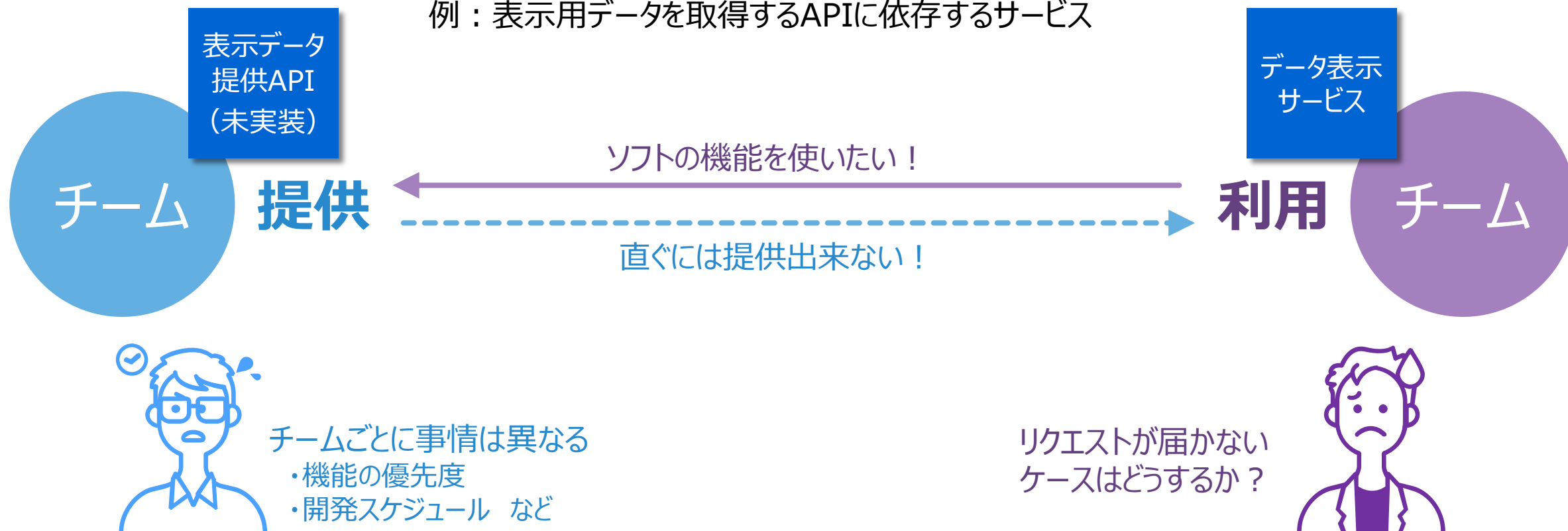


# こんな経験はありませんか？

## ケース

同じ会社にある二つのチームが、別々のソフトウェア部品を提供する時、  
片方のチームのソフトウェアが、もう一方のチームのソフトウェアに依存する状況

例：表示用データを取得するAPIに依存するサービス



# こんな時、あなたならどうしますか？

## 1. 「静観」：黙っている

|      |                 |       |                          |
|------|-----------------|-------|--------------------------|
| メリット | 作業を最小限にすることができる | デメリット | ・要求された機能がいつまでたっても提供されない。 |
|------|-----------------|-------|--------------------------|

## 2. 「回避」：勝手にやる

|      |                               |       |                                                                                         |
|------|-------------------------------|-------|-----------------------------------------------------------------------------------------|
| メリット | 要求機能が足りない部分をローカルに変更・機能追加して補える | デメリット | ・要求された機能がいつまでたっても提供されない<br>・変更した部分の長期的にメンテナンスしなければならない<br>・変更に対する重複したプロジェクトとコードを取得してしまう |
|------|-------------------------------|-------|-----------------------------------------------------------------------------------------|

## 3. 「圧力」：強いてやらせる

|      |                        |       |                                                                           |
|------|------------------------|-------|---------------------------------------------------------------------------|
| メリット | ・強いてやらせる（手に入る（かもしれない）） | デメリット | ・圧力という開発に関係のない作業に注力しなければならない。<br>・何度も使えるものでもなく発展しない。<br>・チーム間や個人間の信頼を損なう。 |
|------|------------------------|-------|---------------------------------------------------------------------------|

どれを選んでも、誰かが損をする

# インナーソース なら、まとめて解決できます！

## インナーソース で**全ての欠点を解決して効果を得よう！**

- エンジニア同士が新しいさまざまな技術やバラエティに富んだ人々と一緒に仕事をする事ができる
- 共通の課題に対する解決方法を再利用することで、組織に高い価値を生み出すことに集中できる

**DevEx(\*)向上**

**開発効率向上  
コスト削減**

… でも、そもそも **インナーソース** って？

(\*) Developer Experience

# インナーソースとは？

- インナーソース は、**企業内のソフトウェア開発にオープンソースの実践とベストプラクティスと原則を適用**したものです。
- インナーソース ソフトウェアは、会社としてはプロプライエタリなものとなりますが、**内部にはオープンで、誰もが利用したり貢献したりできる**ようになります。

