2025.9.25

OctoNihon Forum #3

InnerSource x GitHub が拓く未来

三菱電機株式会社　設計技術開発センター
オープンソース共創推進部(OSPO/ISPO) 部長
追立 真吾

三菱電機株式会社　名古屋製作所
ネットワークソフトウェア開発課　主任
上野 稔晃

MITSUBISHI ELECTRIC
Changes for the Better
serendie

自己紹介

追立　真吾
（おいだて　しんご）

LinkedIn:@shingo-oidate-735328189

FY2023～FY2024
三菱電機のR&D部門における
オープンソースコミュニティ活動を推進。

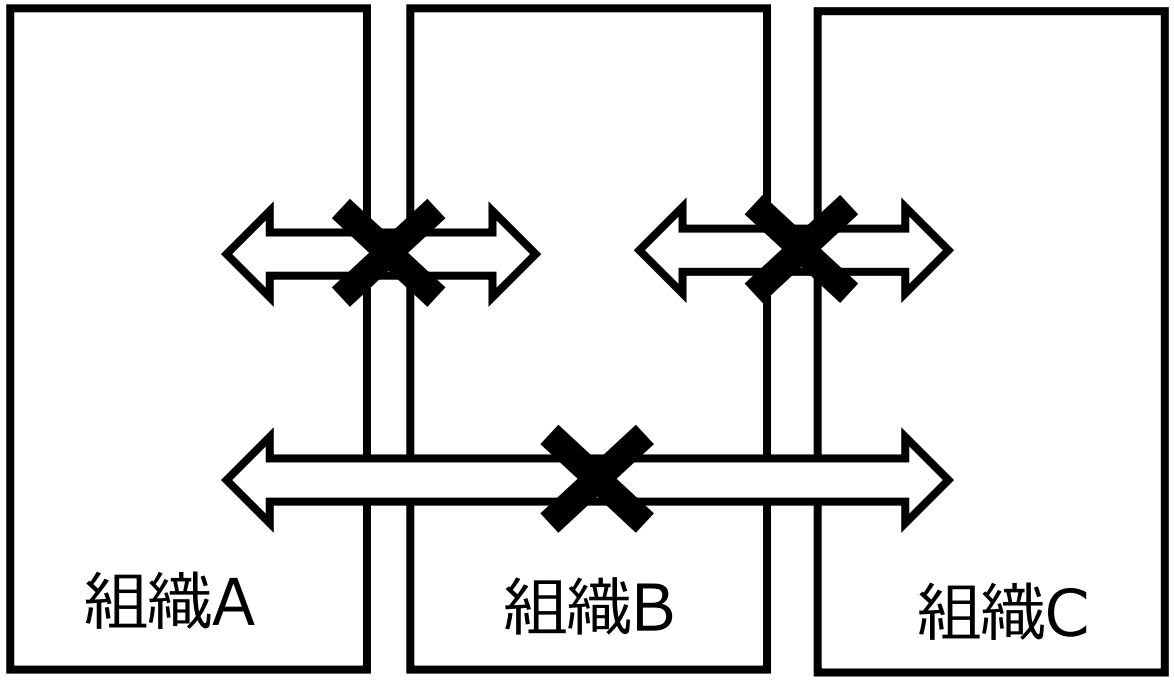
FY2025～現在
全社のオープンソース＆インナーソース推進組織
（OSPO/ISPO）の責任者。
GitHub Enterprise管理者。

今後の発表予定（是非、ご参加ください！）
【10/08】　RedHat Summit: Connect 2025 Tokyo
【11/13】　InnerSource Summit 2025（at 三菱電機）
【12/08-10】Open Source Summit Japan 2025

インナーソース（＝社内オープンソース）で解決したい課題

DX化を阻む「サイロ化された組織」

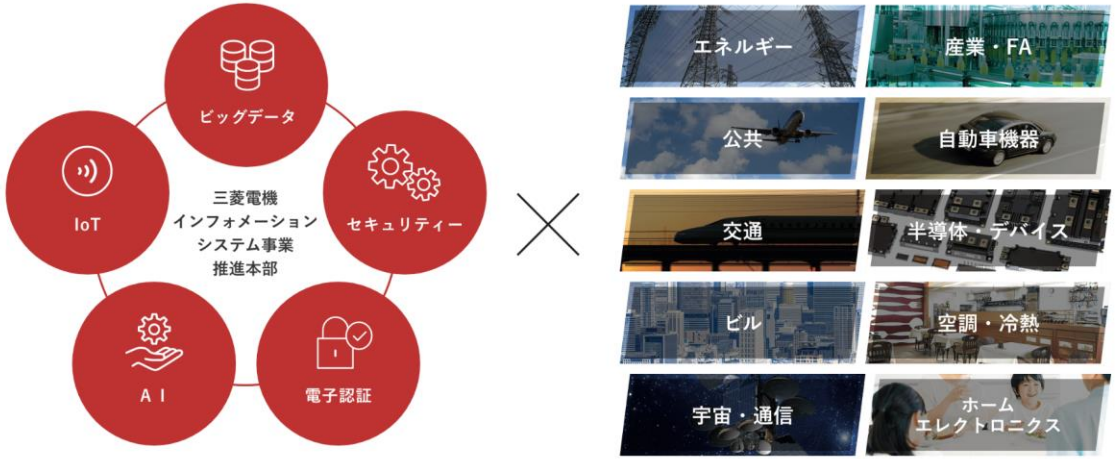
- 社内の各組織が独立して業務を行っており、個別最適化されている状態
- 社内の他組織のことを知らないし、知る必要もなく、自分たちのことだけ考えていけばいい状態



三菱電機が目指すDX

総合電機メーカーならではの強みを、さらなるものに

総合電機メーカーならではの、持てる「総合力」を掛け算すれば、課題の解決を、よりよい社会づくりを加速できます。三菱電機は、そんな思いのもと、ITソリューションを進化させていきます。素敵な明日を胸に描いて。

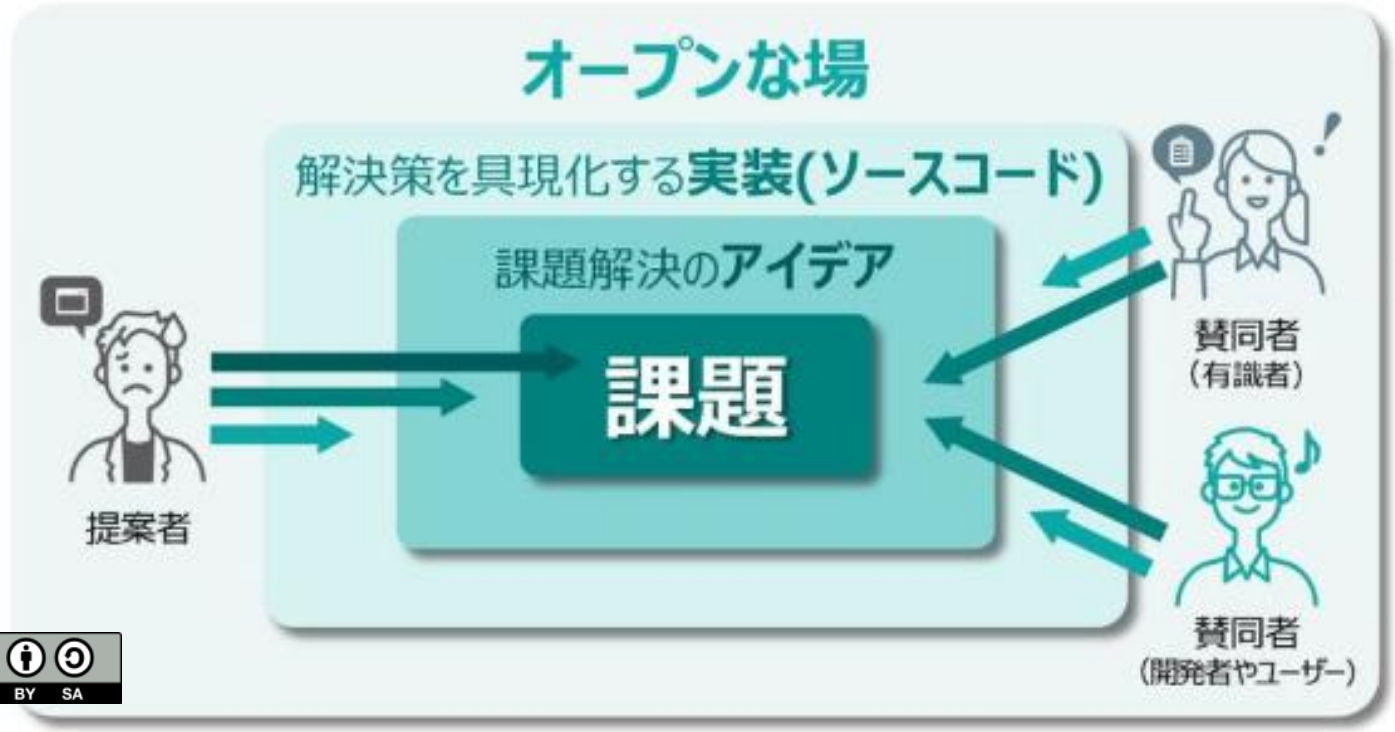


組織のサイロを破壊し、<u>コングロマリット・プレミアムを実現</u>したい

当社インナーソースコミュニティが目指す姿

組織間の壁を超えて、重要な課題を、最適な人財が、タイムリーに解決

- 個別の利益だけでなく全社の利益を志向できる状態
- 社内フルオープンな場で、課題を共有・発見し、意見交換によりアイディアを創出し、協働で課題を解決できる状態(**ソースコードの共通化のみを目的としない**)



GitHub Enterpriseで
課題(Issues)、解決策(Pull Request)、
実装(Code)といった社内知見を集約

GitHub Copilotで
集約した社内知見を有効活用

※図の引用元:小林 良岳,「インナーソースで始める組織内オープンソース開発入門」, InnerSource Commons Japan, 2023/01/17,
https://speakerdeck.com/yuhattor/innersource-learning-path-japanese?slide=80

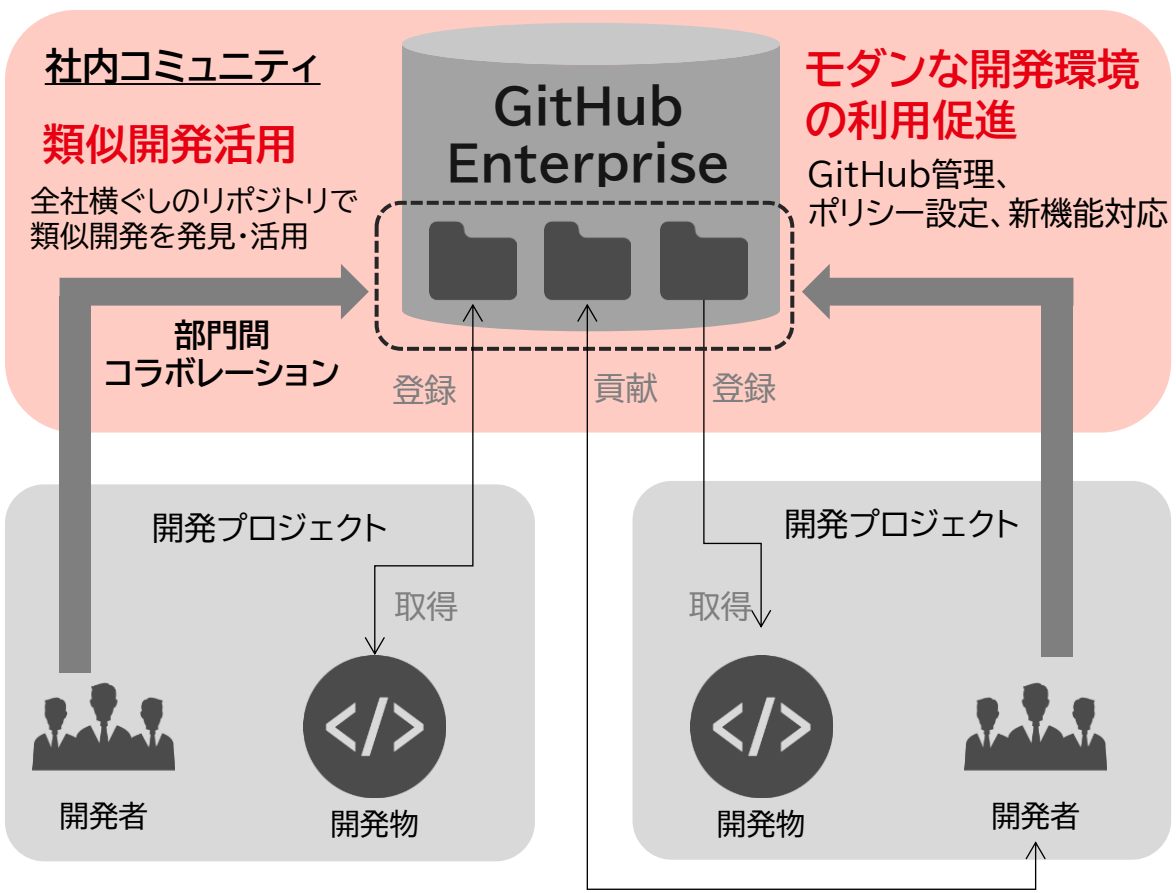
三菱電機OSPO/ISPO(Open Source & InnerSource Program Office) の役割

