インナーソースで未来を築こう 

InnerSource Gathering Tokyo 2025
2025-09-12

ワイクル株式会社 代表取締役
角 征典（カド マサノリ）@kdmsnr
kado.masanori@waicrew.com

スライドの場所

https://kdmsnr.com/slides

オープンソースで未来を築こう（2025-05-30）

私の（数少ない）オープンソースプロジェクト

きょうお話すること

- 1️⃣ **いろいろと意識合わせ**
  - 最初の講演なので前提を共有したい
  - インナーソースのことを知らなかったので調べたよ！
- 2️⃣ **インナーソースに参加する動機**
  - 「なんで参加しなきゃいけないんですか？」と言わないために
- 3️⃣ **インナーソースを組織に広げる**
  - 自分が納得できたらあとは他の人にも広げよう

1 いろいろと意識合わせ

インナーソースとオープンソースのあれこれ

まずは歴史から：インナーソースの誕生 🎂

1998年にTim O'ReillyがIBMのチームに話した内容がベース:

- オープンソースの長所は**ライセンスとは無関係**である
  （OSS *[1]* はライセンスで定義される、という考え方が嫌い）
- 重要なのは、**コラボレーション、コミュニティ、参入障壁の低さ**
- OSSに乗り出せない企業でも**OSSの開発手法を活用できる**

[1] OSS: オープン・ソース・ソフトウェア

出典：『Adopting InnerSource』
Go through Wayback Machine:
http://archive.oreilly.com/pub/a/oreilly/ask_tim/2000/opengl_1200.html

スコットの法則（別名：インナーソースの法則）

> 十分な人数の開発者がつながっていれば、すべてのソフトウェア開発は、**OSSのベストプラクティス**を模倣する。
> —Tim O'Reilly

スコット・ガスリーとマーク・アンダースがクリスマス休暇中に立ち上げたプロジェクトに社内の開発者たちが協力し、最終的に ASP.NET として製品化された、という逸話から。

出典：『Adopting InnerSource』

インナーソースとは？（ちゃんとした定義）

> 私企業が**オープンソースの手法や戦略**を使用して、ソフトウェアを開発すること。高品質のソフトウェアを生み出し、組織のボトルネックを解消できる素晴らしい手法である。
> – ダネーゼ・クーパー（InnerSource Commons創設者）

出典：Learning Path Introduction - 01: What Is InnerSource? https://www.youtube.com/watch?v=I93ohSHhr5U

そもそも「オープンソース」とは？

1. **成果物がオープンに利用可能であること**
   - ライセンスがOSD *[1]* に準拠していること
   - 法律的な「オープンソース」
2. **コミュニケーションがオープンであること**
   - 誰もがコミュニティに関与できること
   - 文化的な「オープンソース」← インナーソースはこちらに注目

[1] The Open Source Definition

参考：John Mertic『Open Source Projects - Beyond Code』Packt Publishing

文化的なオープンソースと『伽藍とバザール』(1999)

- 「**少人数の計画型**」と「**大人数の適応型**」の対比
  - 前者には法律的な「オープンソース」も含まれる
  - ［余談］アジャイルは「**少人数の適応型**」を目指した（後述）
- **バザール方式の前提条件：**
  - 動作可能でテスト可能なもの（すぐいじれるもの）がある
  - 他者を引きつけ、優れたアイデアや成果を認識できる能力を持つ
- 貢献者の動機は**エゴブー**（egoboo/ボランティア活動が世間に認められる喜び/もともとはSF雑誌の投書欄に名前が載ること）

補助線としての「Worse is Better」

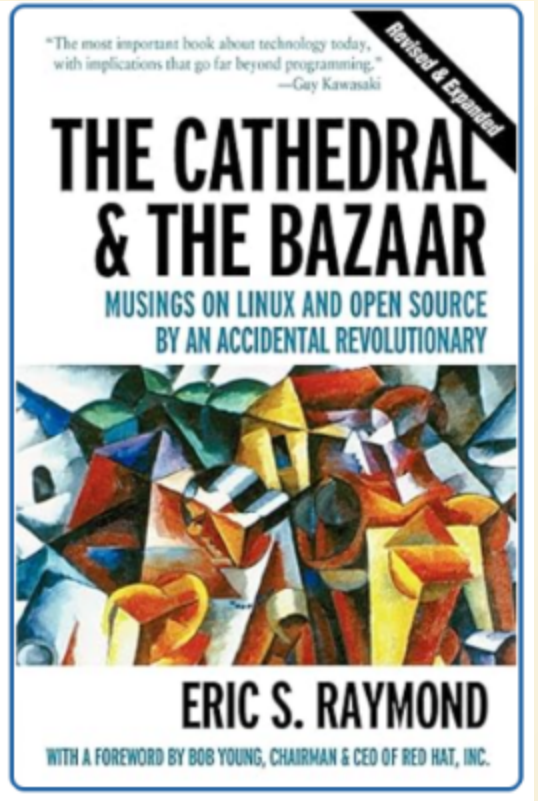
- 「**古典的ハッカー文化**」と「**Unixハッカー文化**」の対比
- 古典的ハッカー（MIT/スタンフォード方式）
  - 正しくて完全な設計を目指す
  - 完成までに時間がかかる（完成しないこともある）
- Unixハッカー（ニュージャージー方式）
  - 簡単な実装を目指す（使いにくくても目をつぶる）
  - とりあえず動く、大勢を巻き込む、移植が容易、迅速に普及