つまり … **Open all artifacts!**  
**Welcome visitors!**

… を企業内で実践することです。

Dirk Riehle, Inner Source (Software Development), InnerSource Summit 2018

# 野望：こんな世界を会社の中（日本中の企業の至るところ）に作りたい！

## 今まで



サイロごとに閉じている  
何をしているか自部門以外のことは知らない

## 目指す姿

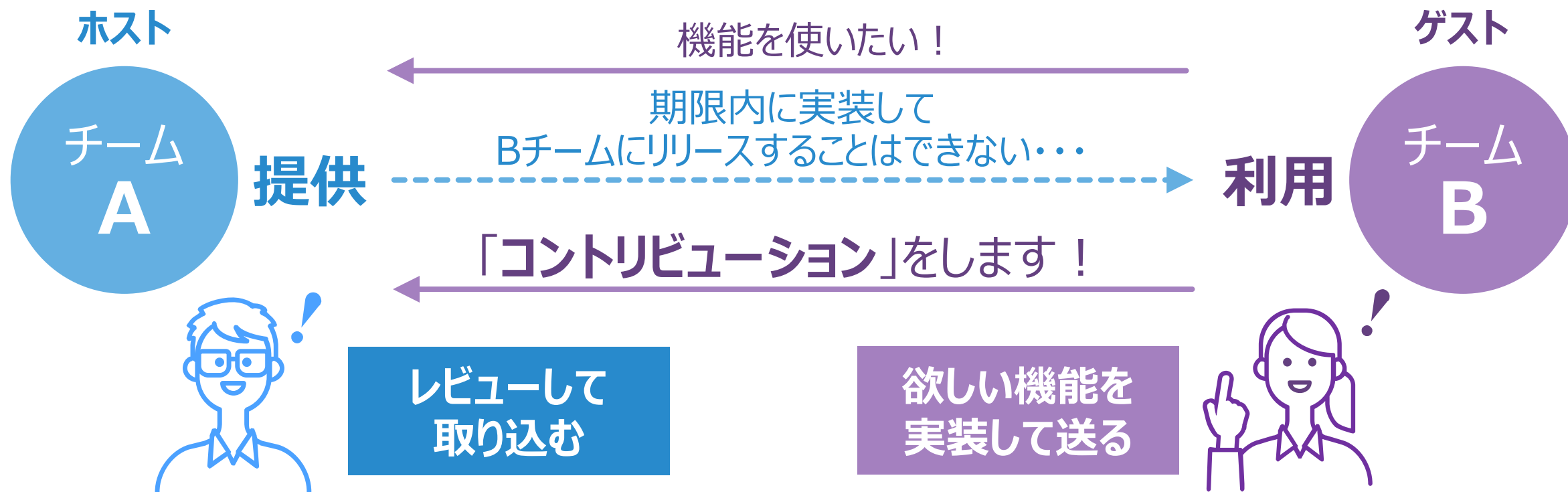


サイロを取り払い、コラボレーションが各所で発生！

# どのように インナーソース は機能するのか？

## ケース

Aチームが提供するソフトウェアをBチームが利用する場合



依頼して待つ関係から、一緒に作る関係へ

# インナーソース の効果とは？

## 開発の利点

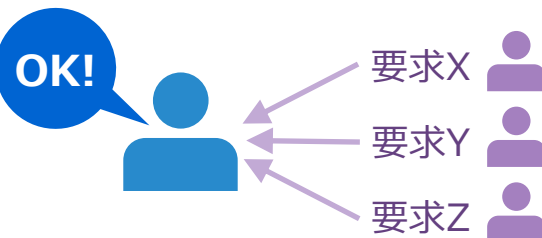
### チーム A〔ホスト〕

- ・ニーズが確定している機能を受け取れるので、良いプロダクトを作るための支援 となる
- ・スケーラブルな戦略ができるようになる

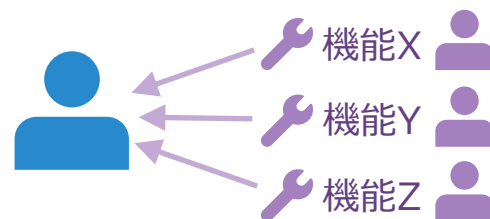
必要な機能が手に入る

品質向上

- ・必要とする時と場所に  
エンジニアリングの時間を  
有機的に投入することが可能



- ・全ての利用者との間で優先順位の調整を行うことができる



### チーム B〔ゲスト〕

長期メンテナンスの負担をせず、  
必要な時に機能要求を手に入れられる

必要な機能が手に入る

戦力倍増する

会社

リソース配分を最適化できる

余力ができ、他へ注力できる

# インナーソースとは？結局何がよくなる？何をよくしたい？

インナーソース = オープンソース開発の「**方法(考え方)**」を企業組織に適用する取り組み

## インナーソース実践

## 経営的な成果の例 (Outcome)

コードの再利用

開発コスト削減・リードタイム短縮

オープンなレビュー文化

品質・セキュリティ向上

ナレッジの共有

組織学習の促進・人材育成・リードタイム短縮・品質向上・・・

サイロの打破

横断的協業 × 部門連携強化

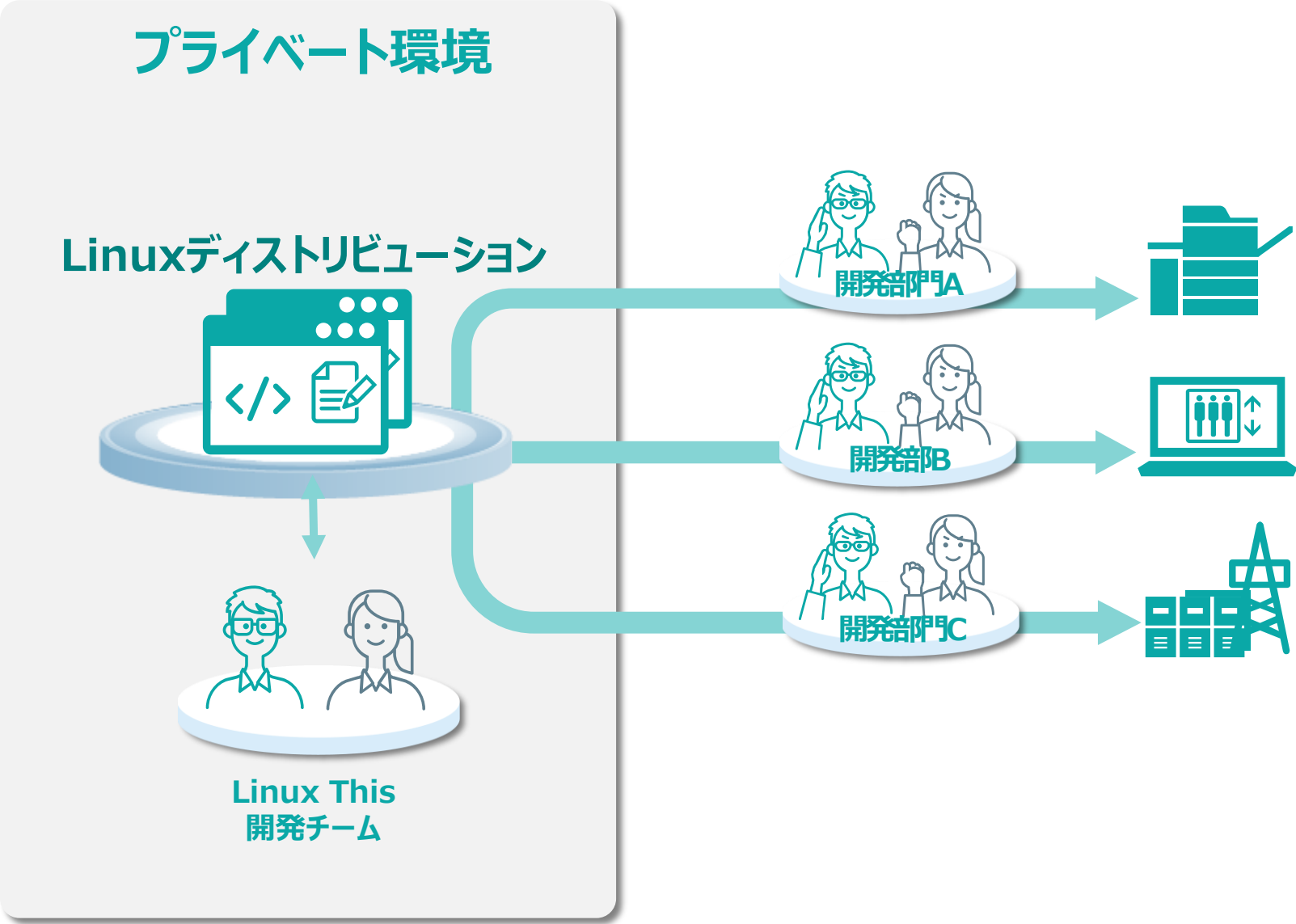
インナーソースは単なる「ソースコードの共有」に留まらず、  
「知」の共有という「考え方」を用いて開発や組織によりよい「成果」をもたらすことが目的