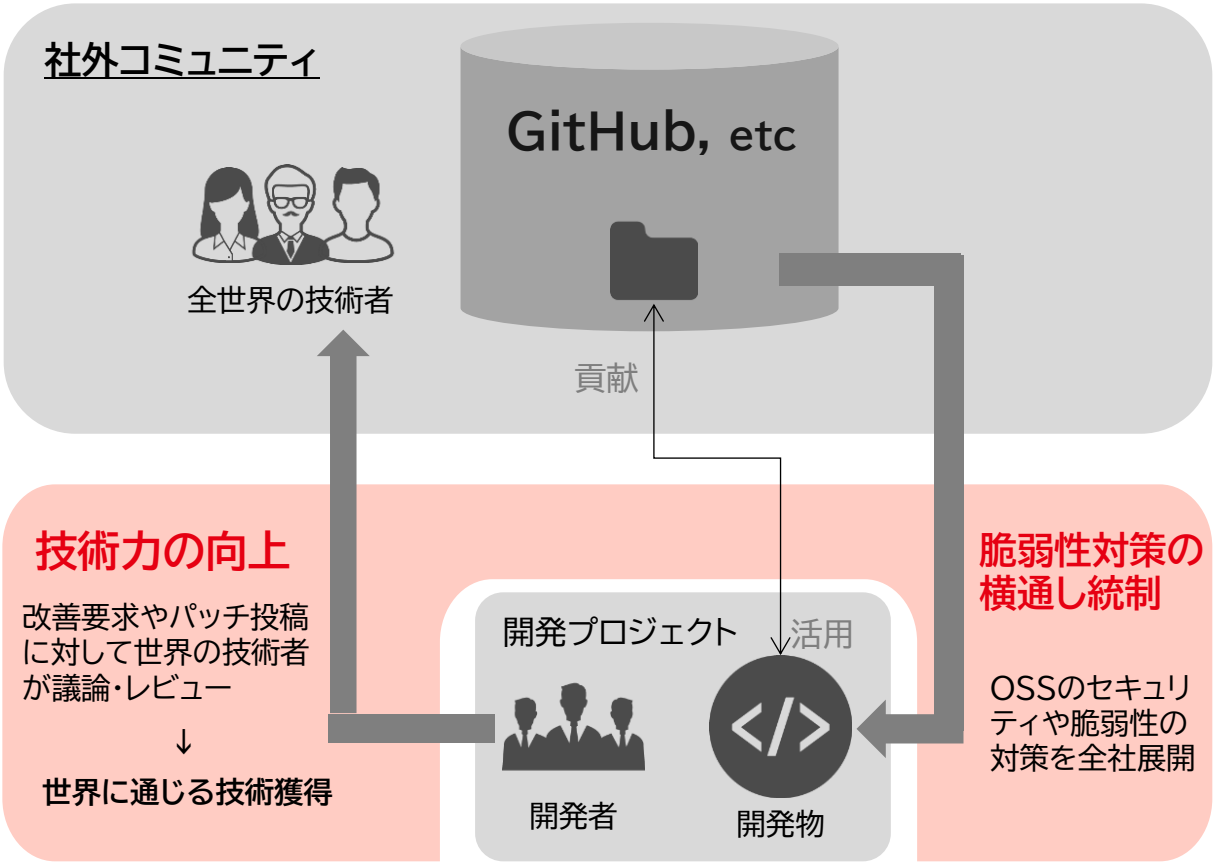
1. インナーソース推進(ISPO)

- 製作所開発SWやOSSを社内リポジトリ上で開発・格納
- 透明性を確保し、設計ノウハウを含めて共有
- 部門間コラボレーションを促進

2. オープンソース推進(OSPO)

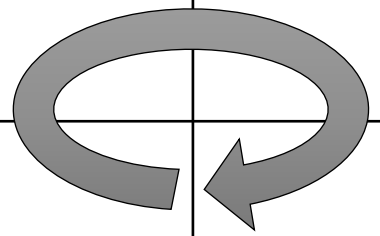
- OSSコミュニティからの先進技術獲得と全社展開
- OSSのセキュリティ/脆弱性対策の展開の仕組みを構築
- 最新OSS技術の評価と適用ノウハウの蓄積





インナーソース x オープンソース による相乗効果

		狙い	
		人財価値向上	製品価値/企業価値向上
活動範囲	インナーソースコミュニティ	・オープンな共創文化醸成 ・Developer Experience向上	・共創による事業間シナジー ・車輪の再発明防止による開発効率化
	オープンソースコミュニティ	・先進技術の獲得 ・共創ノウハウ獲得	・企業としての社会的責任の遂行 ・協調領域(オープンソース)と差別化領域との親和性向上による間口拡大



※参考資料

服部 佑樹, 「エンジニアの共創は結局大企業では無理なのだろうか いや、むりでない」, InnerSource Commons Japan, 2023/09/06, https://speakerdeck.com/yuhattor/developer-experience-and-innersource-ensinianogong-chuang-hajie-ju-da-qi-ye-tehawu-li-nanotarouka

三菱電機のSerendie®　～社内の組織文化の変革～

Serendie®(セレンディ)は、
データ活用を通じて事業横断型のサービスを創出するためのデジタル基盤です。

「思いがけない発見」や「偶然がもたらす幸運」を意味するSerendipity(セレンディピティ)と、Digital Engineering(デジタルエンジニアリング)を掛け合わせて名付けられました。

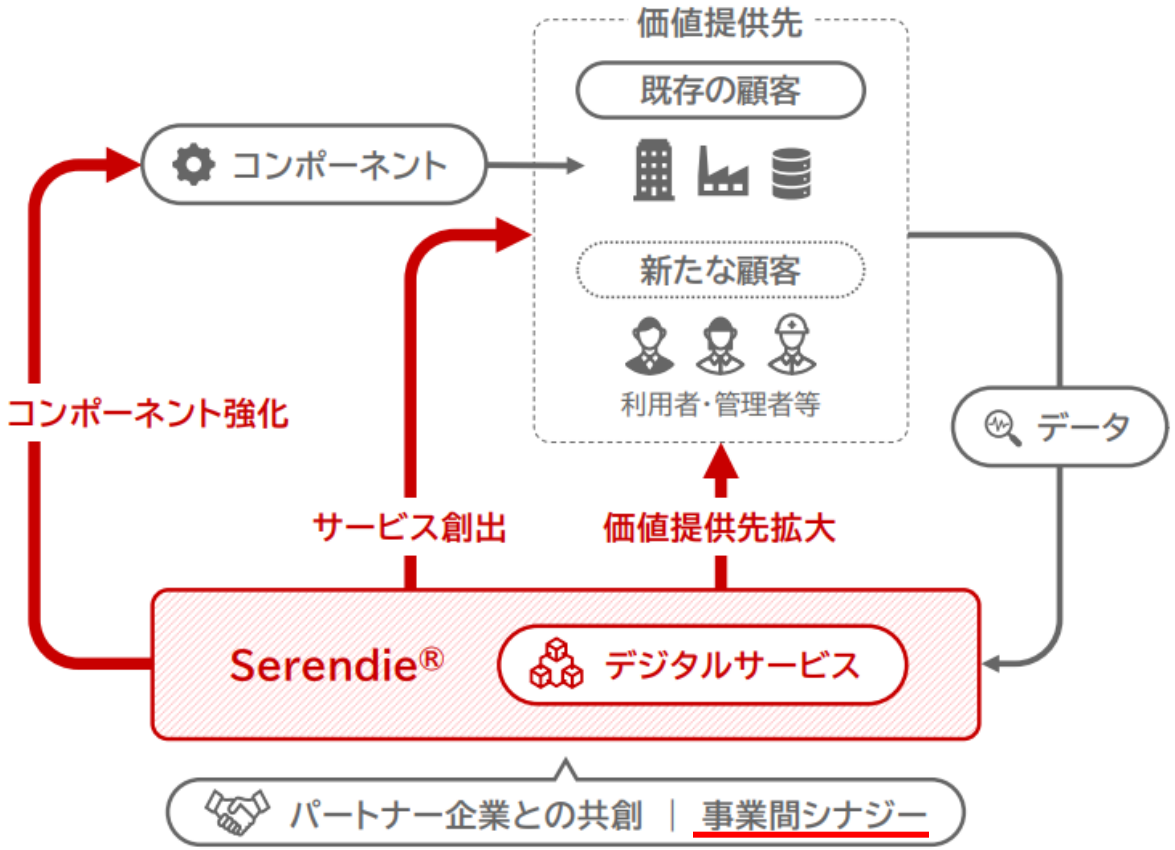
家庭から宇宙まで、あらゆる領域から集めたデータ。
最先端の技術力とプロフェッショナルの創造力。

それらを掛け合わせ、**お客様やパートナーの皆様と共に**、
アジャイルに、持続的に、新たな価値を生み出します。

オープンに価値共創を目指す全社方針が
インナーソース x オープンソース推進の追い風に！

Serendieによるビジネスモデルの変革

コンポーネントを利用することで生まれるデータを起点にビジネスモデルの変革を推進



事業間シナジー

ビル×空調×電力

- スマートビルソリューション
  （快適空間、安全・安心、ロボット配送）

交通×電力

- 鉄道向けソリューション
  （列車運行、電力最適化）

FA×電力

- 工場向けソリューション
  （カーボンニュートラル、SCM*最適化）

InnerSource x GitHub で事業間シナジーを実現！

共創スペース:Serendie Street Yokohama

InnerSource Summit 2025

日時:2025年11月13日(木)
場所:Serendie Street Yokohama

オープンソースの学びを社内に適用する「インナーソース」の国際カンファレンスにぜひご参加ください！
InnerSource Commons 10周年を記念し、横浜(三菱電機) - ベルリン(SAP) - ニューヨーク(IBM) で24時間リレー形式で実施いたします。

Special Session

伊藤かつら
人事院 人事官

「未来を築く技術者と組織の変革力とは？」
IBMやMicrosoft でのリーダーシップのご経験から、未来を見据えた技術者に求められる思考力と行動力、そして組織がそれらを育成・活用するための仕組みづくり。技術者と組織が相互に成長し合う関係性の構築について議論します。

Featured Sessions

Yuki Hattori
President
InnerSource Commons Foundation

Yusuke Shijiki
Executive Officer, Vice President,
Corporate Manufacturing and Engineering
Mitsubishi Electric

Shingo Oidate
GM, OSPO/ISPO, Mitsubishi Electric
Nogi Kazuma
Head Resarcher, Mitsubishi Electric

Shane Coughlan
GM, OpenChain
The Linux Foundation

Nitish Tyagi
Principal Analyst
Gartner

Zack Koppert
Sr. Engineering Manager
GitHub

Jerry Tan
Executive Vice Secretary-General
China Open Source Promotion Union

Tomohiro Nakajima
Product Owner Lead
KDDI Agile Development Center

Mishari Mauqbil
CEO
Zymple

Younes Hairej
Founder & CEO
Aokumo Inc.

Yoshitake Kobayashi
Senior Manager
Toshiba

Chen Wei
Huawei

* 英語セッションにはAIによる翻訳をご用意致します。
* 掲載しているセッションは横浜開催(日本タイムゾーン) のものになります。

innersourcecommons.org

Register Today!

bit.ly/ISC-Yokohama

Ticket Types

会場参加
Free

Tシャツ +
Global Access
$35

Operated by

InnerSource Commons

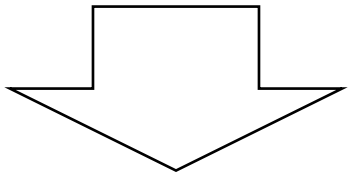
MITSUBISHI ELECTRIC
Changes for the Better

2025年
11月13日(木)
10:00 ~20:00
(9:15開場)

インナーソースに投げかけられる懸念と回答

懸念

- オープンソースに比べて参加者の母数が圧倒的に小さいよね？
- 過去に共通ライブラリや共通プラットフォームは苦労したよね？
- 全く異なる事業間でソフトウェア共有して価値あるの？
- 初期に参画して何かメリットあるの？



回答

- 当社インナーソースでは、他部署メンバからの貢献や幅広いソースコード流用を第一の目的としない
- 社内フルオープンな環境(**GitHub Enterprise**)に知見(課題、解決策、ソースコード)を集約し、親和性の高いAI(**GitHub Copilot**)を用いて認知負荷/学習コストを削減することで、全社横断の知見共有によって、繋がるべき人たちを繋げ、共創を促進することが第一の目的
- GitHub環境の運営は全社まとめてISPOが実施するので、初期の参画者にもメリットがある

上記以外にも様々な懸念が投げかけられ、ひとつひとつに回答していきました
是非、ネットワーキングタイムにもディスカッションさせてください！
(InnerSource Summitでも詳細を発表予定です)

インナーソース参加者にコミュニティの一員となってもらうための言葉

シャープ元副社長 佐々木正氏 の言葉

いいかい、君たち。分からなければ聞けばいい。

持っていないなら借りればいい。

逆に聞かれたら教えるべきだし、持っているものは与えるべきだ。

人間、一人でできることなど高が知れている。

技術の世界はみんなで共に創る**『共創』**が肝心だ。

※参考文献：大塚 正之, “「共創とは何か」”, 共創学 2019, Vol.1(1)
https://nihon-kyousou.jp/cocreationology/vol1_no1/Cocreationology_1-1-11.pdf?utm_source=chatgpt.com

GitHub活用事例のご紹介

三菱電機株式会社　名古屋製作所
ネットワークソフトウェア開発課　主任（プロジェクトマネージャー）
上野 稔晃

MITSUBISHI ELECTRIC
Changes for the Better

目次

- 自己紹介
- GitHub活用の成功事例
- GitHubに至った背景
  - Tips: 新環境を職場に導入出来た理由
- GitHubでやっている事
  - Tips: 自作バーンチャートを使う理由
- サマリ

自己紹介

自己紹介

FY2016
- 入社

FY2019
- SW設計開発へ転向

FY2025 (現在)
- SW設計開発を継続

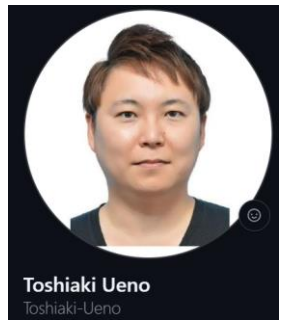
3年
HW設計開発
（回路設計）

6年
SW設計開発
（組込みSW）

名古屋製作所

Factory Automation製品
(MELSEC iQ-Rシリーズ)