出典：デザインの「悪い方がよい」原則 http://chasen.org/~daiti-m/text/worse-is-better-ja.html

ハッカーたちから尊敬されるためにできること

1. ソフトウェアを書く
2. テストやデバッグを手伝う
3. 有益な情報を公開する
4. インフラが機能し続けるように
   手伝う
5. ハッカー文化そのものへの貢献

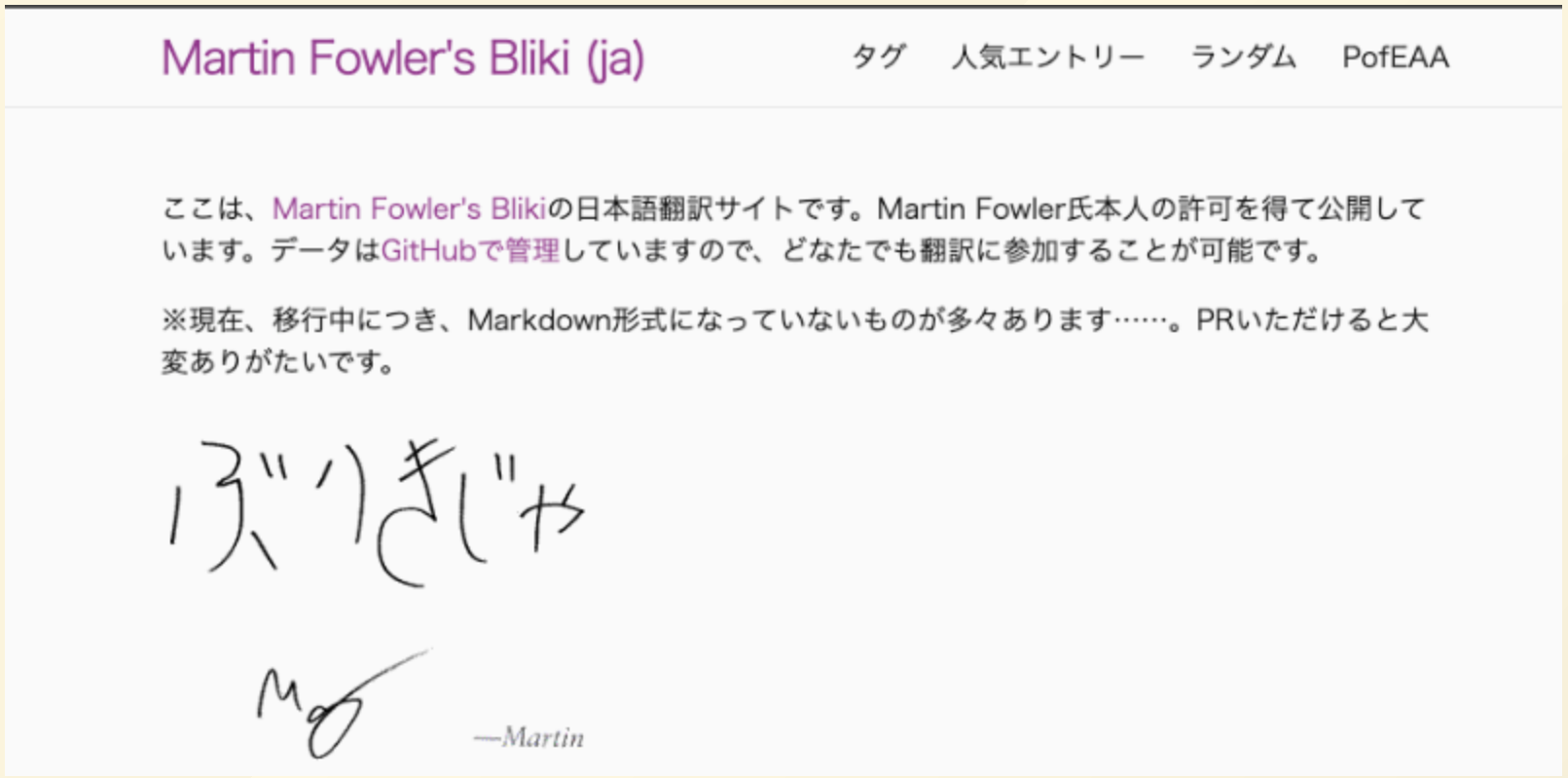
コード以外でも貢献できる!! 😊



出典：ハッカーになろう https://cruel.org/freeware/hacker.html

コード以外の貢献の例（bliki-ja.github.io）

私の（数少ない）オープンソースプロジェクト2（2003年〜）



コード以外の貢献の例（bliki-ja.github.io）

- 当時（2003年頃）は日本語のアジャイルの情報が少なかった
- 英語のブログを読んでメモを書いていたが、翻訳したほうが良さげ
  - だが、英語に自信がなかったので、間違っていたときに誰でも書き換えられるようにWikiを使った。議論の場にもなった。
- 2015年に GitHub Pages に移行
  - 貢献者の増減はよくわからない（Wikiのときは記録なし）
  - Wikiよりもハードルは高くなったが、記録が残るようになった
- 翻訳や文書作成は誰もが着手しやすい貢献のひとつ

第6章 プルリクエスト以外で貢献する

ファイルを触らない貢献の例:

1. （他の人の）貢献のレビュー
2. （他の人の）貢献のテスト
3. イシューのトリアージ

出典:『オープンソースで未来を築こう』第6章

1️⃣貢献のレビュー

- 大前提：経験豊富な人でも間違える && メンテナーは忙しい
  - **リーナスの法則「目玉の数さえ十分あれば、どんなバグも深刻ではない」** → レビューアは多ければ多いほどいい!!
- AIコーディングの時代はレビューに時間がかかる!!
- まずは他の人の貢献やレビューを観察する（自分で答え合わせ）
- 小さくて時間のかからないものからレビューする
- ただし、自分の好みよりもプロジェクトのドキュメントを優先する
- 自分が適任かどうかは常に考える（手に負えないこともある）

参考:『オープンソースで未来を築こう』第6章

1️⃣貢献のレビュー（フィードバックする）

- フィードバックは正解や命令ではなく「対話」の始まり
  - **対話が両者（とコミュニティ全体）の学びの場になる**
- ポジティブなフィードバックを心がける
  - コントリビューターがまた貢献したいと思うことが重要
  - **ポステルの法則「送信は厳密に、受信は寛容に」**
    - 言葉遣いには特に注意する
  - LGTM、スタンプ、絵文字👍だけでもいい

参考:『オープンソースで未来を築こう』第6章

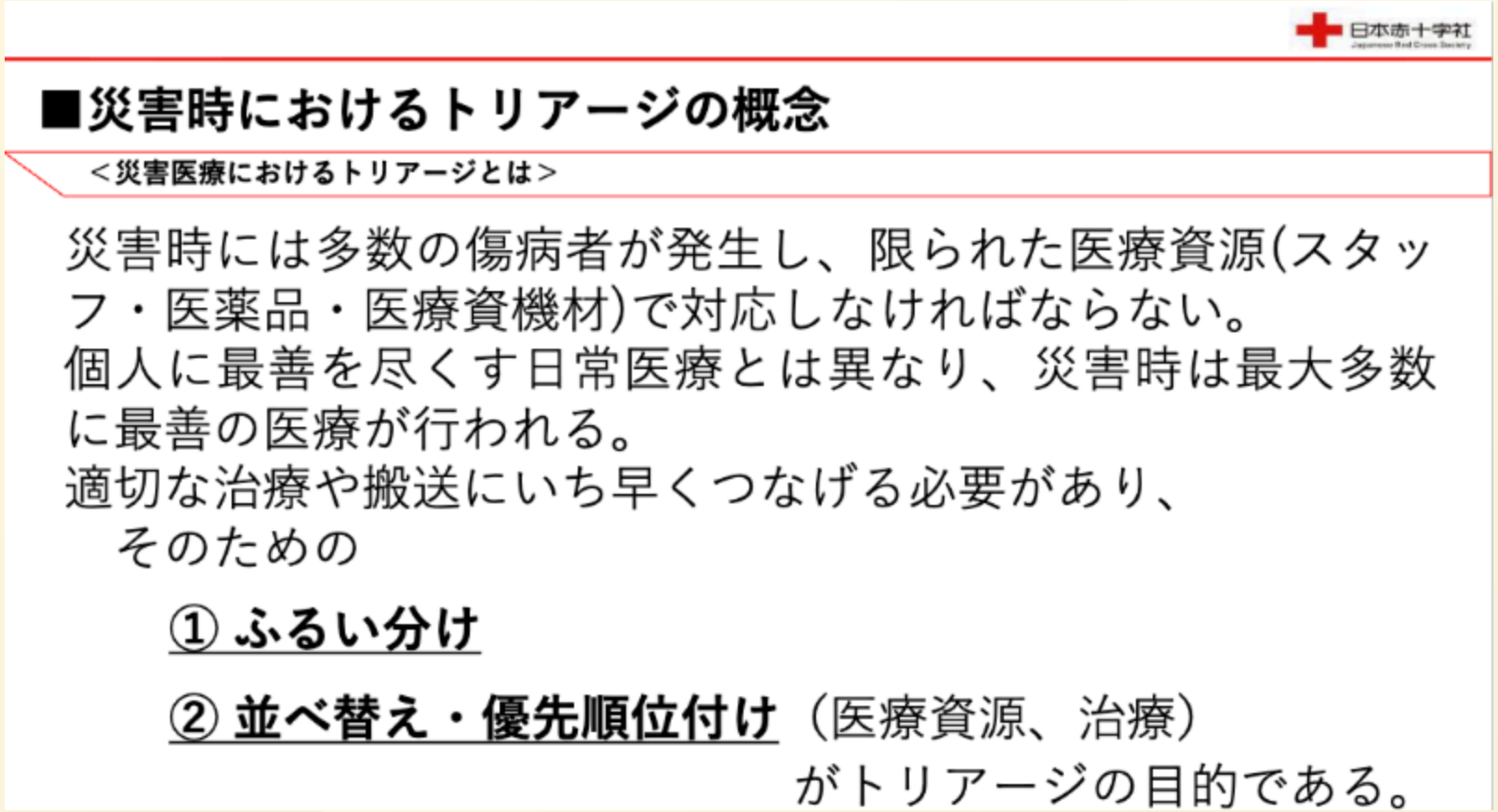
2️⃣貢献のテスト

- 規約に従っているか？
- 期待や仕様の通りに動作するか？
- 読みやすいか？ 使いやすいか？
- アクセシビリティ対策は十分か？
- セキュリティ対策は十分か？

▶️ **専門分野によって見るところは違う** → 誰でも貢献可能

参考:『オープンソースで未来を築こう』第6章

3 イシューのトリアージ（トリアージとは？）



出典: https://www.jrc.or.jp/chapter/fukushima/apeal/2025/0508_046860.html

3️⃣イシューのトリアージ

- イシューを最後まで読んで理解する
- イシューにラベルを付ける（GitHubの例）

`bug`: 予期しない問題または意図しない動作
`documentation`: ドキュメントに改善が必要
`duplicate`: 似たイシューがある
`enhancement`: 新機能のリクエスト
`good first issue`: 新人さん向け
`help wanted`: 誰か手伝って
`invalid`: 関係ない話になっている
`question`: 追加の情報が必要
`wontfix`: 対応しない