# 実際のインナーソースはどのように機能する？

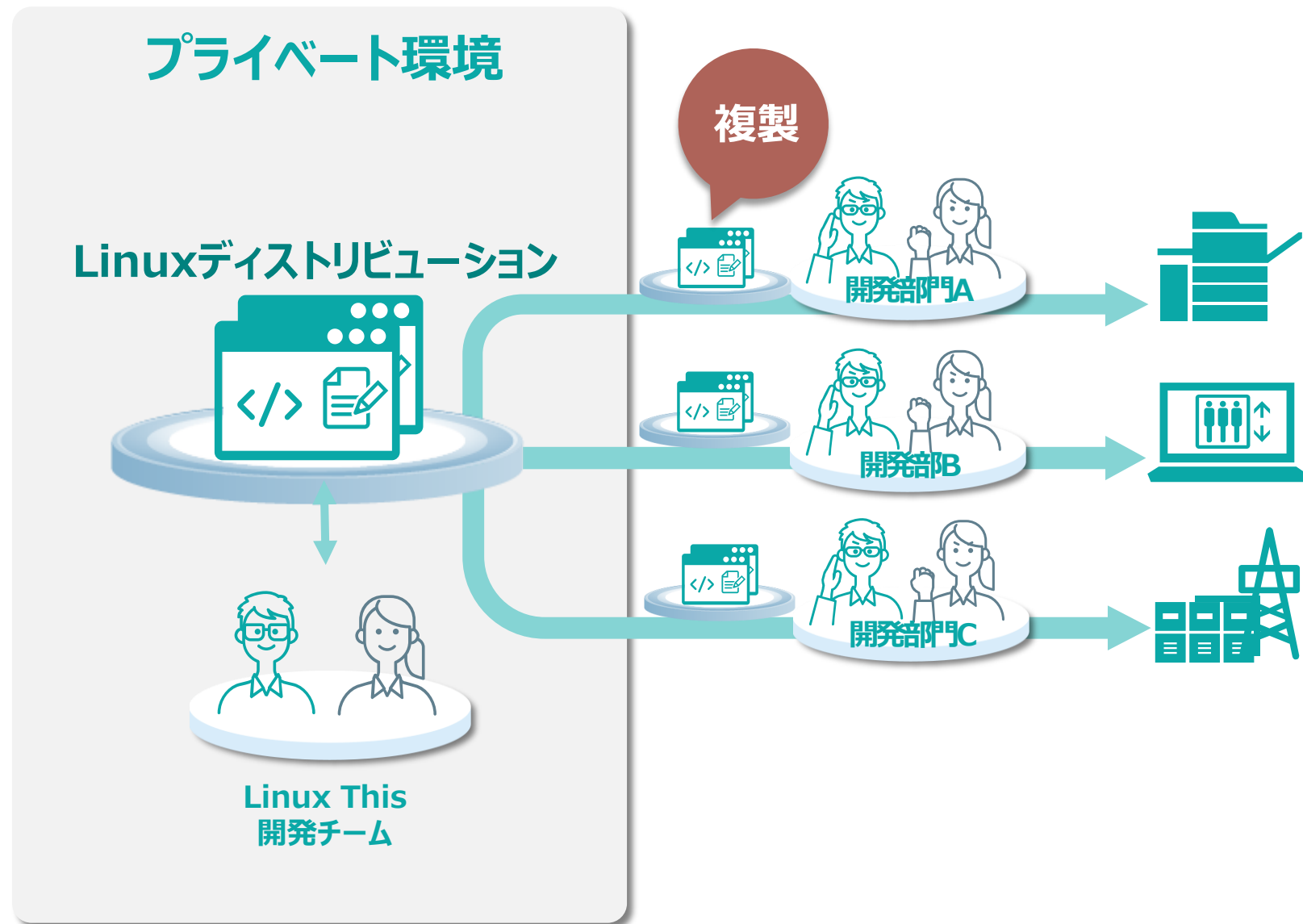


# インナーソース ケーススタディ

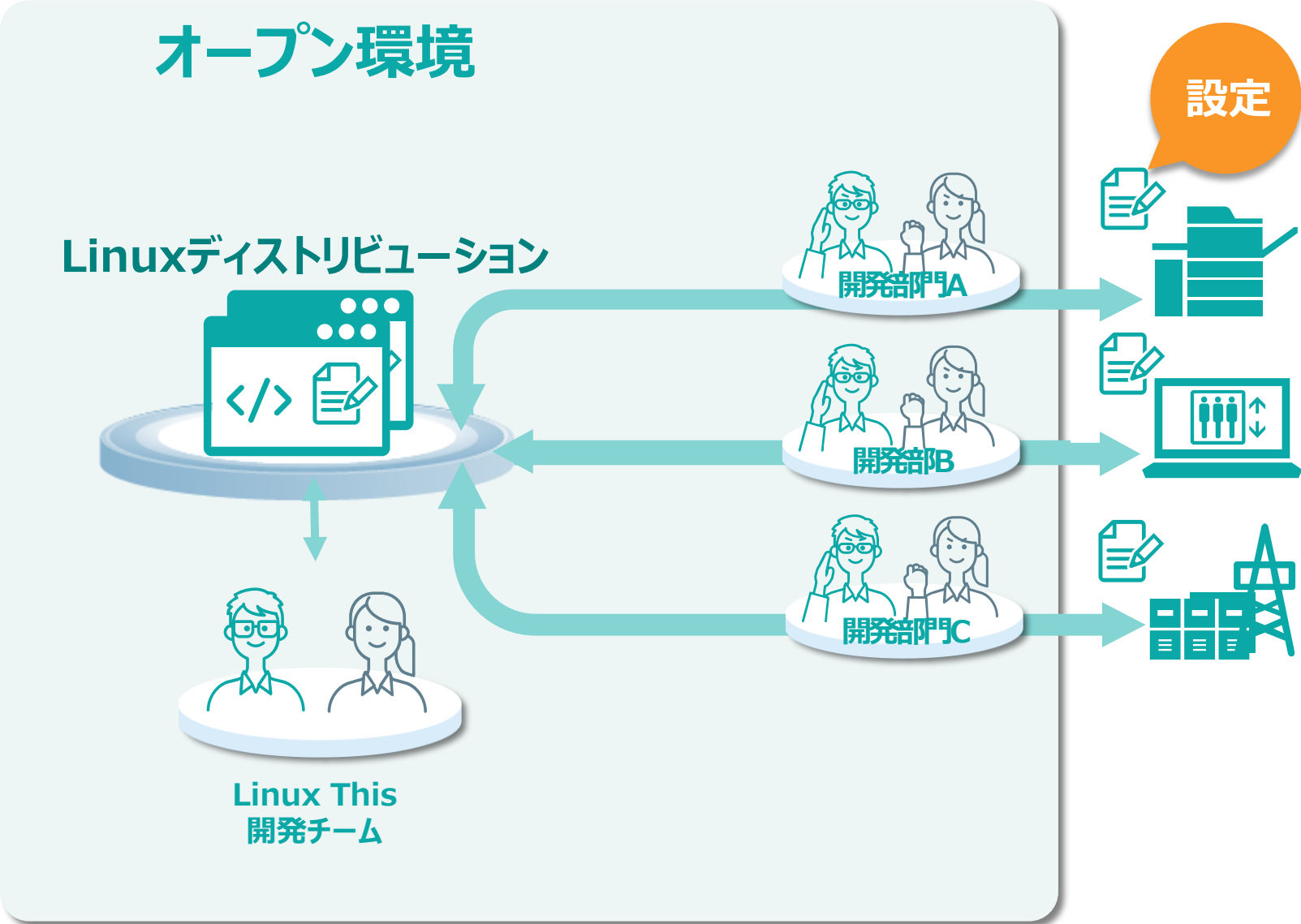
# ケース#1:全社向け共通Linuxディストリビューション



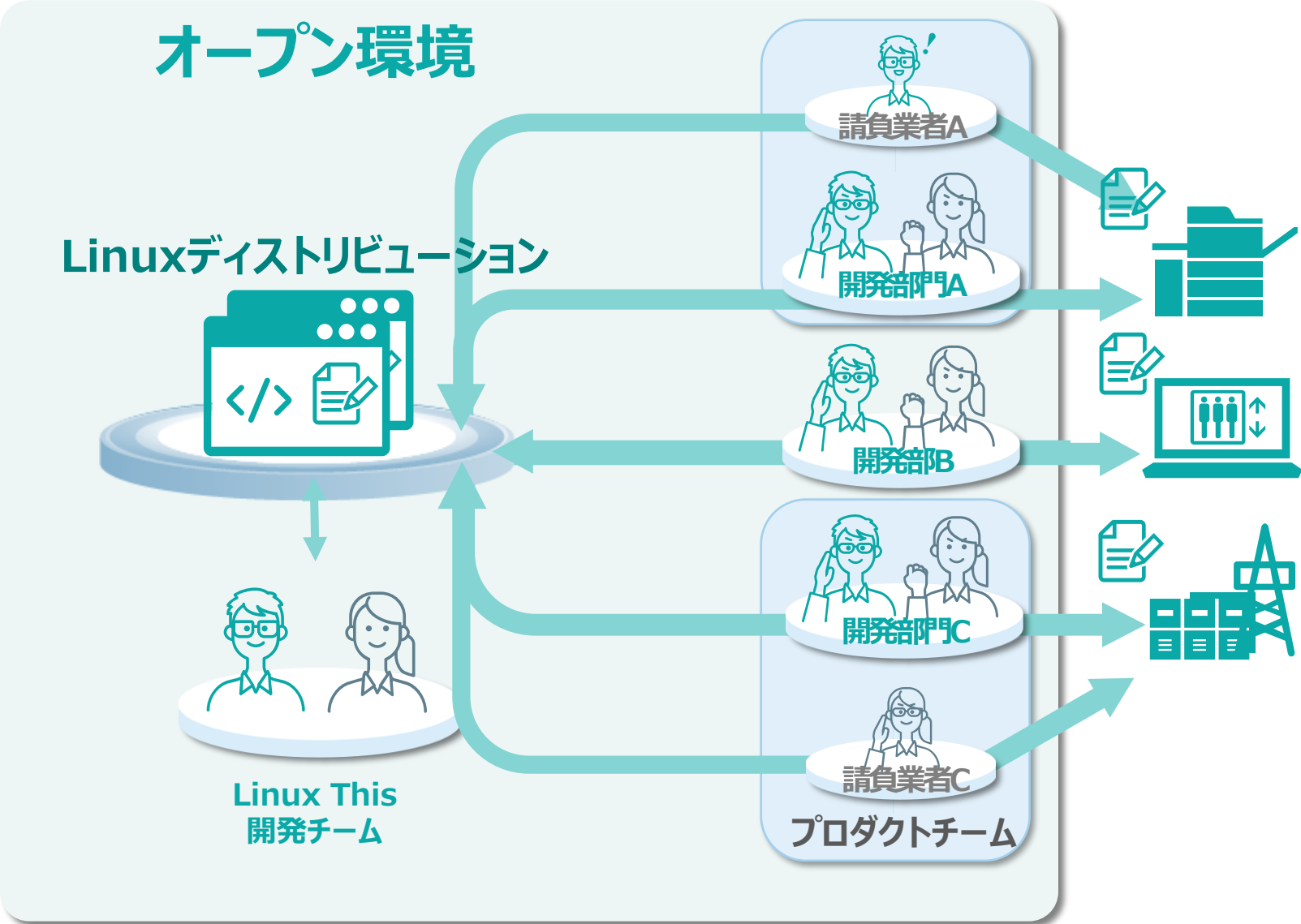
# ケース#1: 全社向け共通Linuxディストリビューション



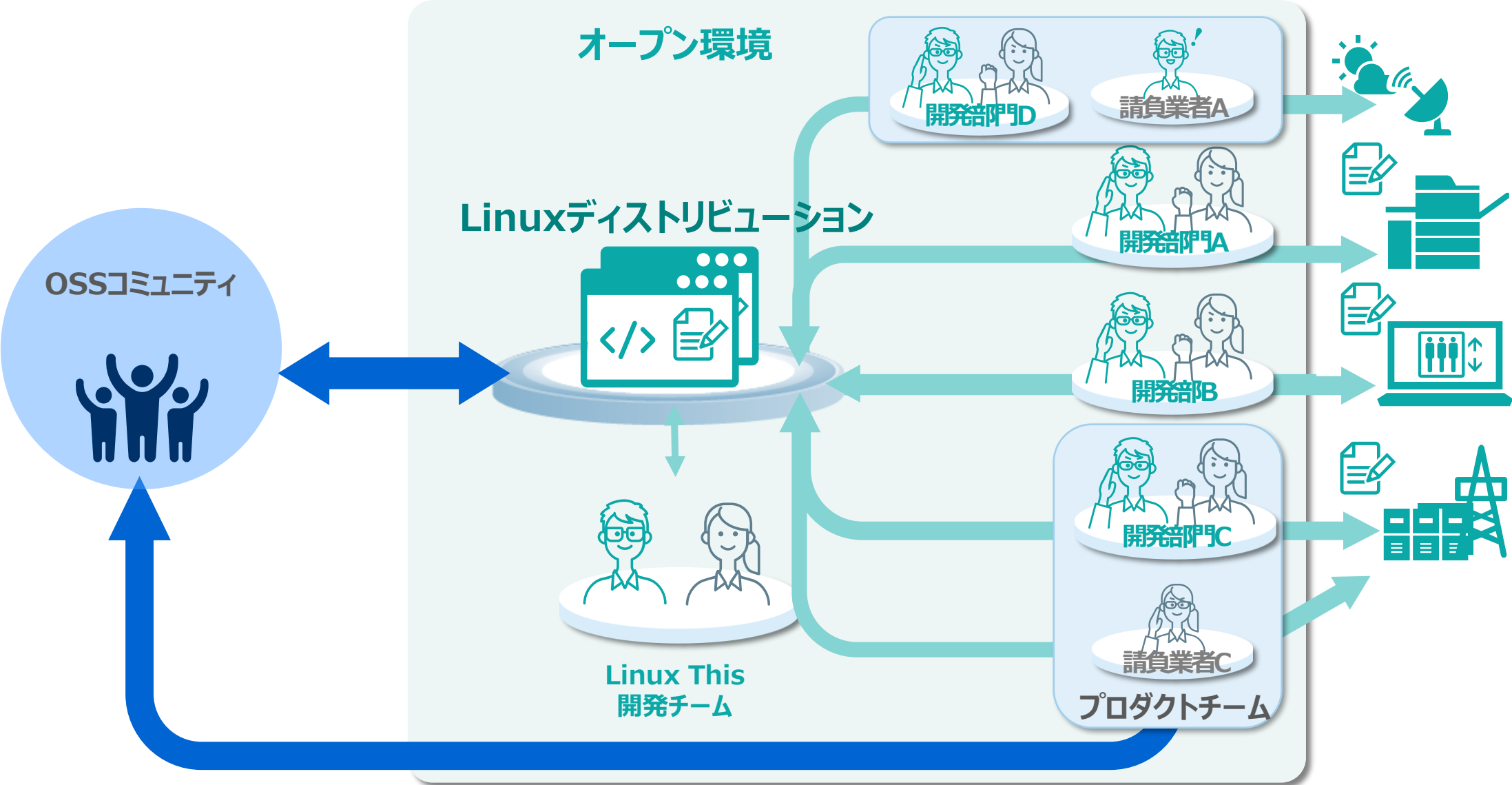