GitHubのアカウント



GitHub活用の成功事例

GitHub活用の成功事例

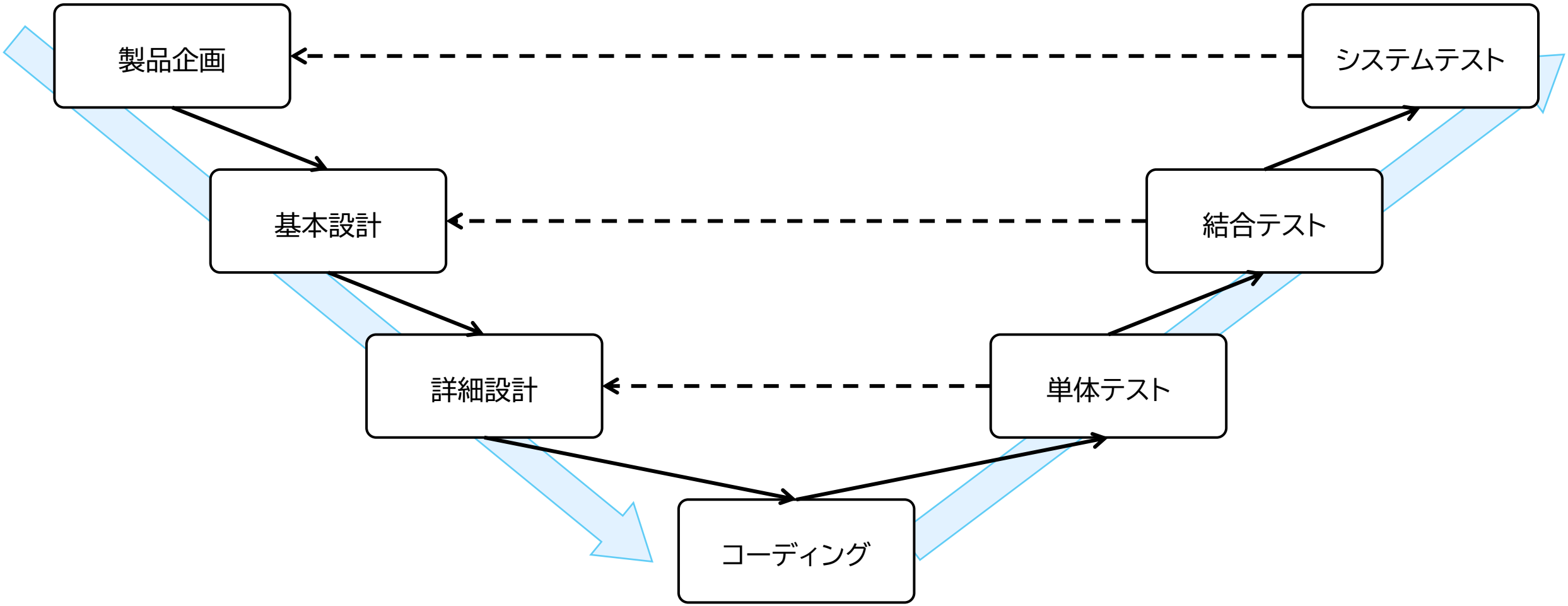
過去いろいろな開発環境を試してきたが、**組込みSW開発でも、現時点では GitHub が最適解**

- 実績
  - 過去の開発で、18カ月 ⇒ 14カ月と、開発期間を22%短縮したが、GitHubが持っている機能を使わないと達成不可能だった。
- 理由
  - **GitHubは、SW開発プロジェクト管理に必要な機能をデフォルトで全搭載している**
  - **そのため、【タスク管理】⇔【ソースコード管理】や、【ドキュメント作成】⇔【CI/CD】連携が容易**
- 補足
  - （現在であれば）【ドキュメント】⇔【ソースコード】の双方を生成AI（Copilot）に食わせて解析やレビューが可能
  - （現在であれば）IssueにCopilot Agentをアサインして、作業をお任せ可能
  - （弊社内であれば）**環境の運営コストを、ISPOが一元管理することで最適化している**

名古屋製作所の開発手法 ～ V字モデル開発 ～

名古屋製作所の開発手法 ～ V字モデル開発 ～

アジャイル / ウォーターフォール共通で、V字モデルに準じた開発



GitHubに至った経緯 ～ なぜGitHubを選んだのか ～

SW開発プロジェクトで必要なシステム

➢ SW開発をする上で、必要なシステム

必須

推奨

タスク・進捗管理
- Redmine
- Jira
- Asana
- Backlog
- Excel

ドキュメント作成・管理
- ファイルサーバ + Word, Excel
- Confluence
- Notion

ソースコード管理
- Git
- Subversion（SVN）

or

- 管理しない（Strong Style）

CI/CD
- Jenkins
- 自前のスクリプト

or

- CI/CD環境を組まない

GitHubの機能

➢ GitHubは、全要素が統合された開発環境＋αを提供

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

＋

生成AI
- GitHub Copilot

and More…
- GitHub Codespaces

SW開発プロジェクト管理ツール遍歴(タスク進捗・管理)

2019年
- Excelガントチャート
- Excelマクロ

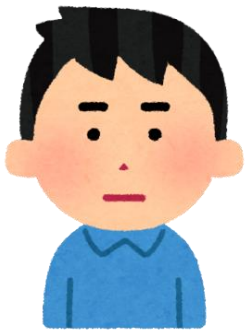
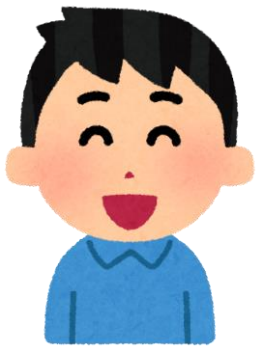
2019年
- Redmineサーバ立上げ
- ガントチャート使用

2020年
- Redmineにカンバンチャート導入
- EVMも追加

2022年
- GitLabサーバ立上げ

2025年
- InnerSource参画
- GitHubへ移行



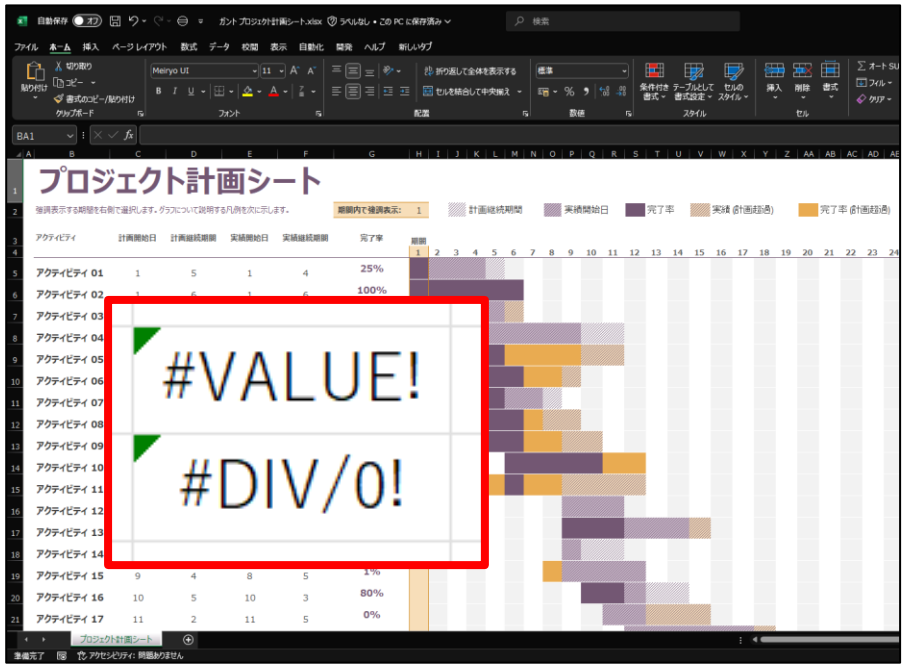




Excelガントチャート + Excelマクロ

【2019年】Excelガントチャート、Excelマクロによるプロマネが始まる。

しかし・・・、辛い・・・。

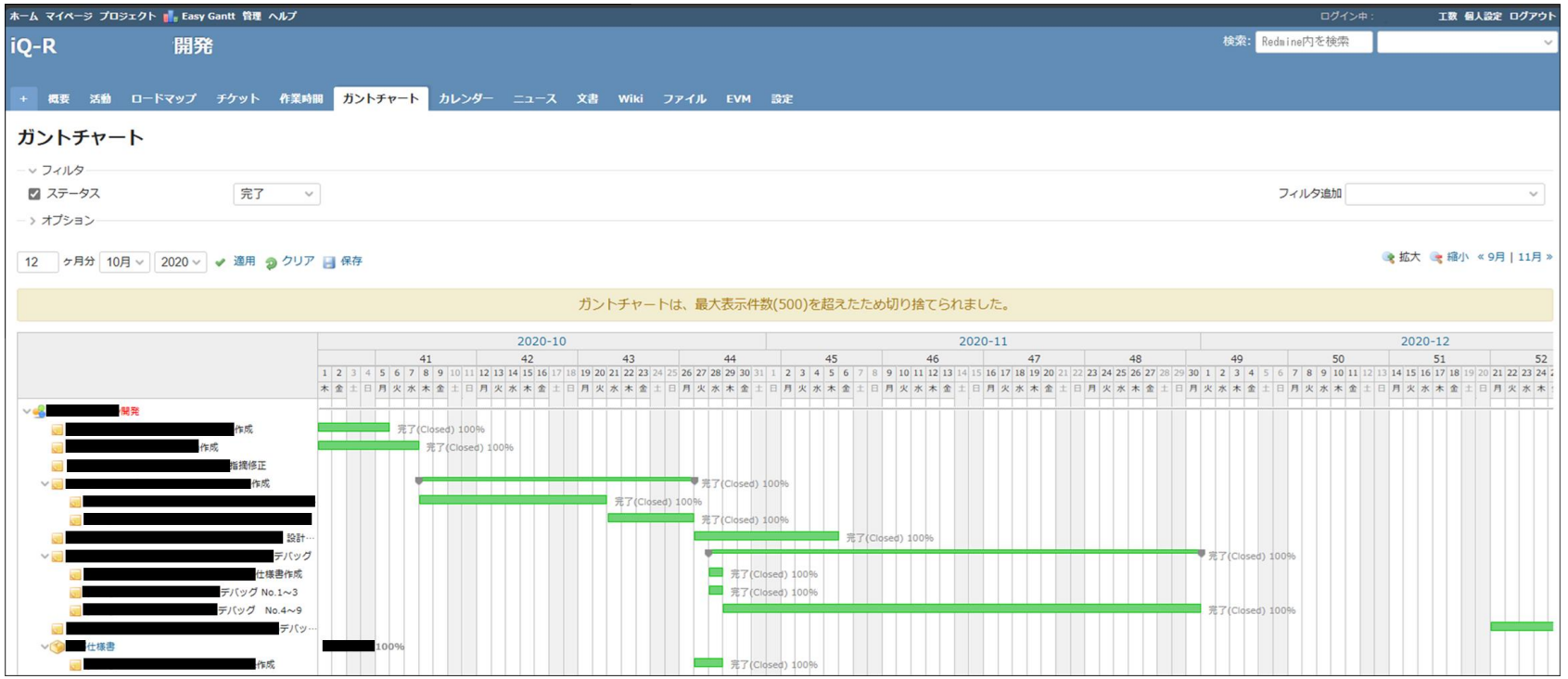
- いつの間にか、誰かの手によって数式が崩壊している。
- ちょっとした変更(ファイル名変更など)でマクロがエラーを吐く。
- 具体的な作業内容を記述出来る場所がない。



Redmineでガントチャート

【2019年】Redmineサーバ立上げ ⇒ チケット起票形式 + ガントチャートで可視化

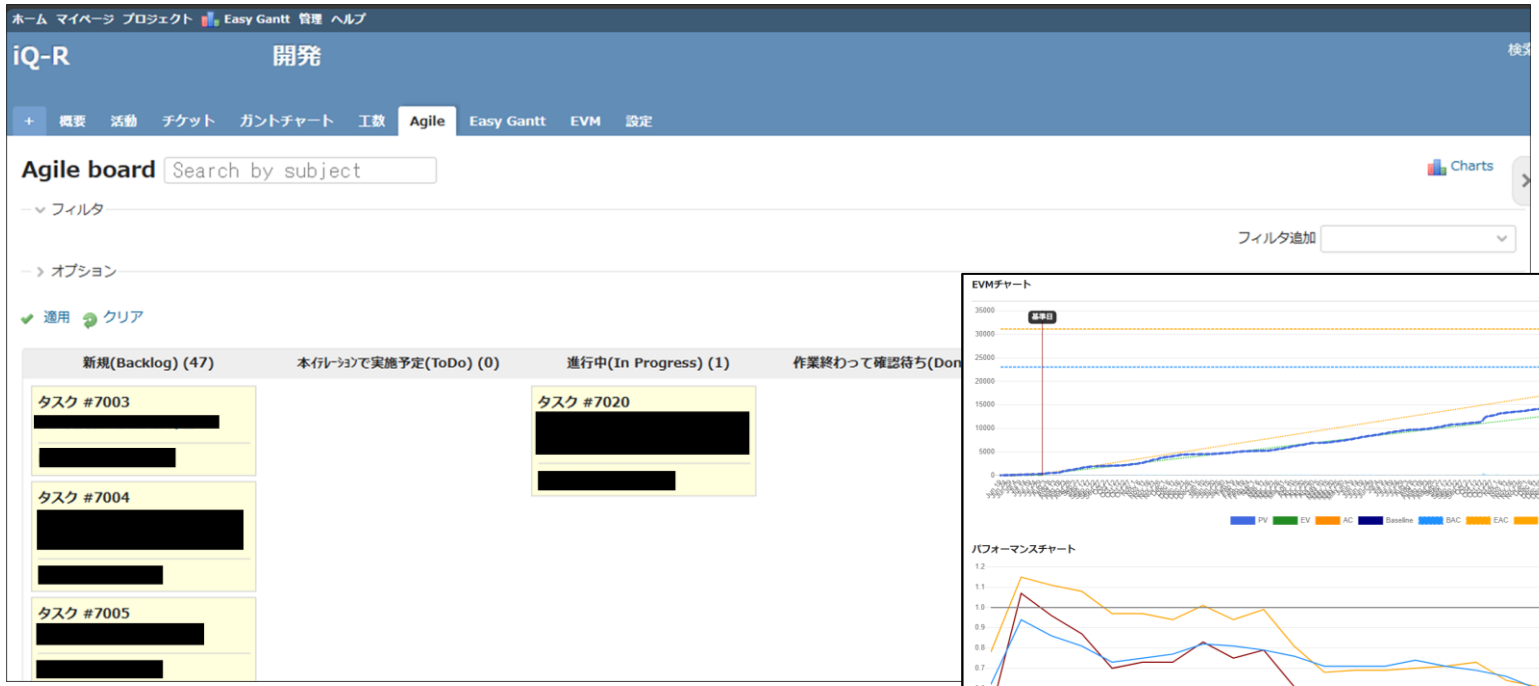
- Excelよりマシにはなったが・・・、それでも辛い。
  - 遅延が発生した場合のリスケがとにかく面倒。後続のタスクを軒並みずらす作業が発生する。
  - 「いつこのプロジェクトが終わるのか？」が不透明。ズルズル遅れた場合、「いつ終わるの？」が計算出来ない。