- すぐに対策が必要なイシューは、担当者にエスカレーションする

参考:『オープンソースで未来を築こう』第6章

OSSとアジャイル開発の類似点（1/2）

- **アジャイル**: 不確実で不安定な環境において、変化を生み出し、変化に対応する能力のこと *[1]*
  - そのために **「動くソフトウェア」「対話」「協調」** を重視する
  - リファクタリング、テスト駆動開発、ユーザーストーリー、CI/CD、ペアプログラミングなどのプラクティスが特徴的
- ESR「**まるでUnixの伝統やハッカーの教義のようだ**」*[2]*
  - 「だが、Unixの伝統よりもうまく説明している」

[1] *https://agilealliance.org/agile101/*

[2] *https://www.artima.com/weblogs/viewpost.jsp?thread=5342*

OSSとアジャイル開発の類似点（2/2）

ハイブリッド（企業人 + ボランティア）OSS開発チームを調査した結果、以下のプラクティスが一致していることが判明

> “頻繁なリリース、自己組織化チーム、ソフトウェア要求の進化、事前計画の回避、リファクタリング、作業の分解、変化を許容”

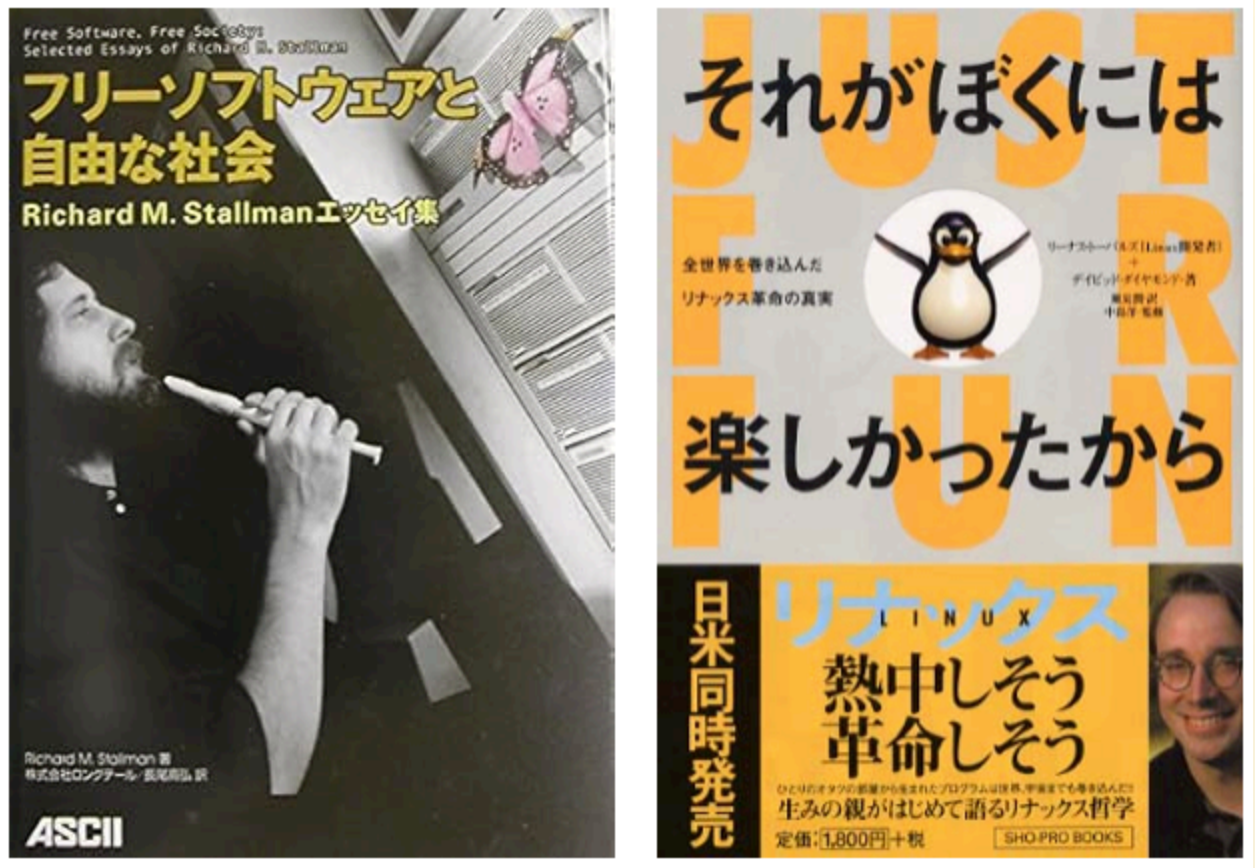
Chengalur-Smith, I., Nevo, S., & Fitzgerald, B. (2021). Enhancing Hybrid OSS Development Through Agile Methods and High Media Synchronicity. ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 52(4), 92-118.