# ケース#1: 全社向け共通Linuxディストリビューション



# ケース#1: 全社向け共通Linuxディストリビューション



# ケース#1:全社向け共通Linuxディストリビューション



# ケース#1:全社向け共通Linuxディストリビューション

## インナーソースの原則は標準化と再利用を加速させました

### 背景

- 複数の事業部門がLinuxを必要とした → Linux適用における共通の課題もあった

### アプローチ

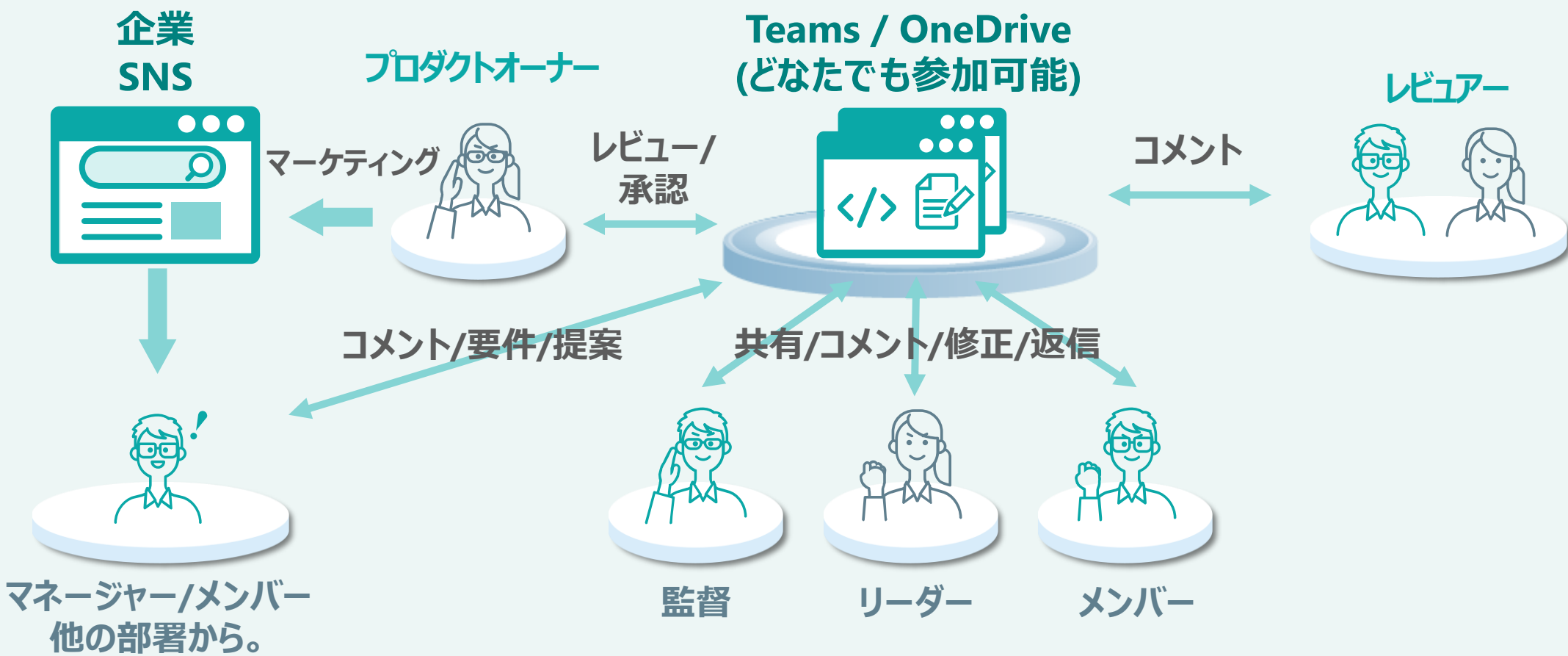
- 全社オープンでアセットをリリース
- 改善提案は、メールやプルリクエスト、 이슈を通して受け付け