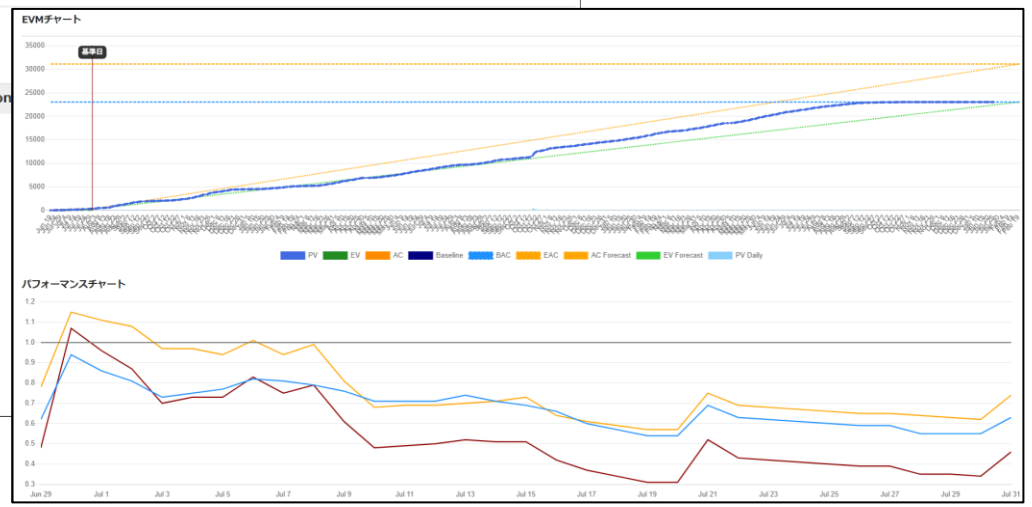


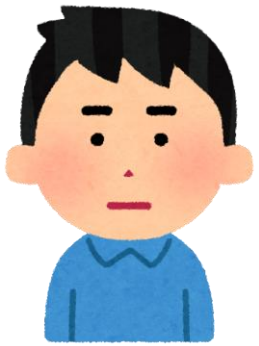
Redmine + カンバンチャート + EVM を導入

【2020年】Redmineにカンバンチャート + EVMプラグインを追加

- 悪くはない。
  - EVMでプロジェクト進行速度(Velocity)が見える。
  - タスクの割当てもドラッグアンドドロップで楽になった。







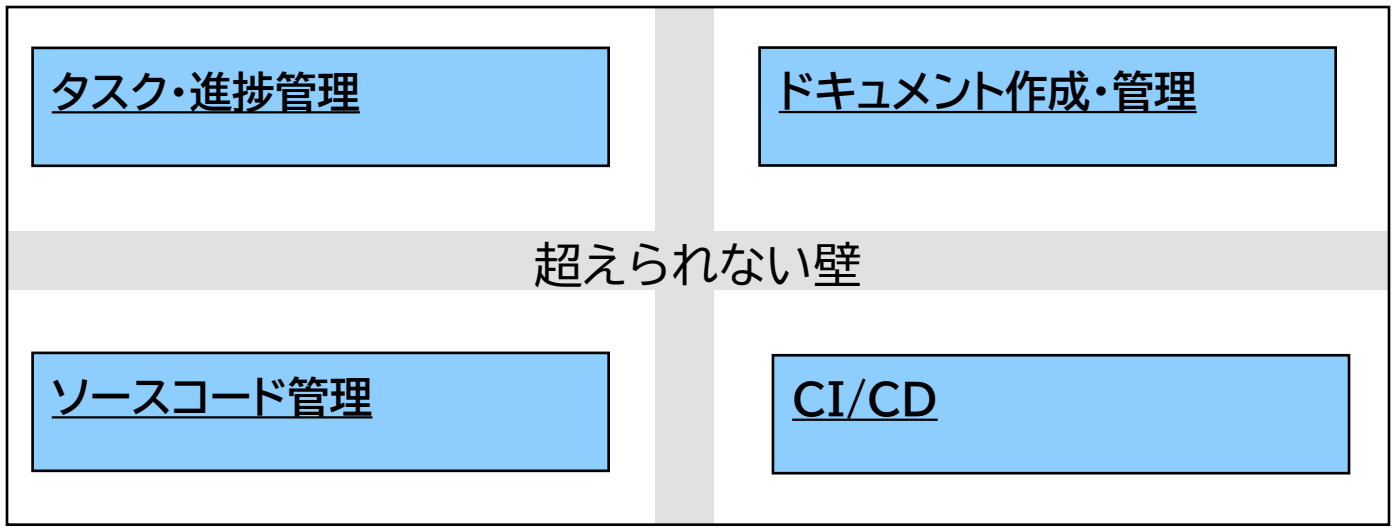
Redmine + カンバンチャート + EVM の課題

Redmineにカンバンチャート + EVMプラグインを入れた環境は、プロマネとしては悪くはない・・・。

でも**まだ、課題(以下)が残る**。

- タスクを手動でクローズするのが面倒くさい
- どのタスクが、どのドキュメント・ソースの変更と関連しているのか追えない
- CI/CD環境が無いので、繰り返し作業も手作業に

⇒ **各システム間の連携が取れていない**

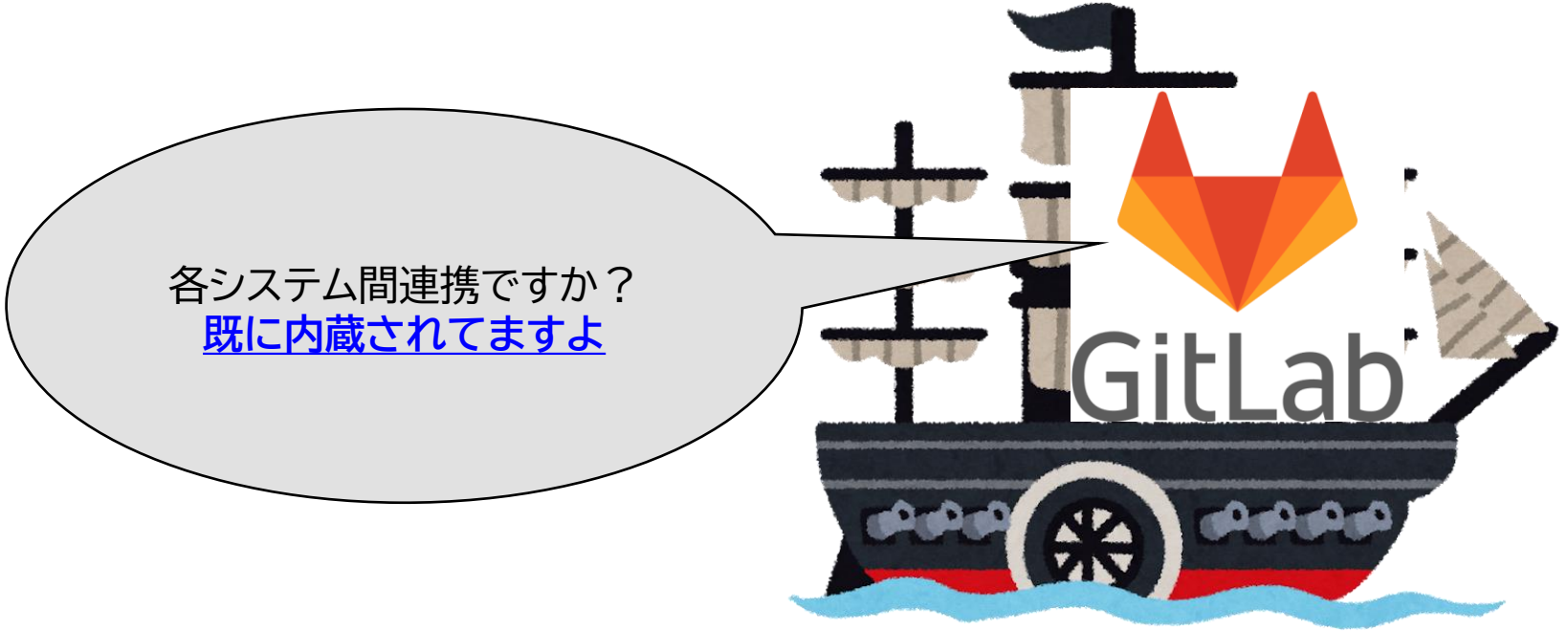


GitLab到来 ～ 鎖国状態だった各システムが開国 ～

【2022年】GitLab導入。鎖国状態だった各システムが開国した。

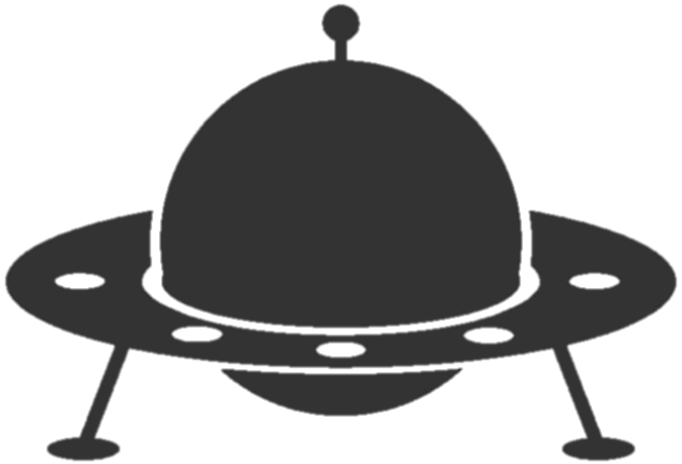
残っていた以下の課題を、GitLabが払拭した。

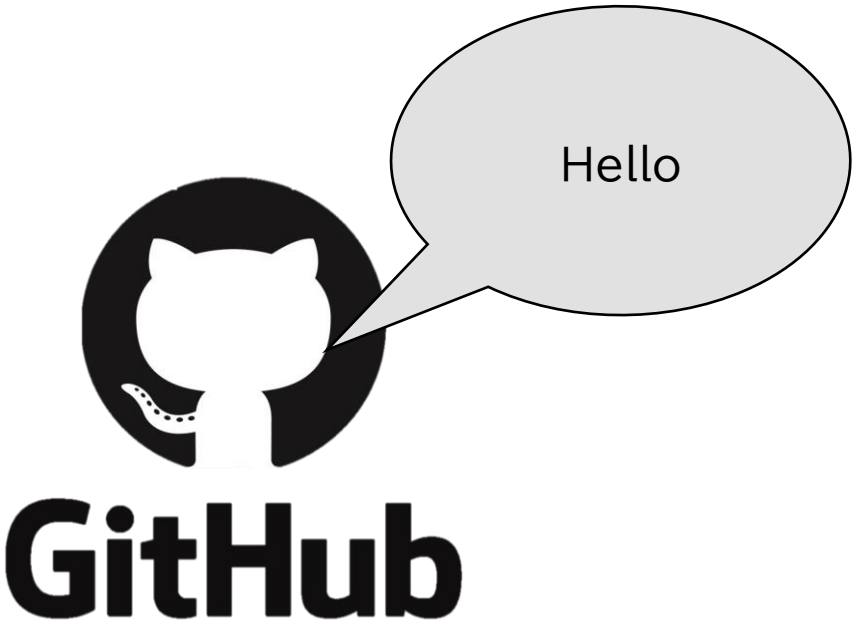
- ~~タスクを手動でクローズするのが面倒くさい~~
- ~~どのタスクが、どのドキュメント・ソースの変更と関連しているのか追えない~~
- ~~CI/CD環境が無いので、繰り返し作業も手作業に~~



【2025年】GitHubとの遭遇

- InnerSourceコミュニティが、環境をメンテしてくれている（なのでメンテナンスフリー）
- InnerSourceコミュニティの、生成AIスペシャリストとのコラボが期待できる
  - 生成AIが自動でタスクをこなしてくれたり・・・？
  - 生成AIが自動でコードレビューしてくれたり・・・？
- **やれる事が増えた！！**