📝 「いろいろと意識合わせ」のまとめ

- **InnerSource:** OSSの手法やプラクティスを企業に取り入れること
  - 法律的よりも文化的な「オープンソース」
  - 高品質なソフトウェア、組織のボトルネック解消
- **OSSの手法やプラクティス：**
  - バザール方式による適応型の開発
  - Unixハッカーの実用性
  - **コード以外でも貢献できる余地がある**
  - アジャイル開発との類似点

2 インナーソースに参加する動機

これまでのOSSの動機（哲学 → 楽しさ）



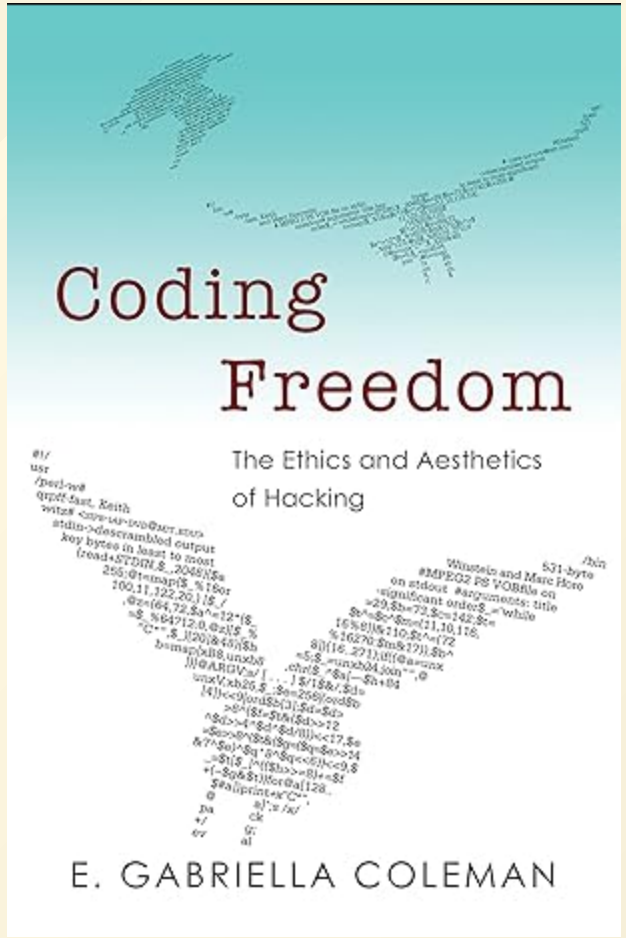
Debianプロジェクトの研究

- 技術的な喜び
- 実用性、費用対効果
- コミュニティ楽しい

↓コミュニティをまとめるために

- フリーソフトウェアへの賛同
- 倫理的原則の文書化

楽しさ/実用性 → 哲学



それでは持続性を説明できない

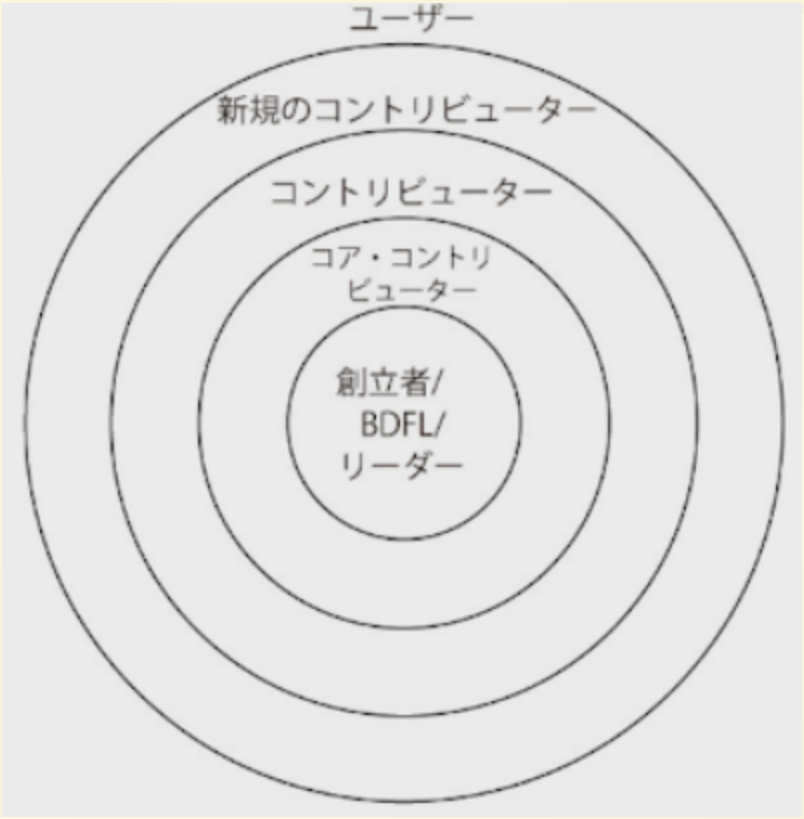
前提：コミュニティは**正統的周辺参加**の形をとり、参加者の段階ごとにやるべきことが異なる（次ページ）