### 結果

- 社内標準化したことで開発効率が向上
- 外部パートナーと連携して開発する際にもバグ報告を受け付けたことで品質向上
- 再利用性が向上し、複数の事業部で適用コストが削減
- 社内から上流のOSSコミュニティへ容易に貢献が可能

# ケース#2:技術中計の策定

## オープン環境



# 計画は立場を超えて本来必要とする人に届いてこそ価値を持つ

## 背景

- 共通技術部門は複数事業の利益となる技術を開発
- 技術中計の策定において文書と意思決定プロセスの両方を会社全体に公開
- 「誰でもコメントできる環境」を作り、透明性を確保

## ポジティブな影響

- 事業部門からの多様なフィードバック取得に成功
- 異なる視点が入ることによって優先順位や盲点が明らかになった
- 自部門の活動の可視性を高めたことで、潜在的な協力者を生み出した
- オープンな議論の文化を築くことに興味を持ってくれるメンバーが現れた

閉ざされた部屋で議論した結果の回答は不信感を生み、議論も進歩も生まない

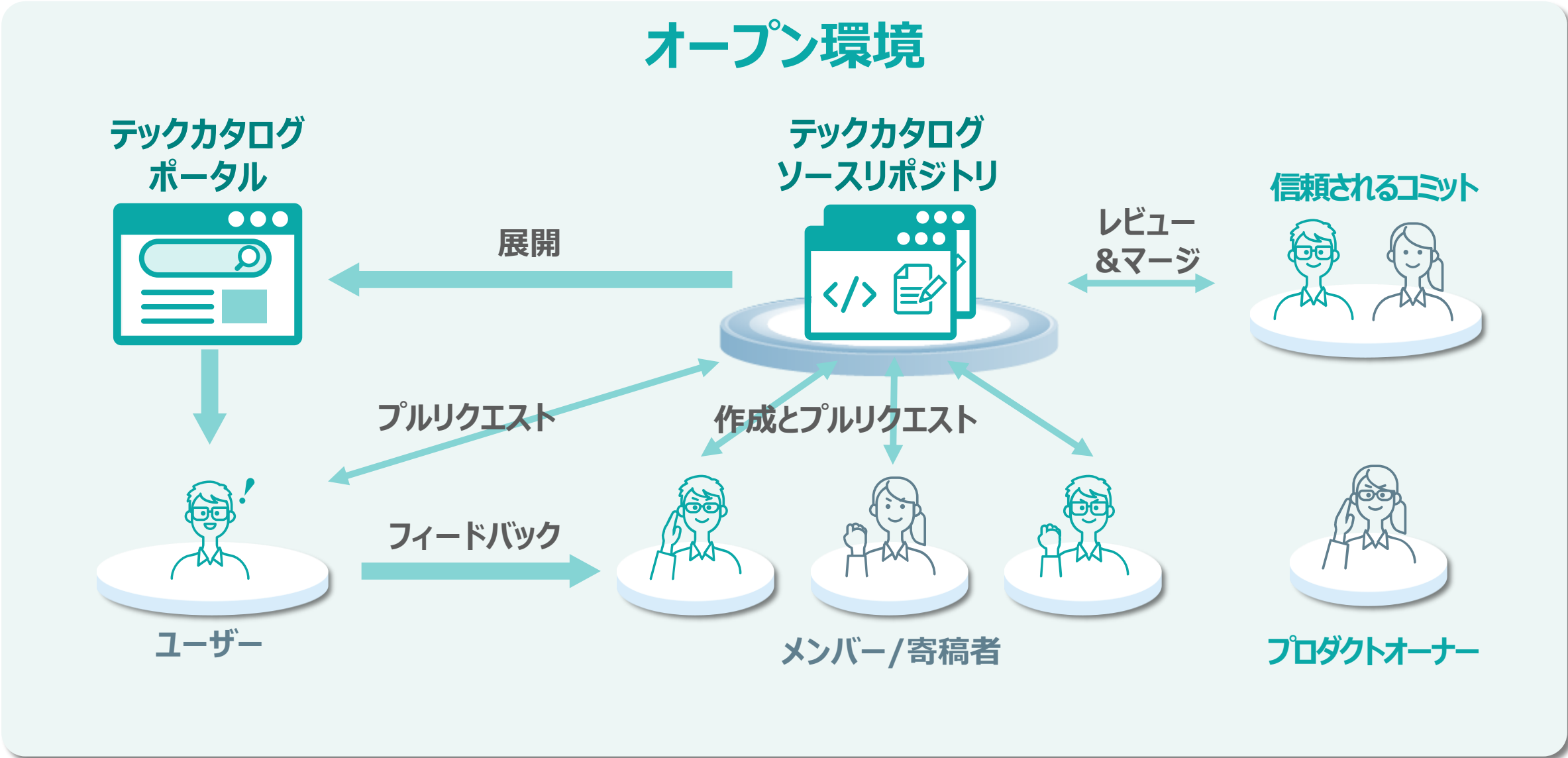
### 失敗

- 最初のコメントへの反応は遅く(1週間以上も何の対応がないことも)
- コメントをくれた人に見えないところで結論を出してから回答
- 既存の立場を擁護する方向に偏りがちで、新しい代替案の提案が不足

### 学んだこと

- フィードバックには何かしら即対応
- (中で長く議論するより) 議論の過程を見える化することの重要性を改めて認識
- コミュニティ視点から対話のための空間をデザインすることの重要性

# ケース#3:技術カタログ



# 静的文書から生きた知識に

## 概要

- 社内共有用の技術カタログをMarkdownベースの文書で作成
- ブラウザからアクセス可能;プルリクエストを通じて提出された改善提案

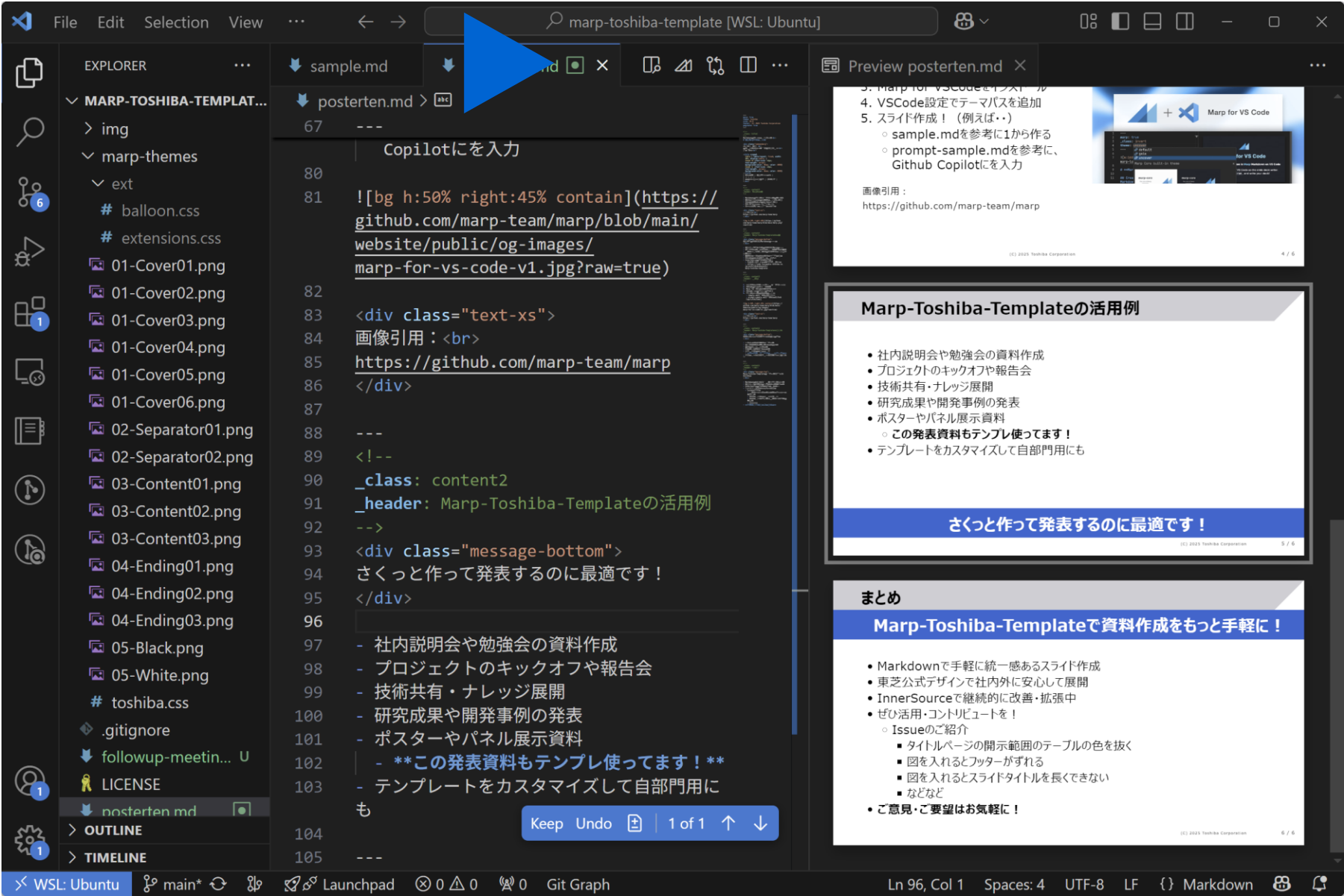
## 役割の明確化

- プロダクトオーナー: カタログ全体の監督
- トラストッドコミッター: PRをレビューし、一貫性を確保する
- コントリビューター: コンテンツや修正を作成してPRを通して提出
- 低摩擦での改善:「何か問題があればPRで直す」と、参加を促します