TIPS: 新環境を職場に導入出来た理由

とりあえずやってみて、それが成功した

⇒ 成功した結果を上長に報告したら、認められた

GitHubでやっている事 ～ 実運用の紹介 ～

GitHubの機能

➢ GitHubは、全要素を搭載済み

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

＋

生成AI
- GitHub Copilot

and More…
- GitHub Codespaces

GitHubのタスク・進捗管理システム

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

＋

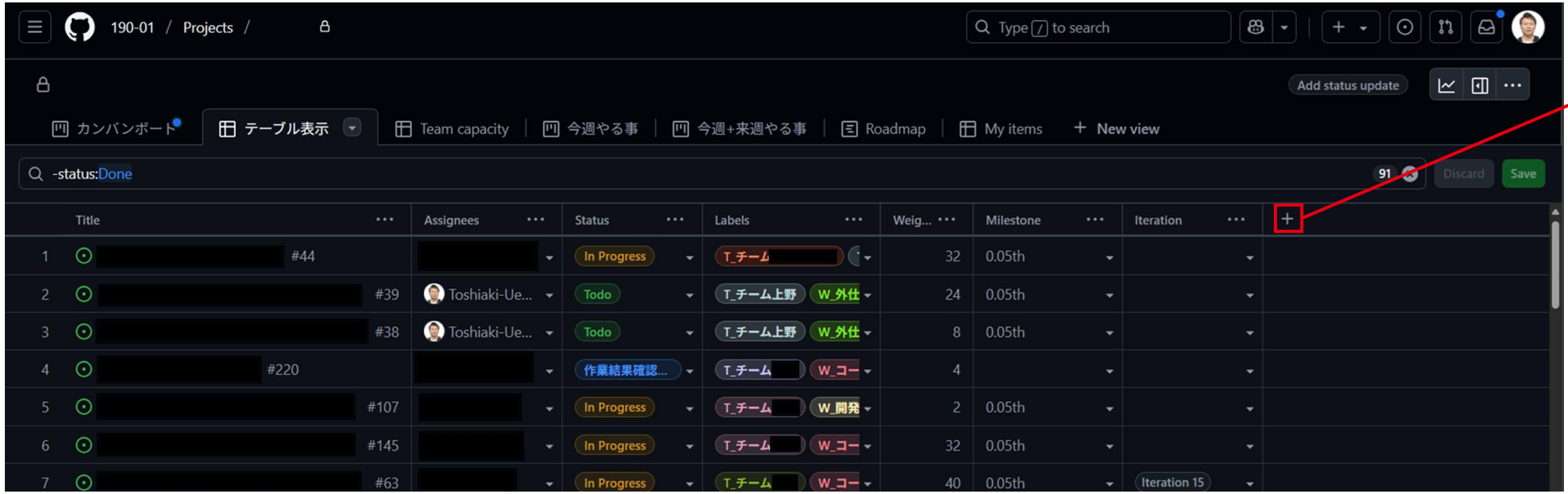
生成AI
- GitHub Copilot

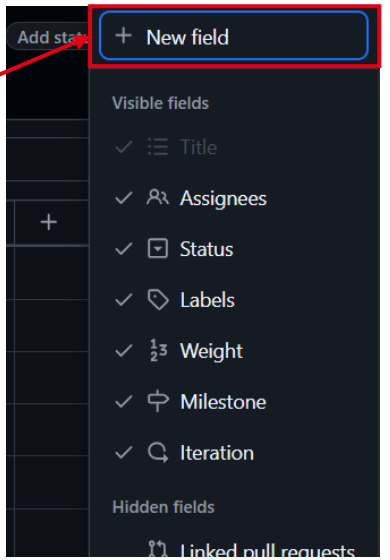
and More…
- GitHub Codespaces

タスク管理 ～ チケット起票型タスク管理 ～

GitHubは、チケット起票型のタスク管理システム

- 項目のカスタム化も可能
- Issue一覧のテーブル表示など、表示の仕方も融通が効く





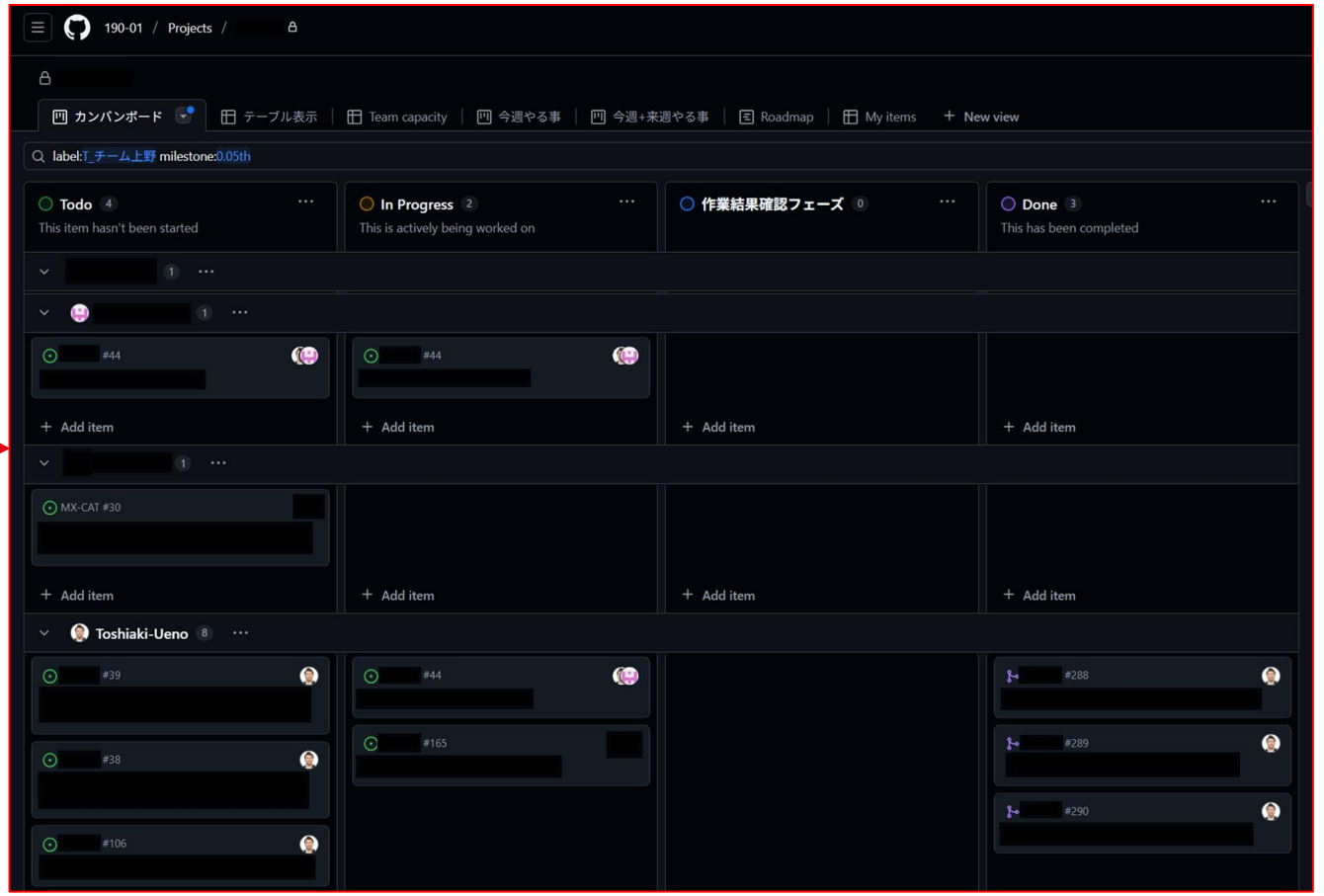
タスク管理 ～ カンバンチャート ～

タスク管理はカンバンチャートを使用

タスクボード

各チームのタスクです。

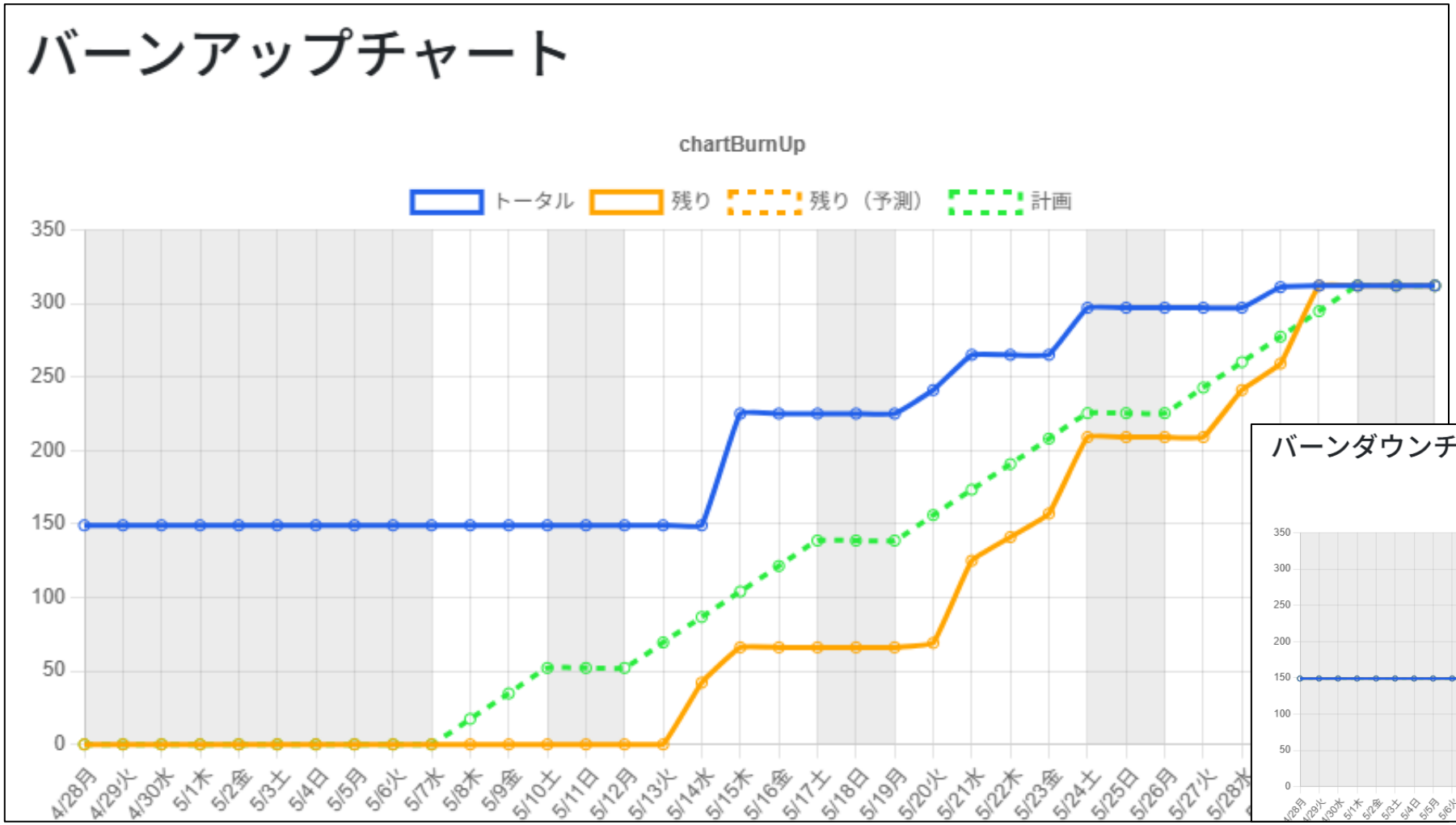
	全体	チーム上野	チーム平	チーム内	チーム佐	チーム衣	チームマイクロ
今週やる事	リンク	リンク	リンク	リンク	リンク	リンク	リンク
今週+来週やる事	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.01th (2月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.02th (3月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.03th (4月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.04th (5月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.05th (6月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.06th (7月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.07th (8月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.08th (9月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.09th (10月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.10th (11月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.11th (12月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
0.12th (1月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
1.00th (2月)	リンク	リンク	リンク	リンク	リンク	リンク	リンク
全アイテム	リンク	リンク	リンク	リンク	リンク	リンク	リンク



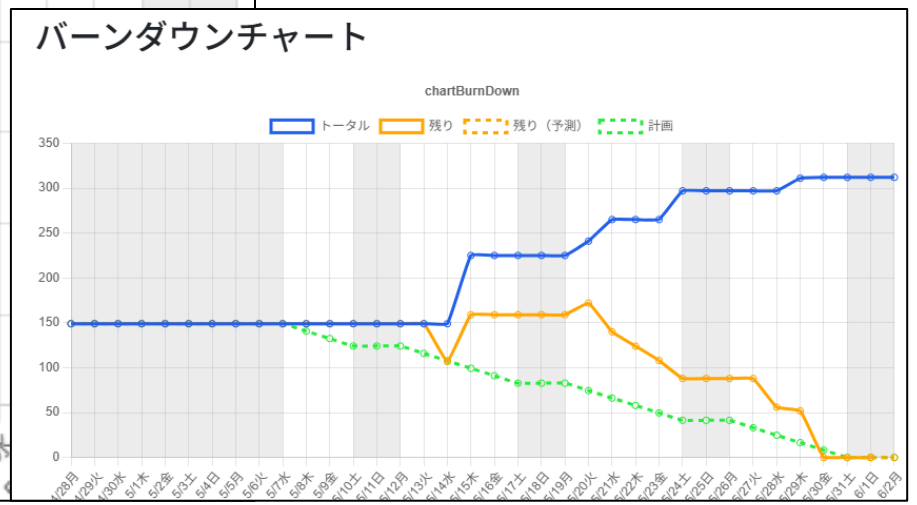
進捗管理 ～ バーンチャート ～

進捗管理はバーンチャートを使用（GitHub QraphQL APIを使用し、HTML + JavaScriptで描画）

バーンアップチャート



バーンダウンチャート



Tips: なぜバーンチャートを自作したのか？

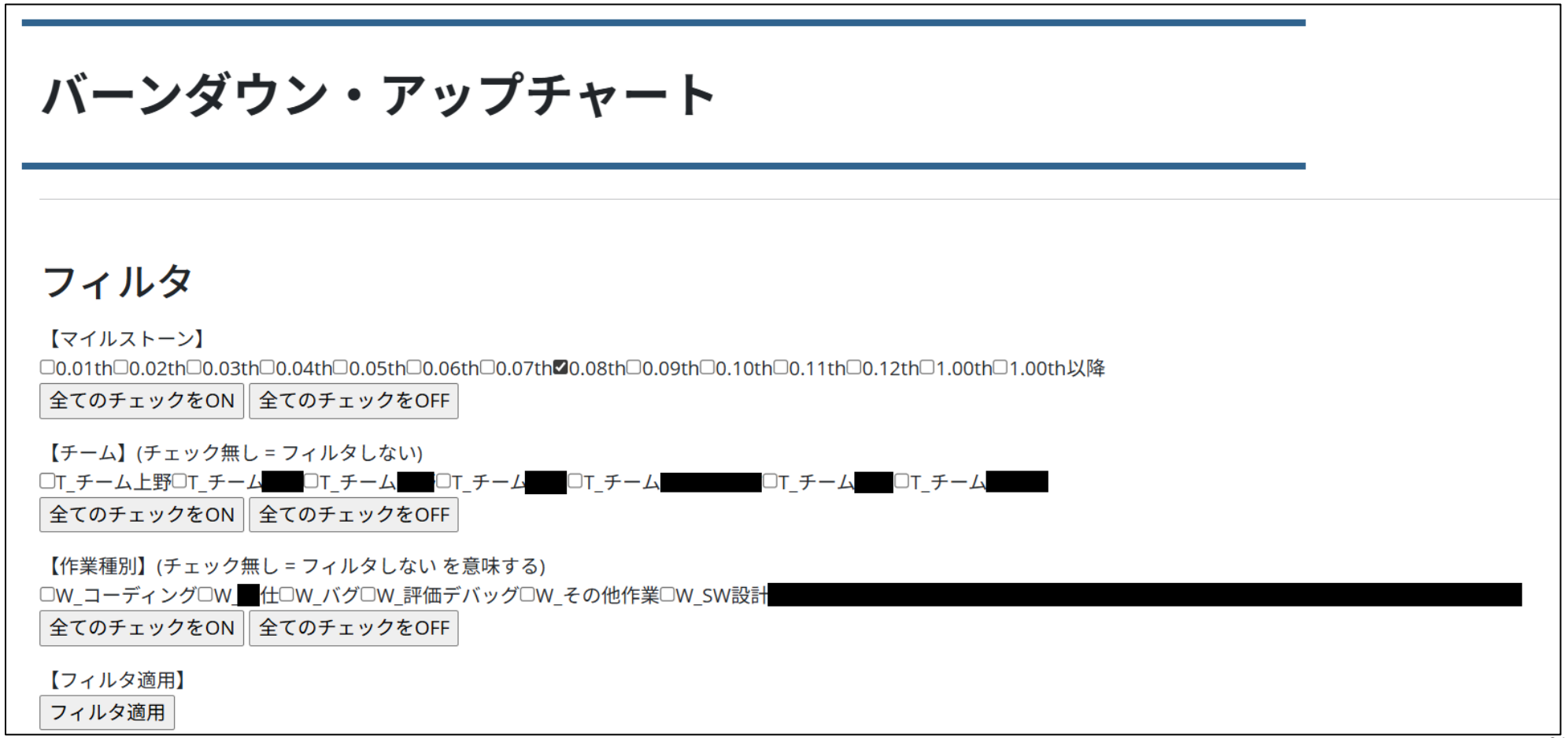
以下の機能が欲しかった

- 柔軟なフィルタ機能（動的なフィルタ項目の生成）
- 会社休日の設定
- 実績から計算した完了予定日の算出（実績と計画の差分）
- 各チームのWeight円グラフ、密度の表示
- 各担当者のWeight消化状態の表示