1. 自分の段階に合わせてソフトウェアに貢献する
2. 新しいソフトウェアが新しい学習機会を生み出す
3. 新しいことを学習すれば、自分の段階が変化する

継続的な「学習」が重要な要素ではないか？

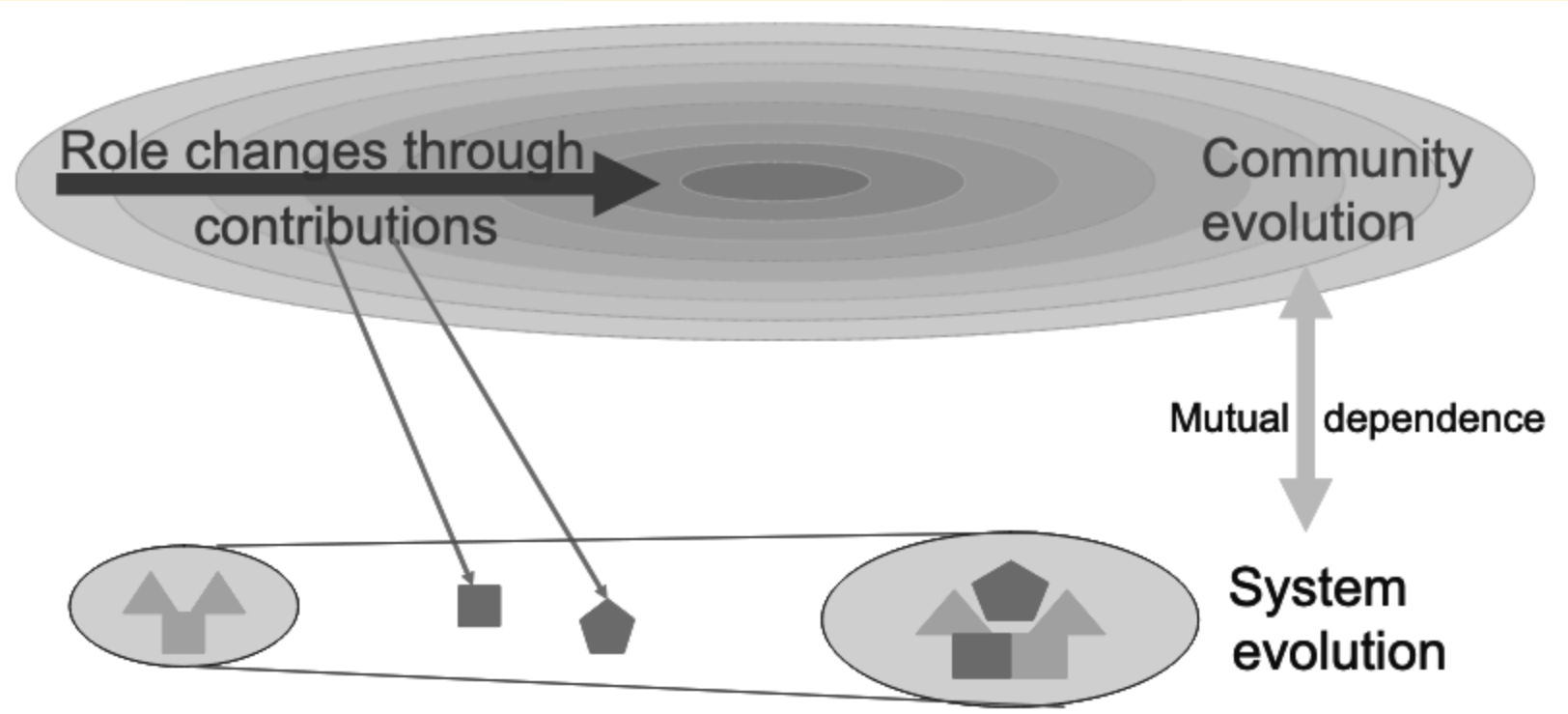
Ye, Y., & Kishida, K. (2003, May). Toward an understanding of the motivation of open source software developers. In 25th International Conference on Software Engineering, 2003. Proceedings. (pp. 419-429). IEEE.

OSSにおける正統的周辺参加



出典：『オープンソースで未来を築こう』

OSSにおける正統的周辺参加



Ye, Y., & Kishida, K. (2003, May). Toward an understanding of the motivation of open source software developers. In 25th International Conference on Software Engineering, 2003. Proceedings. (pp. 419-429). IEEE.

OSSにおける「学習」の種類

- **設立者の学習**：探求的学習、実践的学習
- **貢献者の学習**：
  - 他人のコードを読む
  - 自分が貢献したときのプロセス全般（実践的学習）
  - **コミュニティの議論に参加する**

Ye, Y., & Kishida, K. (2003, May). Toward an understanding of the motivation of open source software developers. In 25th International Conference on Software Engineering, 2003. Proceedings. (pp. 419-429). IEEE.

コミュニティの議論に参加するときに気を付けること

- **適切なメディアを選ぶ**：メール？ イシュー管理？ ビデオ会議？
- **「読む・書く」を心がける**：誤解を避ける。AIに任せてもいい。
- **他者の視点を理解する（できなければ質問する）**：そのまま。
- **競争ではない**：論破よりもプロジェクトの利益を優先しよう。
- **あなたのことではない**：議論は個人攻撃ではない。
- **常に礼儀正しくする**：大人でしょう？
- **クソ野郎は無視する**：大人でしょう？

出典：『オープンソースで未来を築こう』第7章 + 第9章

その後、2010年代から動機に変化が

- 「楽しみ」「学習」「知識共有」などの動機は変わらない
- 「利他的」「仲間意識」「評判」などの**社会的動機が増加**
  - （推測）GitHubが登場したからではないか？
- 「自己利用」などの**実用的動機が低下**
- 初期段階では**「キャリア形成」も重要**

Gerosa, M., Wiese, I., Trinkenreich, B., Link, G., Robles, G., Treude, C., & Sarma, A. (2021, May). The shifting sands of motivation: Revisiting what drives contributors in open source. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1046-1058). IEEE.