## 結果

- 編集履歴の透明化 → 編集の理由が判断できる
- 気づいた人が自分で直す → 更新のボトルネックを回避
- オペレーションの単純化による活動継続性の確保

# ケース#4: Marpテンプレート (社内展示でコミュニティ構築)



Markdownで  
記載した  
スライドの内容

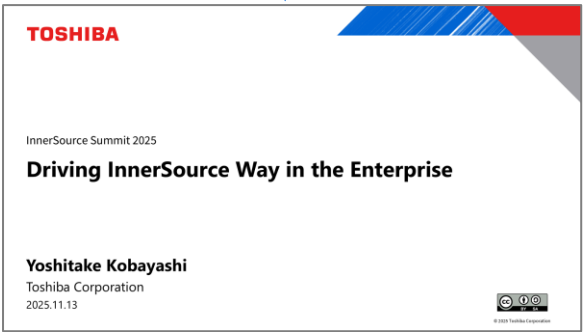


企業向けスライド  
スタイルテンプレート



Marp

参考:<https://marp.app/>



## ケース#4:Marpテンプレート（社内展示でコミュニティ構築）

### 「痛みを取り除く」というシンプルな解決策は広まりやすい

#### 概要

- OSSのMarpを採用 → シンプルなテキストスライド作成を加速させたいという要求を解決
- 企業フォーマットに合わせたテンプレートを作成して公開

#### スケールへのアプローチ

- ユーザー募集や改善提案の収集に向けたポスター
- 初期段階:多くのフィードバック→後期段階:安定した使用継続

#### 結果

- 企業形式のマークダウンベースのスライドが簡単に作成できるようになった
- フィードバックによりテンプレートの品質が大幅に向上
- 長期的にはメンテナンスが少なくなっているが、ユーザーにとっては高い持続的価値をもたらす

**企業におけるインナーソースの推進 どのように実現するか？**

# インナーソースを始めたモチベーション

内向きなモチベーション

似たようなことをするなら  
一緒に解決したい！

興味

外向きなモチベーション

日本の企業文化を  
変えたい！

危機感

社会と組織を変革する

日本も会社も元気にする！

# 再掲：こんな世界を会社の中（日本中の企業の至るところ）に作りたい！

## 今まで



サイロごとに閉じている  
何をしているか自部門以外のことは知らない

## 目指す姿



サイロを取り払い、コラボレーションが各所で発生！

## 小さく始めて、透明性を保ち、貢献を可能にし、共に支えながら成長する

| ステップ                         | 何をすべきか                                 | なぜ重要なのか                    |
|------------------------------|----------------------------------------|----------------------------|
| 1. 「誰のためか」を定義する              | 対象のユーザーやチームを明確にする                      | 誰が得をするかが分かれば、共有すべきものが明確になる |
| 2. 小さく、でもオープンに始める            | 1つの資産と2チーム（または個人）から始め、他の人も使えるようにする     | 安全に試せて、参加しやすく、抵抗が小さい       |
| 3. 透明にする                     | ソース、議論、決定、経緯などの「知」をすべて見える化する           | 透明性が信頼と再利用を生む              |
| 4. 貢献ルールを決める<br>（POとTCのSLAも） | IssueやPRの出し方と応答時間<br>（例：48時間以内）を決める    | 応答が信頼できて初めて貢献は続く           |
| 5. 役割の明確化                    | プロダクトオーナー、Trusted Committerなどの役割を割り当てる | 「オープンだが無管理」を防ぐ             |
| 6. 指標を測り共有する                 | PR数、採用、統合までの時間、貢献者の広がりを追う              | 成果が見えれば、支援も予算も付きやすい        |
| 7. 支援を得てスケールさせる              | 成果と予算を確保し、正式な後ろ盾を得る                    | 拡大には熱意だけでなく仕組みが要る          |

# 個人やチームで始める、小さな一歩

## まず、この3つから

### STEP 1

#### 目的を定めて小さく開始

自分たちのリポジトリを、まず社内に開く。いきなり全社制度にしなくても大丈夫。



### STEP 2

#### 入口を用意する

READMEと貢献の手順を書く。ゲストが迷わないように。



### STEP 3

#### 一度、越えてみる

隣のチームの困りごとに、一度だけ手を出してみる。

## 続けるために必要なこと

#### 貢献が評価される仕組み

評価されない貢献は、続かない。

#### オーナーを、決めておく

誰が対応しているかが曖昧だと、必ず止まる。

技術だけでなく、文化と仕組みの最低限を「小さく」決めておく

# 組織で始める際に備えておきたい3つの武器

## 解決する課題

(場としての)  
システム

Github、  
GitLabなど

(実践可能な)  
ルール

「やってもOK」を  
明示する

マインド

ここが難所！

人

**2021  
Summer**

# **共創 & デジタル人財 BOOST PROGRAM**

プロトタイプ思考による最先端の「もの+こと」づくりを実践する  
アウトプットコミット型プログラム

# OSS / ISS Manifesto

ないものだけを作ろう



あれば使う、なければ作る

ノウハウも共有する

使う時にはルールを守る

コミュニティの仲間を尊重しよう



Give & Givenの精神で共栄する

気楽に意見交換する

批判するより提案する

みんなで共創しよう

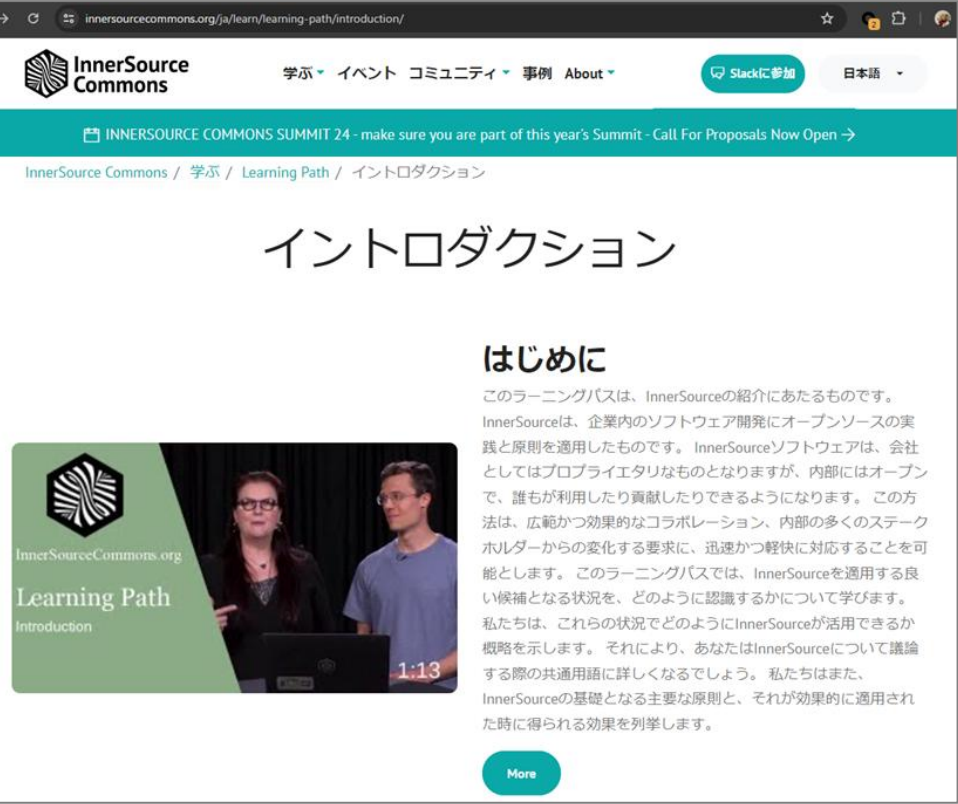


アイデアレベルから公開し、コミュニティで育てる

多くの人の目・手で、バグ修正、品質向上をスピードアップさせる

コミュニティの力を信じ、自分からオープンな気持ちで会話する

① InnerSource Learning Pathを  
全て翻訳！（Upstream first!）



② CodeZineに記事執筆



③ InnerSource Commons Japan Meetupで発表



引用元：<https://innersourcecommons.org/ja/learn/learning-path/>  
<https://codezine.jp/article/detail/14809>  
<https://innersourcecommons.connpass.com/event/271207/>