Tips: なぜバーンチャートを自作したのか？

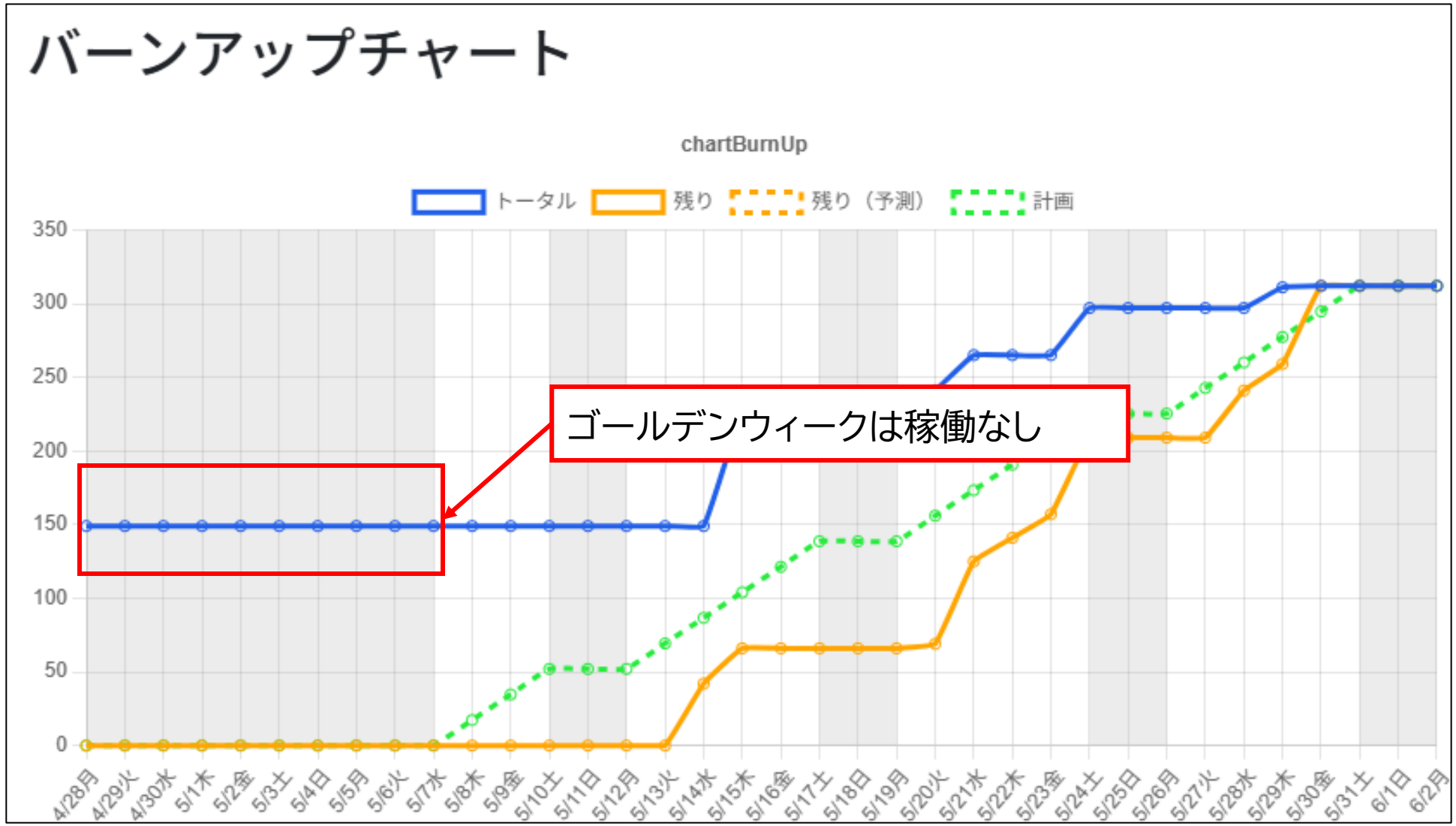
柔軟なフィルタ機能(動的なフィルタ項目の生成)

- ラベル一覧を動的に取得 ⇒ Prefixでフィルタジャンルを仕分け(“T_” ならチーム、”W_”なら作業種別)
- マイルストーン一覧も、リポジトリから動的に取得



Tips: なぜバーンチャートを自作したのか？

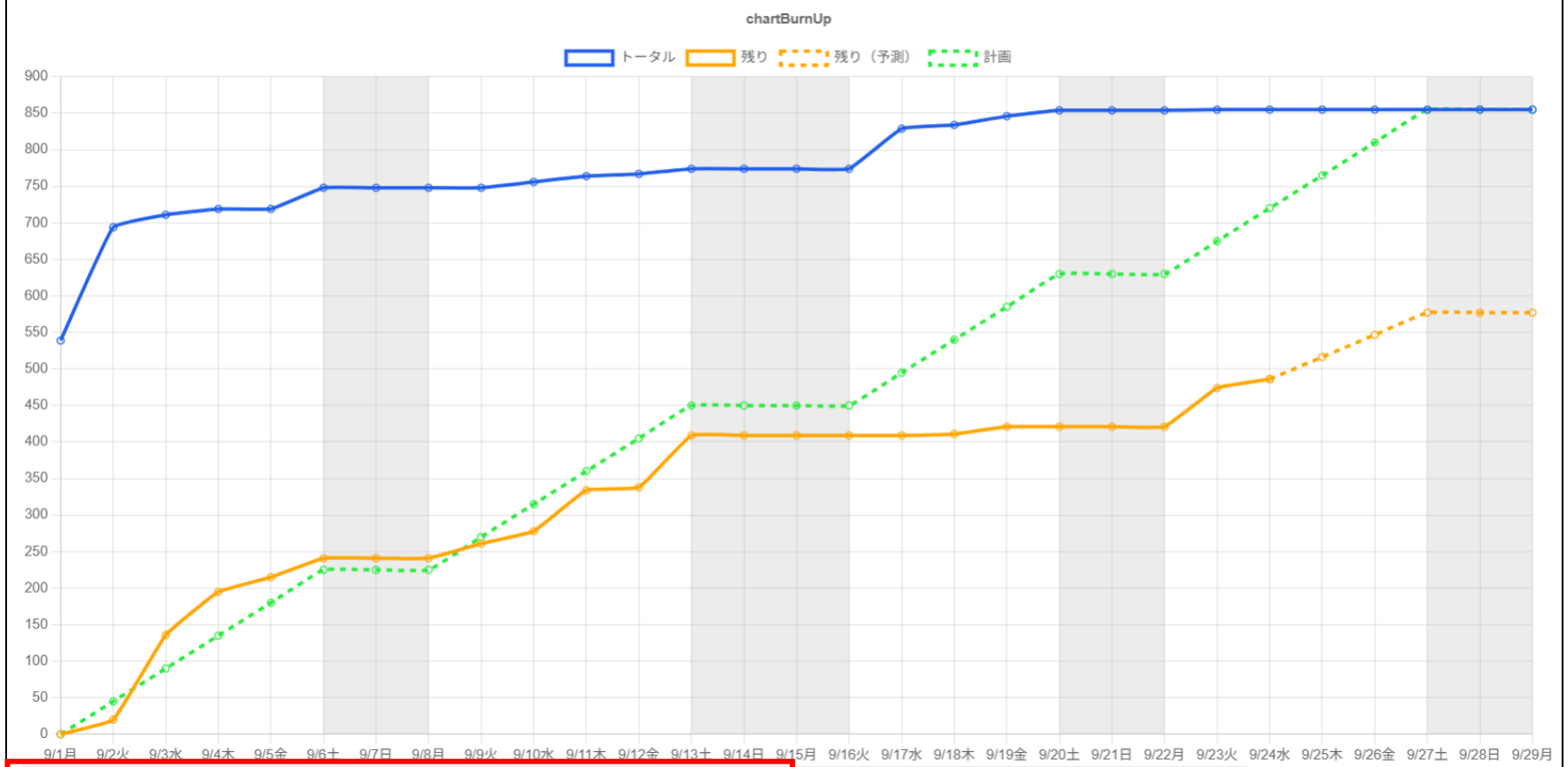
会社休日の設定



Tips: なぜバーンチャートを自作したのか？

実績から計算した完了予定日の算出(実績と計画の差分)

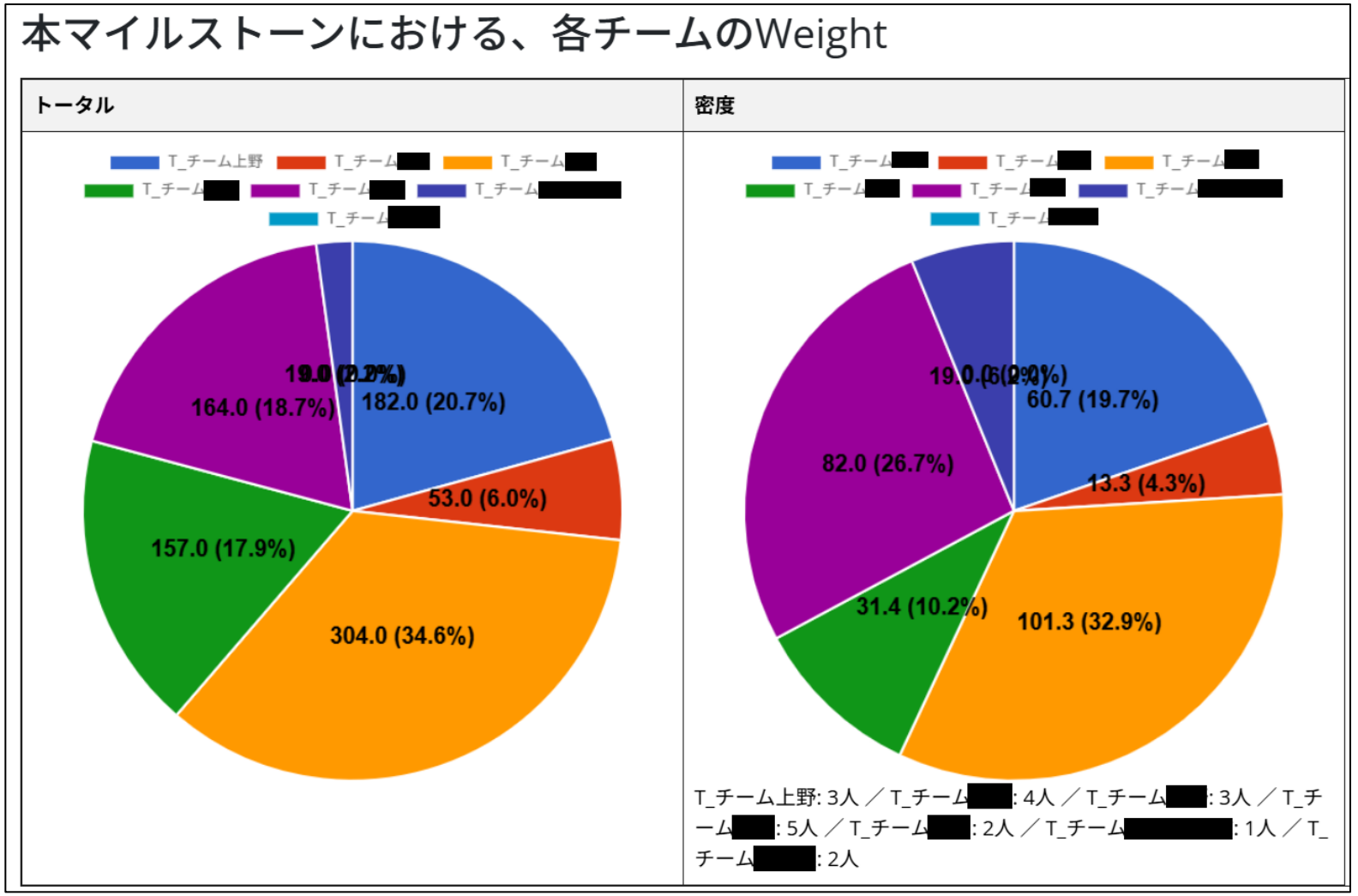
バーンアップチャート



実績と計画の差分	実績	計画
■実績から計算した完了予定日 = 10/14火 ■実績 - 計画 = 17日遅い ■実績Velocity / 計画Velocity = 67.5%	■実績Velocity: 30.38 ■稼働した日数: 16	■計画Velocity: 45.00