💪インナーソースに参加する動機を高める設計

1. 「楽しい」や「便利」は大前提
2. 「学習」や「キャリア形成」は手厚くサポート（次ページ）
3. 「社会的動機」を高める施策

目標シートとスキルシートの作成をサポート

私の目標

- CSS を習得する
- JavaScript の勉強を始める
- デザインにおける git の使い方を学ぶ
- 少なくともひとつの UI のアクセシビリティを向上させる
- プログラマーとのコミュニケーションや協力を練習する
- デザインの提案書を書く
- デザイナーの師匠を見つける
- ポートフォリオに作品をひとつ以上追加する

私が貢献できそうなスキル

- グラフィックデザインのトレーニング
- スペイン語と英語
- ブランディングの経験
- HTML が得意
- CSS がわかる（もっとうまくなりたい！）
- プログラミングの入門コースで優秀だった
- InDesign と Photoshop が得意
- グラフィックデザインの学生ミートアップのリーダーだった

出典：『オープンソースで未来を築こう』第4章

Huawei社のインセンティブ設計

ダニエル・ピンクのモデル（ありがちだが...）を使用：

- **自律**：貢献するプロジェクトや時間を自由に選択できる
- **熟達**：活かしたい or 高めたい専門スキルを選択できる
- **目的**：プロジェクトの交流イベントに参加して、プロジェクトの目的を理解し、メンバーとの仲間意識（Kinship）を醸成する

その結果、**貢献者が687人(2020年) → 9000+人(2022年)に増加**

T. Dey, W. Jiang and B. Fitzgerald, "Knights and Gold Stars: A Tale of InnerSource Incentivization," in IEEE Software, vol. 39, no. 6, pp. 88-98, Nov.-Dec. 2022, doi: 10.1109/MS.2022.3192647.

📝 「インナーソースに参加する動機」のまとめ

- 黎明期は「**哲学**」や「**楽しさ**」が中心
- プロジェクトを持続させる鍵は継続的な「**学習**」
- 近年は「**キャリア形成**」や「**社会的動機**」も重要
- **インナーソースに参加する動機**をうまく設計したい
  - 「**目標シート**」や「**スキルシート**」を一緒に作る
  - モチベーションの3要素（自律・熟達・目的）を支援する

3 インナーソースを組織に広げる

アイデアを組織に広めるための48のパターン+α

日本語未翻訳の続編『More Fearless Change』（2015）もある。

[目覚めの電話] 

- 多くの組織は、現状維持に満足している（ように見える）
- まずは、組織が抱える問題（**組織的負債**）に光を当てる
- なぜ変化が必要なのかという切迫感を共有することから始める

The Matrix ©1999 Warner Bros. Entertainment Inc.

組織的負債の症状

- **開発者**：複雑すぎるコード、時間のかかるレビュー
- **デザイナー**：一貫性のないUI、更新されないデザインガイドライン
- **PM**：不明確な要件、スコープクリープ、リリースの遅延
- **経営者**：市場の変化に対応できない

Ahmad, M. O., Al-Baik, O., Hussein, A., & Abu-Alhaija, M. (2025). Unraveling the organisational debt phenomenon in software companies. Computer Science and Information Systems, (00), 12-12.

組織的負債の原因

- **急速な成長**: 何も対策なしに人員や部門を拡大しようとした
- **リソース不足**: 時間・人材・予算が限られており、改善が後回し
- **コミュニケーション不足**: 部門間の連携不足（サイロ化）
- **時代遅れのプロセスやツール**: 手順や技術がまったく更新されない

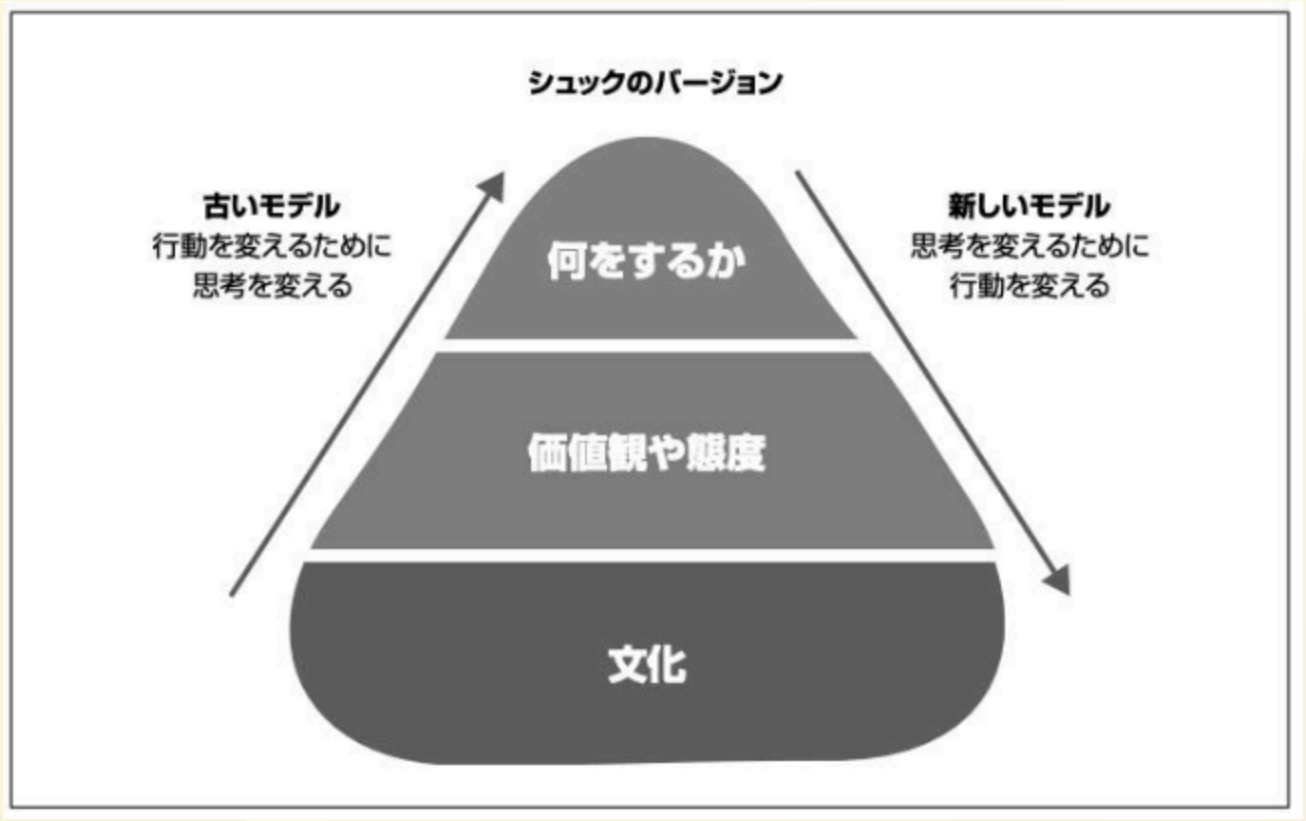
Ahmad, M. O., Al-Baik, O., Hussein, A., & Abu-Alhaija, M. (2025). Unraveling the organisational debt phenomenon in software companies. Computer Science and Information Systems, (00), 12-12.