# (武装する) インナーソースを広める武器一覧



Ref: [InnerSource Commons: エンジニアの組織内コラボレーションを加速する - イベント一覧](https://innersourcecommons.org/)  
<https://innersourcecommons.org/>

# (武装する) インナーソースを広める武器一覧



## (武装する) 集まって考えた・・・



## (武装した) みんなで歌を作った



# インナーソースを広める武器一覧



# ニフティ株式会社のインナーソース活動開始に（いつのまにか）貢献

## インナーソースを始めるきっかけ

2022年のあの頃。エンジニアリングマネージャーとして、リソースの効率化について悩みがありました。

- 外部のリソースを使わず、もっと内部で回せるやり方はないか？
- 特定のチームに開発集中してもリードタイムを少なくできる方法はないか？

衝撃的な出会いでした！！  
この世には「インナーソース」というものがありました！！

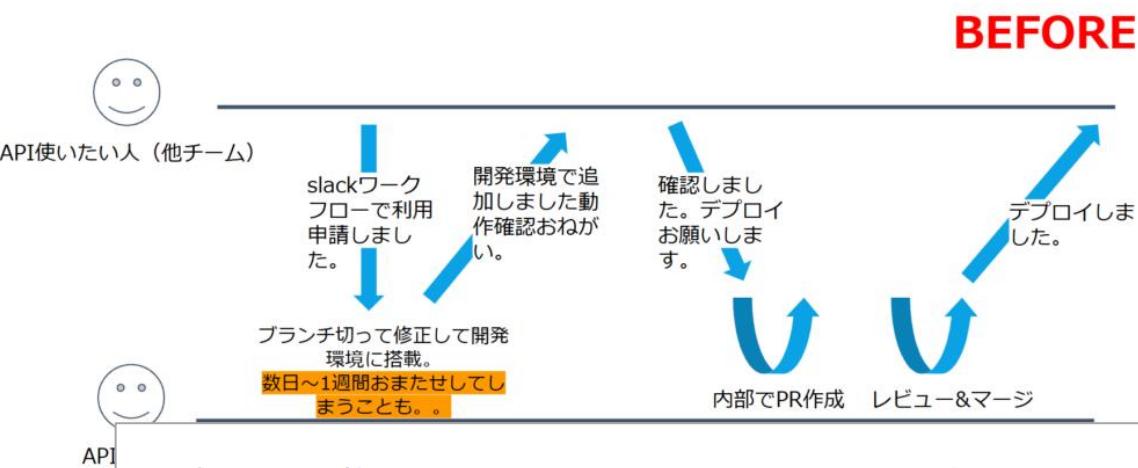
やって  
組織に



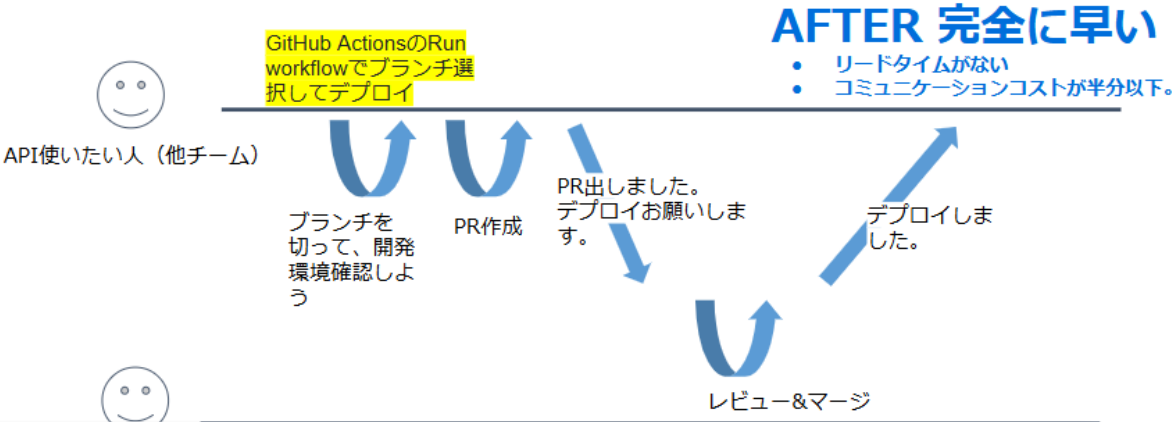
最初に私が読んだのは、右のcodezineに乗っているという東芝の小林さんが書いた記事「オープンソースの開発スタイルを企業内で実践するインナーソースとは？ メリットとポイントを理解する」です。ものすごくわかりやすかったです。で、これ導入したら余力のある人がほかのチームの開発手伝えるようになるのでは？と安易に思いまして、組織に「インナーソース」とやらを導入したいと強く思ったわけです。

# ニフティ株式会社のインナーソース活動開始に（いつのまにか）貢献

## API利用者定義ファイルのリードタイム短縮



## API利用者定義ファイルのリードタイム短縮



個人で作ったツールへのコントリビュートや、業務に少し踏み込んだところだと、API利用者定義ファイルの追加について明らかにリードタイムを短縮できた事例などを紹介しました。

## インナーソースがうまくいっていないところ

このブログでは、これまであまり語っていなかった、いまのところうまくいっていないところを文章化しておきます。やってみていくと、ハマらないパターンもあることが体感としてわかってきました。例えば、ドメイン知識が違う同士のストリームアラインドチームのコラボレーションです。

良いだけでなく、うまくいかない事もノウハウとして公開してくれた

# 三菱電機株式会社のインナーソース開始とOSPO & ISPO設立に貢献



**Yoshitake Kobayashi** • You  
Toshiba Corporation

1d ...

Congratulations!

Like ·  1 | Reply · 1 reply | 78 impressions

とってもポジティブな  
フィードバックがもらえた！



**Shingo Oidate** **Author**  
General Manager, Open Source & InnerSource Program Office

13h ...

Thank you, Kobayashi-san! I have great respect for you as a pioneer of InnerSource in Japan. It was because of your presence and influence that we were able to establish an InnerSource Program Office within our company!

Love ·  1 | Reply

外堀から埋めていく・・・

# 4-4. AI時代のインナーソース

AIが人と知をつなぐ

今も「共創」— AIが、開いた土台の上の共創を加速する

| 目的地                                                                                                                       | よりどころ                                                                                                                 | 進み方                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ソフトウェア・デファインド<br><br>●出荷後も価値を作り続ける状態 | 価値を作る3つの領域<br><br>●領域ごとに、打つ手を変える | Collaboration<br><br>●外はOSS、内はインナーソース |

# AI時代のインナーソース —「ソース」の定義が広がる



ADR: Architecture Decision Record

AIは、コードだけでなくこの「知」を食べて動く

# 人間中心から、AIネイティブへ — 価値転換マトリックス

| 観点      | これまで（人間中心） | これから（AIネイティブ）                     |
|---------|------------|-----------------------------------|
| 主役となる資産 | ソースコード     | ADR、プロンプト、規約、意図など<br>すべてのドキュメント・知 |
| 人間の役割   | ロジックを書く    | 生成物を検証、審判し、指示をする                  |
| 知の扱い    | 局所的・属人的な蓄積 | AIがアクセスできる透明な資産                   |
| 発見の方法   | 人間が検索して探す  | AIが文脈に合わせて届ける                     |