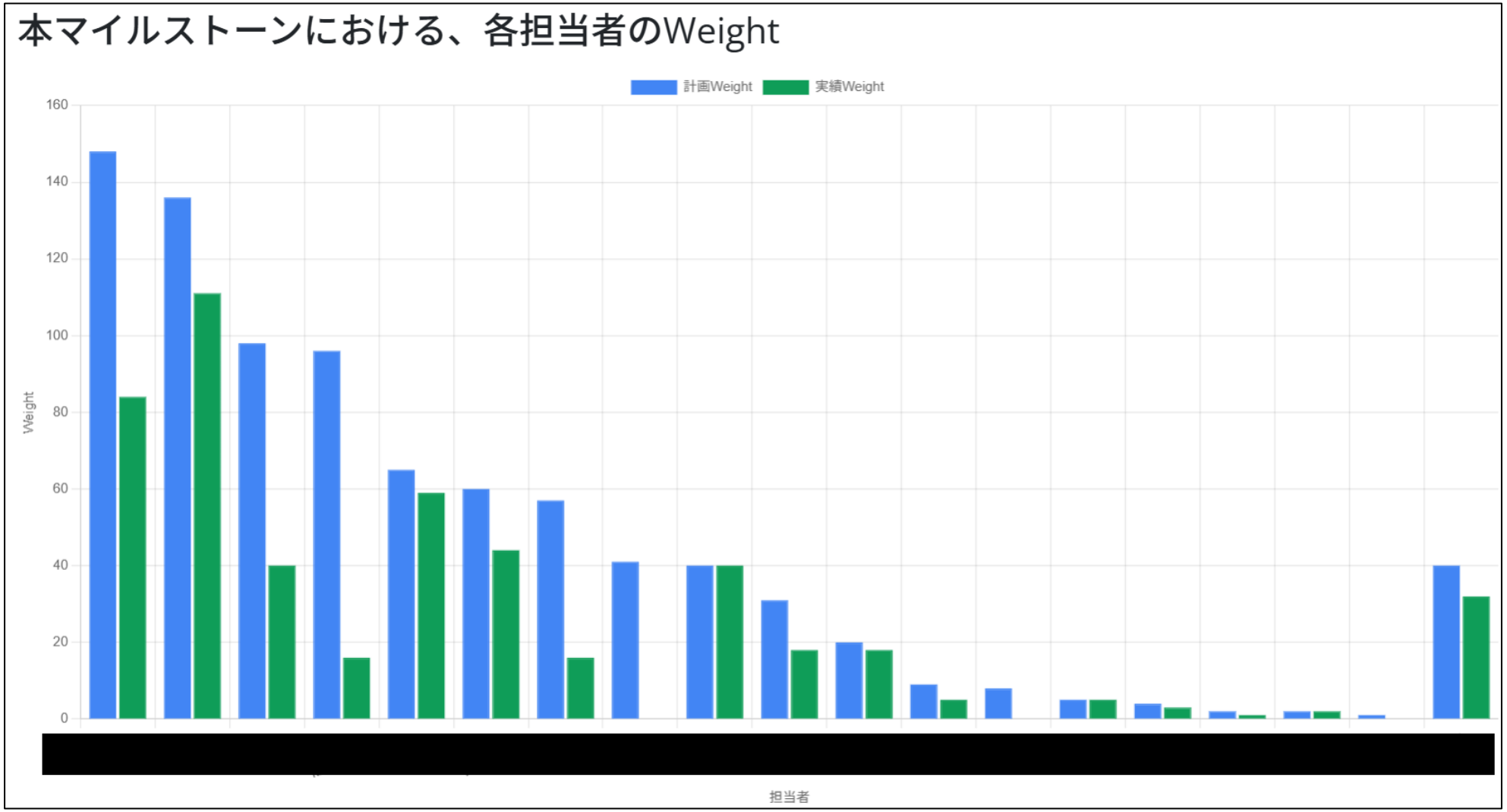
Tips: なぜバーンチャートを自作したのか？

各チームのWeight円グラフ、密度の表示



Tips: なぜバーンチャートを自作したのか？

各担当者のWeight消化状態の表示



GitHubのドキュメント作成・管理システム

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

＋

生成AI
- GitHub Copilot

and More…
- GitHub Codespaces

ドキュメント作成 ～ VS Code + Markdown ～

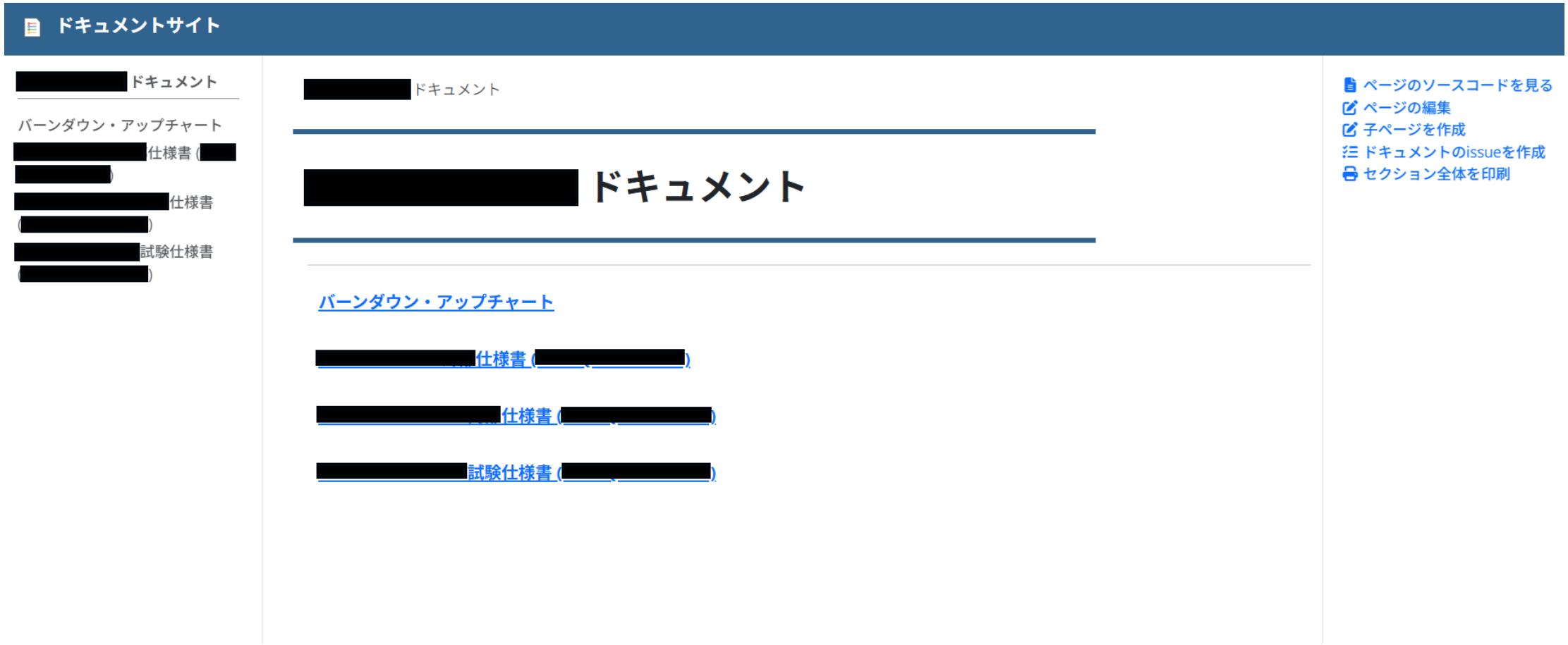
ドキュメント作成は VS CodeでMarkdownファイルを編集

（※ 本画像はイメージ図です。実際の設計書ではありません）

ドキュメント公開 ～ GitHub Pages ～

ドキュメント公開は GitHub Pagesを使用

- Hugo + Docsy で、MarkdownをHTMLにビルドしてWebサイト化



GitHubのソースコード管理

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

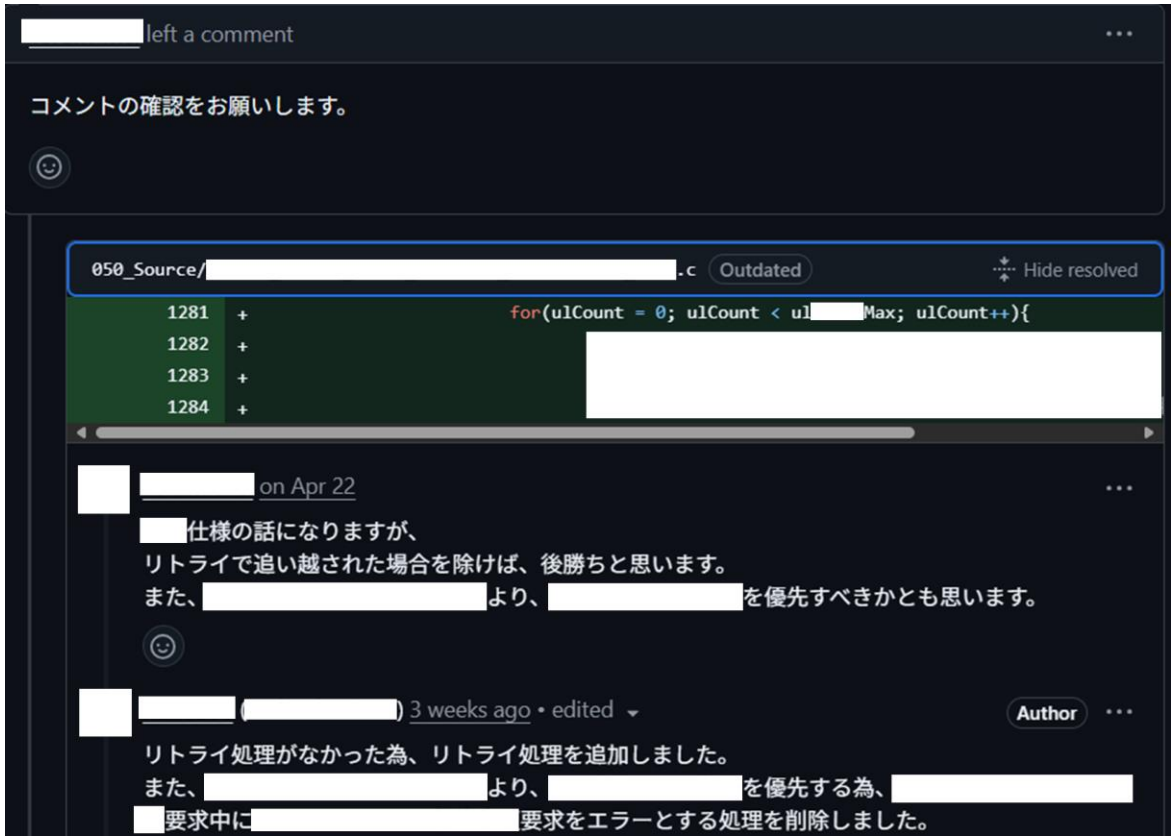
＋

生成AI
- GitHub Copilot

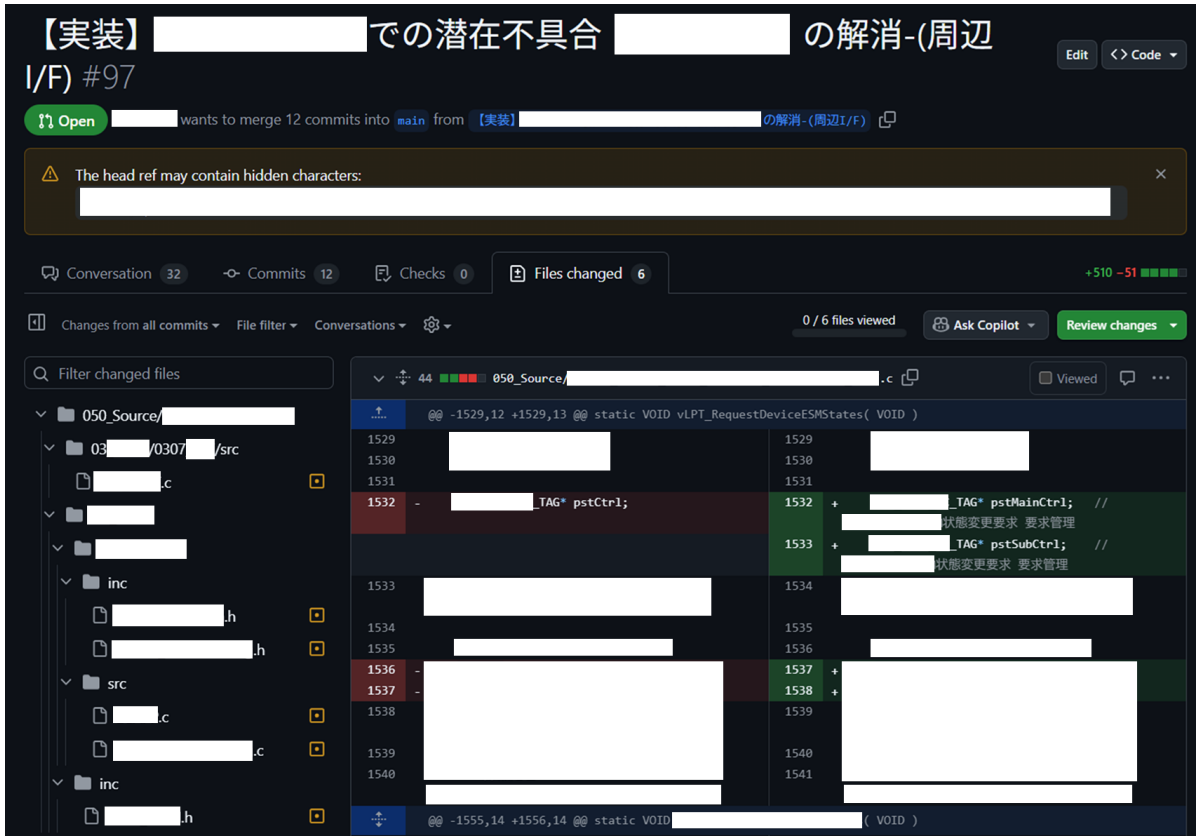
and More…
- GitHub Codespaces

ソースコード管理 ～ Pull Request ～

- Pull Requestで、ソースコードレビューが楽になる
- Pull Requestがマージ完了した時、紐づけたタスク（issue）も自動で完了扱いになる



レビューコメント



Diff(リッチ表示も可)