組織的負債を生み出す「組織文化」

- 「**前例踏襲**」の文化
- 「**失敗を許さない**」文化
- 「**協力すると損をする**」文化
- 「**不適切な人材を放置する**」文化
- 「**品質よりもスピード優先**」の文化
- 「**適切なプロセスを踏む時間はない**」文化

Ahmad, M. O., Al-Baik, O., Hussein, A., & Abu-Alhaija, M. (2025). Unraveling the organisational debt phenomenon in software companies. Computer Science and Information Systems, (00), 12-12.

だが、組織文化を直接変えることはできない😭



出典：『リーンエンタープライズ』オライリー・ジャパン

アイデアを組織に広めるための48のパターン+α

日本語未翻訳の続編『More Fearless Change』（2015）もある。

😢 1. 開発者の時間的な制約や動機不足

開発者は日々の業務が忙しく、インナーソースのための時間を確保できない。また、貢献する動機が薄い。

- [協力を求める]：貢献してもらいたい人に直接話しかけてみる
- [感謝を伝える]：小さなことでも直接または文章で感謝を伝える

😢 2. 貢献したいプロジェクトが見つからない

潜在的な貢献者は、どのプロジェクトが活発で、どのように参加すればよいかわからない。

- [電子フォーラム] ：プロジェクトの特徴をサイトで閲覧してもらう
- [簡単な道筋] ：CONTRIBUTING.mdに貢献方法を明記する

😢 3. 貢献が受け入れられないかもしれない不安

貢献者は、自分の提案が相手のチームの期待に沿わず、受け入れられないかもしれないと不安を抱えている。

- [予備調査] ：プルリクエストの前にイシューで相談してもらう
- [低い位置の果物] ：新規の貢献者向けのタスクをやってもらう

😢 4. 貢献したいプロジェクトの事情がわからない

貢献者は、プロジェクトの技術的な側面やプロセスを理解するのに苦労する。

- [勉強会]：プロジェクトの様子を体験してもらう
- [メンター]：直接相談に乗る（a.k.a. トラステッドコミッター）
- [ちょうど十分]：相手の知識レベルに合わせて段階的にドキュメントなどを用意する（読むものが多すぎると貢献してくれない）

😢 5. 貢献したいのに継続できない

本来の仕事が忙しく、他のプロジェクトに関わっていられない。

- [未来の約束]：いつかまた手伝ってほしいと約束する
- [定期的な連絡]：有望な貢献者には定期的に連絡する
- [空間を演出する]：プロジェクトのことを思い出してもらう

成功したら「物語」として語る🎊

- [体験談の共有]：インナーソースの経験を社内向けに語る
- [感情的なつながり]：論理的な話だけでは人は説得されない

「物語」が終わったら…

- [ふりかえりの時間]：インナーソースの取り組みをふりかえる
- [進化するビジョン]：インナーソースの取り組み自体も見直す

軌道に乗ったら インナーソースパターン も使える！

https://patterns.innersourcecommons.org/ja

📝 「インナーソースを組織に広げる」のまとめ

- インナーソースの導入は「**組織的負債**」の解消につながる
- 組織的負債の根本原因は「**組織文化**」にある
- だが、組織文化そのものを変えることはできない
- Fearless Changeのパターンで「**何をするか**」から変える
- 成功したら**体験談を物語として**共有する

Pythonのドキュメンタリー：OSSのことがわかる!!

Blender：Pythonの人々が「ナイス」だったから

全体のまとめとメッセージ💌

- **ソフトウェア開発は人々のことである**
- 人々は「ナイス」でなければならない（クソ野郎ではダメ）
- パターンを使って「**何をするか**」から変えよう
- コード以外にも貢献できることはある

> 仮面ライダーオーズ/OOO「**手が届くのに手を伸ばさなかったら死ぬほど後悔する それが嫌だから手を伸ばすんだ それだけ**」

インナーソースで未来を築こう!!

このあといろいろな事例が聞けると思います😊