インナーソースの価値は、AI時代にむしろ上がる

## AIはデータの断片化を嫌う — サイロが生産性を殺す

AIが力を出す前提は、コード・設計判断・議論ログなどの「知」が途切れず繋がっていること  
縦割りのサイロは、AIにとってもコンテキストの断絶になる

つながっていないことで生まれる類似成果による負債



保守



障害対応



AIコスト

分断されたままではAIも力を出せず、高い生産性が技術的負債を生むリスクに

# ガバナンスのジレンマ — 閉じるか、開くか

アクセス権限を絞り情報を閉ざせば、AIの生産性は死ぬ。 **かといって全開放はできない — だから「峻別」する。**

## 真の秘匿情報

個人情報・絶対的機密



**厳格に保護する**

## 組織の知恵

社外秘だが、社内には公開できるもの



**デフォルト・社内オープン**

**厳格な保護範囲を抑えつつ、AIが学べる面積を最大化する**

# 「背中を見て覚える」からの卒業



これまで  
探しに行く



これから  
AIが、文脈に合わせて届けてくれる

日常の活動から自然に知恵を抽出し、誰でも使える形式知に変換する

## 今日から取り組めること

### 再利用ファースト

AIに生成させる前に、  
社内の共有資産を当てる

### 意図を伝えて残す

たとえばADRを残すことを、  
開発プロセスに組み込む

### 審美眼を磨く

目利きAIを育てるコミュニティを、  
組織横断で

暗黙知を、人もAIも使える形式知へ

# AIが、境界を越えるときの摩擦を下げる

越境のコストは、いつも「文脈が分からないこと」だった

## 読み解きを代行する

他部署のコードや設計判断を、  
AIが橋渡しする

## 越境が速くなる

部門横断のコード検索・レ  
ビュー・オンボーディングが加速

## ただし、前提がある

文脈が途切れているとAIも力  
を出せない

**共有すべき資産は、コードから「意図」へ⇒すなわち「知」**

**AI時代においてインナーソースは知を循環させるインフラとなる**

## 難所：オープンにすれば済む、わけではない

### ① オープンには、責任が付いてくる

供給網の責任 サプライチェーンセキュリティ  
出所の管理 SBOM、ライセンスと出所の追跡  
規制との関係 CRAとは、裏表の関係にある

### ② 失敗するのは技術ではなく、文化

過剰な「社外秘」 知もAIも、閉じ込めてしまう  
オーナーシップ 引き取り手が曖昧だと止まる  
インセンティブ 評価されない貢献は続かない

ここは私自身、何度も失敗してきた・・・

環境やルールがあっても人が付いてこないとAIも人も動けなくなる

# オープン化が進むとAIがコラボレーションを加速する！

## オープンな場

課題の共有が  
最初のステップ

実装(コード)や文書などの成果物

課題解決のアイデア



課題

提案者



AIが人と課題と  
実装をつなぐ

ハース大事！



賛同者  
(有識者)



賛同者  
(開発者やユーザー)

# 5. Conclusion

AI時代のキョウソウ戦略

## AI時代の価値創造サイクル — 開いた土台の上で回る

変わらないのは3つの領域。 変わるのは「開き方」。



肝は、人と人。 コラボレーションとコントリビューションが、この循環を回す。

AIが、ひらく・出会う・創るの3つすべてを加速する

3～5年後、組込み開発の現場はこうなっている（いてほしい！）

「開いた土台」が当たり前になった日常。 変わるのは道具より作法。

## 過程の公開

設計判断が標準成果物

「なぜそう決めたか」がコードと一緒にレビューされる



## AI検証

レビューの主役が交代

人は意図と前提の妥当性を見る



## インナーソース

社内公開リポジトリが増加

部門を越えたPR・Issueが日常化

残す・任せる・開く。 この3つが習慣になった現場が、いちばん速い。

判断を残し、検証を任せ、リポジトリを開く現場が標準になる

# AIがどれだけ速くなっても、価値を生むのは人と人が寄り添うこと



01

### 隣に、手を伸ばす

隣のチームの困りごとに、  
手を伸ばす



02

### 知を、置いておく

自分が持っている「知」を、  
誰もが探せる場所に置く



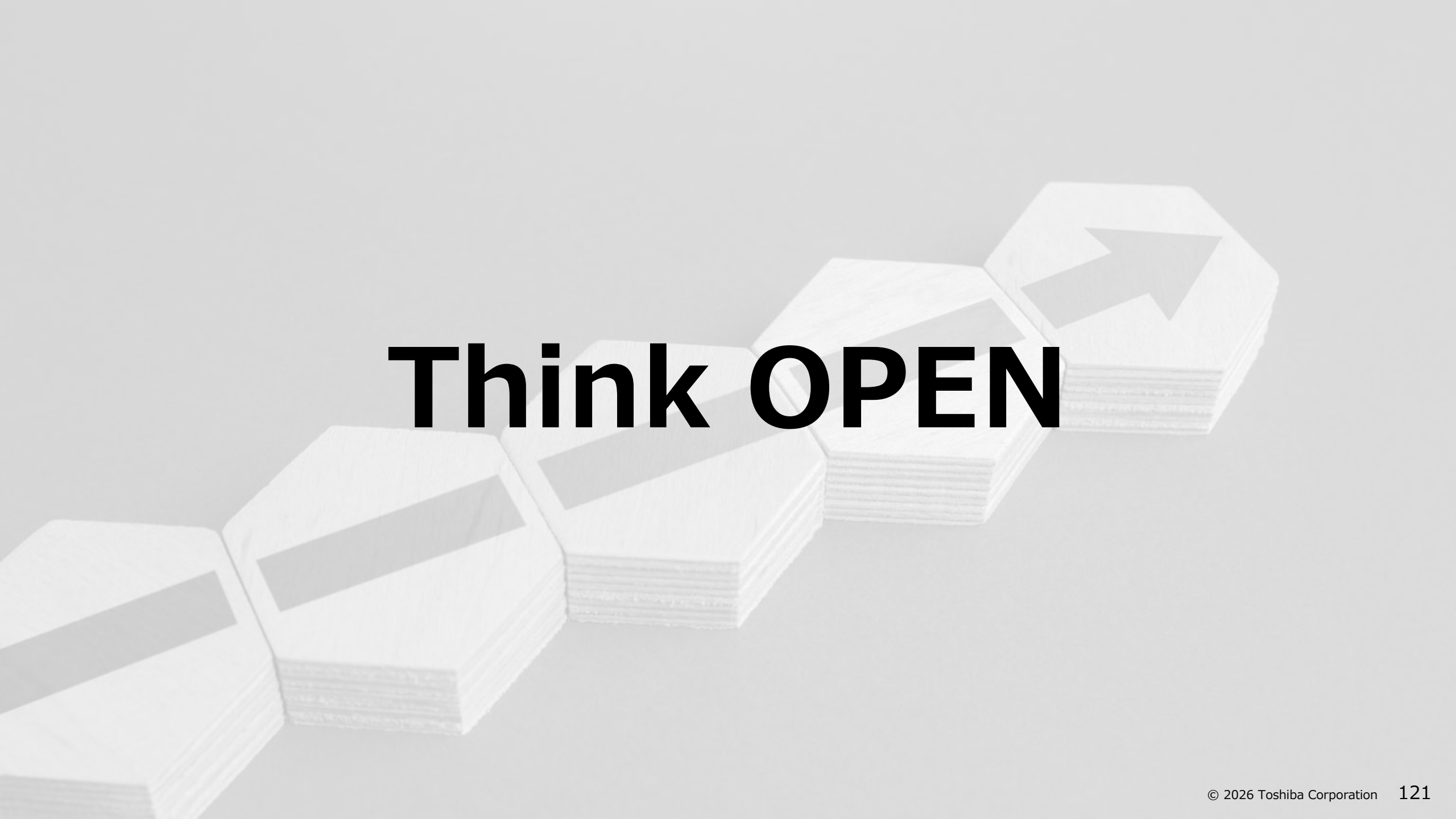
03

### 感謝を、返す

誰かの一步に、  
レビューと感謝を返す

すなわち 「今日添う」

AI時代だからこそ仲間を大切に！

A series of white, folded paper napkins are arranged in a slightly overlapping, diagonal line from the bottom left towards the top right. Each napkin has a large, light gray arrow printed on it, pointing towards the right. The text "Think OPEN" is superimposed in the center of the image, with "Think" in a smaller font and "OPEN" in a larger, bold font.

# Think OPEN



Questions?



# 人と、地球の、明日のために。

**Committed to People,  
Committed to the Future.**

# **TOSHIBA**