GitHubのCI/CD

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

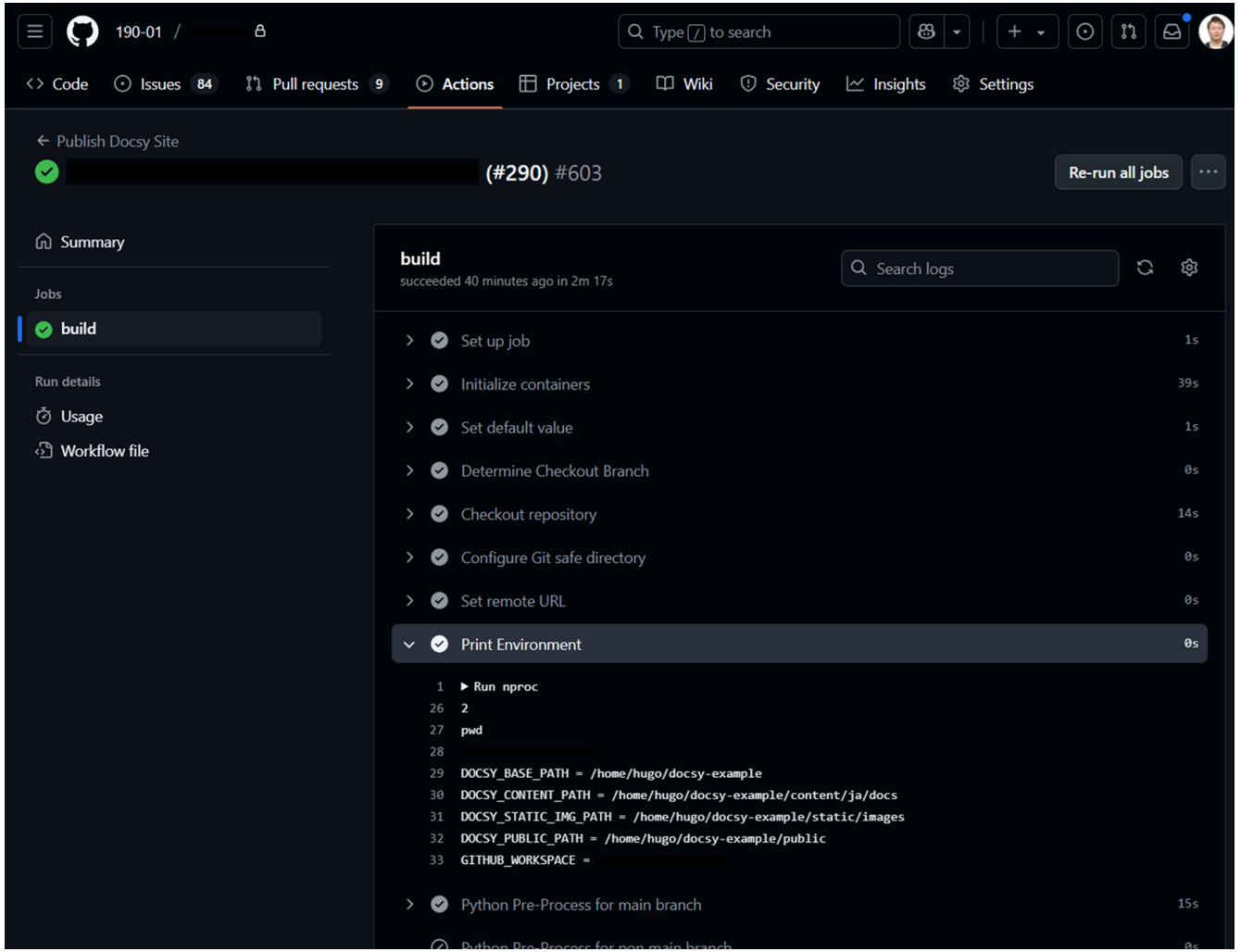
＋

生成AI
- GitHub Copilot

and More…
- GitHub Codespaces

CI/CD ～ ドキュメント生成 ～

ドキュメント（仕様書・設計書）は、CI/CDでMarkdownファイルをビルド（Hugo + Docsyでビルド）



CI/CD ～ ドキュメント生成 ～

仕様書の改訂履歴を自動生成するパイプラインも構築

改訂履歴

改訂	No.	概要	改訂者	改訂日	改訂内容（リンク）	[redacted]	[redacted]
A	1	以下の章を削除した。 • [redacted]	[redacted]	2025/06/03 15:32:40	Ticket#027a29ed	-	-
A	2	以下の章を削除した • [redacted]	[redacted]	2025/06/03 15:42:51	Ticket#0a9a5e15	-	-
A	3	• [redacted]から不要な項目を削除した	Toshiaki Ueno	2025/06/03 18:33:35	Ticket#1c6fba7b	-	-
A	4	• [redacted]の誤記を修正した。	Toshiaki Ueno	2025/06/03 19:01:30	Ticket#852752d0	-	-
A	5	• [redacted]機能の記述を削除した	Toshiaki Ueno	2025/06/03 19:25:03	Ticket#d7aa058f	-	-

改訂履歴一覧

変更前

No	名称	本開発での開発対象 ○：開発対象である ×：開発対象でない	説明
1	[redacted]	○	[redacted]ユニット[redacted] [redacted]を設定し、[redacted] [redacted]の管理、[redacted]通信、[redacted] [redacted]通信を行う。

変更後

No	名称	本開発での開発対象 ○：開発対象である ×：開発対象でない	説明
1	[redacted]	○	[redacted]を指す。 [redacted]を設定し、[redacted] [redacted]の管理、[redacted]通信、[redacted] [redacted]通信を行う。

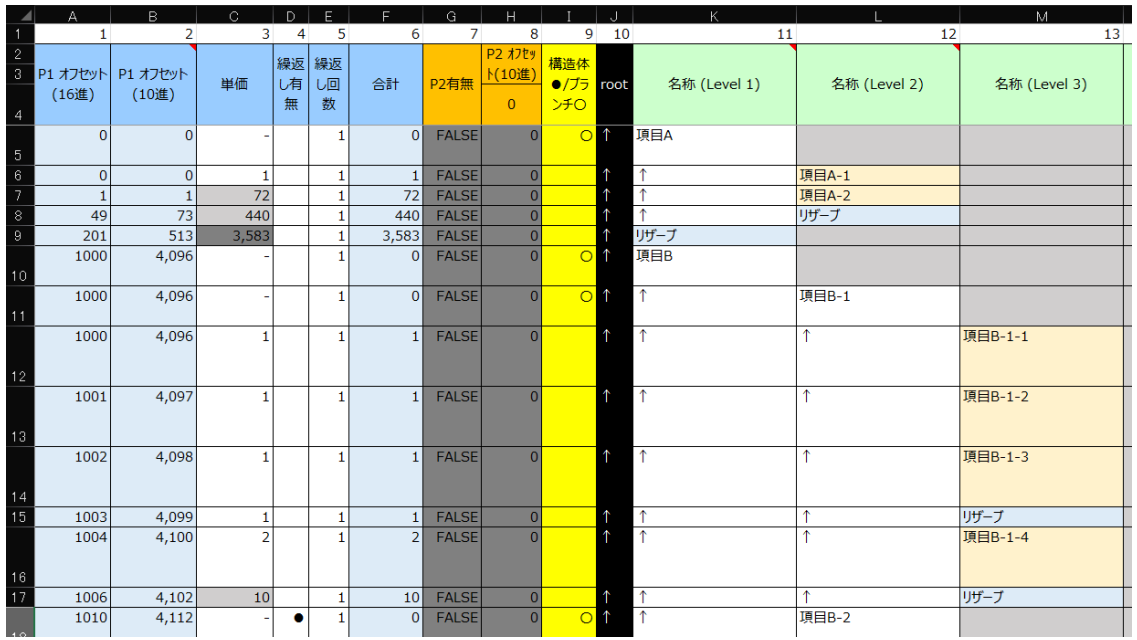
変更箇所ハイライト

CI/CD ～ 仕様からコード生成 ～

仕様書からソースコードの一部(.h)を生成するパイプラインも構築

- メモリマップExcel ⇒ メモリ構造定義の.hファイル生成
- エラーコード一覧csv ⇒ エラーコード定義の.hファイル生成

(※本画像はイメージ図です。実際の仕様書やソースコードではありません)



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1	2	3	4	5	6	7	8	9	10	11	12	13
2								P2 オフセット(10進)	構造体●/ブランチ○				
3	P1 オフセット(16進)	P1 オフセット(10進)	単価	繰返し有無	繰返し回数	合計	P2有無	0		root	名称 (Level 1)	名称 (Level 2)	名称 (Level 3)
4													
5	0	0	-		1	0	FALSE	0	○	↑	項目A		
6	0	0	1		1	1	FALSE	0		↑	↑	項目A-1	
7	1	1	72		1	72	FALSE	0		↑	↑	項目A-2	
8	49	73	440		1	440	FALSE	0		↑	↑	リザーブ	
9	201	513	3,583		1	3,583	FALSE	0		↑	リザーブ		
10	1000	4,096	-		1	0	FALSE	0	○	↑	項目B		
11	1000	4,096	-		1	0	FALSE	0	○	↑	↑	項目B-1	
12	1000	4,096	1		1	1	FALSE	0		↑	↑	↑	項目B-1-1
13	1001	4,097	1		1	1	FALSE	0		↑	↑	↑	項目B-1-2
14	1002	4,098	1		1	1	FALSE	0		↑	↑	↑	項目B-1-3
15	1003	4,099	1		1	1	FALSE	0		↑	↑	↑	リザーブ
16	1004	4,100	2		1	2	FALSE	0		↑	↑	↑	項目B-1-4
17	1006	4,102	10		1	10	FALSE	0		↑	↑	↑	リザーブ
	1010	4,112	-	●	1	0	FALSE	0	○	↑	↑	項目B-2	

```
447  // メモリ構造体
     ...
448  typedef struct tag_MEMORY {
449      A_TAG stA; // オフセット=0x0 項目名=A
450      USHORT ausReserved35[3583]; // オフセット=0x201 ワードサイズ=3583 項目名=リザーブ
451      B_TAG stB; // オフセット=0x1000 項目名=B
452      USHORT ausReserved36[4080]; // オフセット=0x3010 ワードサイズ=4080 項目名=リザーブ
```

メモリマップExcel

メモリ構造定義の.h

GitHub Copilot

GitHub

タスク・進捗管理
- GitHub Projects
  - カンバンチャート
  - バーンチャート
  - ラベリング
  - カスタムフィールド

ドキュメント作成・管理
- GitHub Pages

ソースコード管理
- Git
  - Pull Request

CI/CD
- GitHub Actions

＋

生成AI
- GitHub Copilot

and More…
- GitHub Codespaces

設計書 ⇔ ソースコードの乖離をチェック

- .github/prompts/ で設計書⇔ソースコードの配置構造を教えておく
- レビューの目的を教えておく

Source Code and Document Reviewer Agent Instructions

- あなたは、ソースコードと、ソースコードの設計書をレビューする存在です。
  - ソースコードの設計書は、ソフトウェア設計書 (SW設計書) と呼称されます。
- また、レビューするだけでなく、必要に応じてSW設計書を作成・追記・修正することもあります。
- ソースコードとSW設計書のディレクトリ構造を理解し、レビューと設計書の作成を行ってください。

ソースコードとSW設計書の格納ディレクトリと構造

- ソースコードとSW設計書の格納ディレクトリと構造は、src-doc-dir.md に記述されています。

プロンプトショートカット定義

使い方

- `#REVIEW クラス名` と入力すると、そのクラスのSW設計書レビューを行い、実装との乖離点を指摘・修正案を提示します。
- 以下、使用例です。
  - `#REVIEW QUE` と入力すると、QUEクラスのSW設計書レビューを行います。

ショートカットルール

- #REVIEW {クラス名}
  - {クラス名}クラスのSW設計書レビューを行い、実装との乖離点を指摘・修正案を提示せよ。
  - もし、修正が必要な場合は、SW設計書を修正して、修正内容を提示せよ。

```
40_Unit-A/
├─ README.md : Unit-A全体のSW設計書です。
├─ 03_DRV/
└─ 04_LIB/
   ├─ README.md : 04_LIBパッケージ全体のSW設計書です。
   ├─ CAL/
   │  └─ README.md : CALクラスのSW設計書です。
   ├─ CRC/
   │  └─ README.md : CRCクラスのSW設計書です。
   ├─ MAC/
   │  └─ README.md : MACクラスのSW設計書です。
   ├─ QUE/
   │  └─ README.md : QUEクラスのSW設計書です。
   └─ ZIP/
      └─ README.md : ZIPクラスのSW設計書です。
```

配置構造のインプット

「実装との乖離点を指摘・修正案を提示せよ」という依頼

サマリ

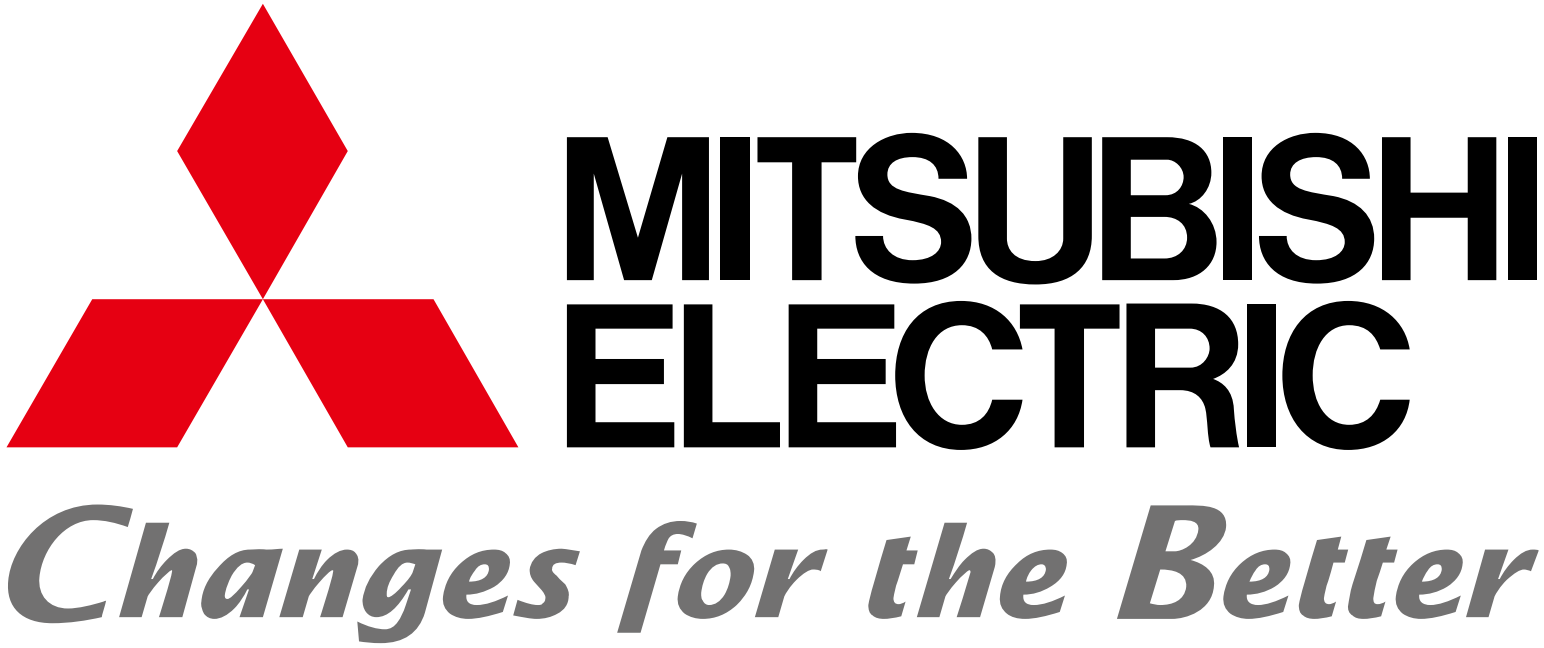
サマリ

過去いろいろな開発環境を試してきたが、**組込みSW開発でも、現時点では GitHub が最適解**

- 実績
  - 過去の開発で、18カ月 ⇒ 14カ月と、開発期間を22%短縮したが、GitHubが持っている機能を使わないと達成不可能だった。
- 理由
  - **GitHubは、SW開発プロジェクト管理に必要な機能をデフォルトで全搭載している**
  - **そのため、【タスク管理】⇔【ソースコード管理】や、【ドキュメント作成】⇔【CI/CD】連携が容易**
- さらに・・・
  - 現在であれば、【ドキュメント】⇔【ソースコード】の双方を生成AI（Copilot）に食わせて解析やレビューが可能
  - 現在であれば、IssueにCopilot Agentをアサインして、作業をお任せ可能

⇒ **Copilotをもっと活用出来れば、さらなる開発効率化が可能！**

InnerSource x GitHub を活用し、
コンポーネント開発の生産性を向上させ、
それらを社内フルオープンにすることで、
事業間シナジー実現に貢献します


MITSUBISHI
ELECTRIC
Changes for the Better