Claude Code で作る
全自動並列AI駆動開発システム

VWSオンライン勉強会 #046

石川栄和@Vektor,Inc.

Vektor,Inc.
Web Solutions

ようこそ！はじめに

この勉強会について

運営：株式会社ベクトル

WordPressやウェブ制作にまつわる様々なテーマをとりあげて開催しているオンライン勉強会。

ご興味がある方であれば、経験や技術レベルに関係なく、どなたでもご参加いただけます。

この勉強会は無料ですが...

知見をシェアします！

というような崇高な精神ではやっていません。

＿人人人人人人人人人人人人人人人人人＿
＞　全ては自社製品PRのためです！！　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣

宣伝

■ WordPress テーマ Lightning / X-T9 （無料）

■ コピペで簡単にページが作れる VK Pattern Library
https://patterns.vektor-inc.co.jp/

■ 数クリックでデモサイトをまるごとインポート
https://vk-fullsite-installer.com/

無料版もあるけど...

■ Vektor Passport

- 合計500以上のブロックパターン
- パスポートユーザー専用デモサイトデータ
- ブロック拡張プラグイン
- スマポンシブテーマ
- AB テストプラグイン

その他多機能プラグインなどいろいろ使えます！

https://vws.vektor-inc.co.jp/vektor-passport

宣伝 その2

■ 予約管理プラグイン

https://vk-booking-manager.com/

メディアスポンサー

SAKURA internet

さくらのレンタルサーバー

ディレクトリ単位や簡単インストールでインストールした
WordPressを指定してステージング環境の作成や
スケジュールバックアップができます

ベクトル製品 2,000円OFF クーポン付き

勉強会中のコメント

勉強会中のコメントは YouTube の方によろしくお願いします。
なるべく拾っていきたいとは思っています。

歓迎されること

- **チャットでわいわい** コメントしてください。
- ぜひSNSにも投稿して盛り上げてください **#wpvektor**
- **やさしい言葉使い** を心がけて、誰にとっても快適な勉強会となるようにご協力ください。

本日の内容

- ご挨拶・その他お知らせ（約10分）
- 本編（約60分）
- 質疑応答（～20分程度）

セッションの内容は後から振り返りできます

URLリンク情報などはコメント欄や
X のアカウントから投稿します。

https://x.com/vektor_inc

フォローよろしくお願いいたします。

https://www.threads.com/@vektor_inc

それでは本編スタート

今日の内容

AIでコードを書くだけではなくできるだけ自動化したい！

- 複数のAIに役割を分担させる
- レビューや修正まで自動化する
- 並列でタスクを進める

といった開発を実現するために...

- ルールの作成
- スキルの作成
- エージェントの作成
- チームでのAI設定の共有
- 統合GUIの作成
- 自動化システムの作成
- スマホで出先やお風呂から制御

についてお話します。

※ はじめに

参加者にはAIコーディングに慣れてない人もいると思います。
そういった人にもなんとなく流れがわかるように、
ロードマップ的に説明します。
特に前半の説明では
「今時そんなやり方しねーよ」って手法から入りますが、
なぜ現状の手法にたどり着くのかを説明するためにあえて
セオリーでないかもしれないやり方から説明します。
あらかじめご了承ください。

あと...

＿人人人人人人人人人人人人人人＿
＞　そもそも私も手探りです　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣

間違い・やり方が古いなどあるかもしれませんが、
生暖かく見守りいただけると幸いです。

ルールの整備

コーディング規約などを指定

実装にあたって持たせておきたいルールを設定

- 一般的にスタンダードな規約
  - WordPressなら WordPress Coding Standards
  - Pythonなら PEP 8 (命名、整形、docstring)
  - Goなら gofmt と Go のレビュー規約

その他事前に用意するルール例

- チーム内の共通独自ルールなど
  - 関数やメソッドの命名規則やディレクトリ構造など
    coding-rules.md
  - CSSの命名規則やネストやディレクトリ構造など
    css-rules.md
  - デザインシステム
    design-rules.md
  - PHPUnit テストの形式
    phpunit-rules.md

特に気にしているルールなどの例

PHPUnit テストの形式

自動でテストコードを量産されてもそのテストコードが何をしているのかもわかりにくい

- 設定値と期待値をセットで記載して、そこだけ確認すれば抑えないといけない動作の漏れがないか確認できるように

ルールの保存先例

- 全プロジェクト共通
  Users/ユーザー名/.claude/rules/*****.md
- プロジェクト固有
  ./claude/rules/*****.md

任意に設置した場所のファイルを自動では読んでくれない。
→ プロジェクト直下に AGENTS.md ファイルを置いてそこから
からルールファイルを読み込ませたりしてたのですが...

本当は必要な時だけルールを読み込まないと...

- 無駄にルールを全部読み込んでトークンを消費する
- 不要なルールまで読み込むと精度が落ちる

後々作るスキルやエージェントで、必要な時だけ必要なルールで読み込むように設定します。

よくわからなければ

○○○ の開発をするにあたって、

コーディング規約を作っておきたいです。

どんなルールにしておくのが一般的？

みたいに雑に聞いてもそれなりに案内してくれます。

初期実装

仕様やルールができたら部分的にテスト実装

最初は Claud Code や Codex みたいな
ターミナルから一気に実装するのではなく、
VS Code や Cursor などのエディタのチャットウィンドウで
対話しながら部分的に実装してもらいます。

→ AIが実装した内容を確認・チューニングするため

実際に実装してもらうと...

「いや...そうじゃないんだなぁ...（＝ｗ＝」

という不満が沢山出ると思います。

ルール化してあったのに...

ルールはある程度読ませてあっても、
実際にプロが今まで気を配っていた細かい知識は膨大

- 関数やCSSの命名（一般的ではなく独自ルール）
- テストの書き方
- UIでの説明文のわかりやすさの粒度
- マークアップの作法
- 余白のとり方
- デザインの視線誘導

A. やっぱりAIは使えない。→ 一生使えるようにならない。
B. ダメな部分を言語化してルールをブラッシュアップしていく

ルールは定期的にブラッシュアップする

運用しているとどんどん追加されて長くなる。

長ければいいわけではない

- トークン消費量が増える
- 長いとそれだけノイズが増えて、ルールが守られなくなる。

```
既存のルールファイルについて、表現が誤解を招きやすかったり、
説明が冗長になっていないか確認して、AIが誤解無くルールを理解できるように
改善してください。
```

コーディング以外もルール化

初期実装以降であれ追加変更であれ、
ファイル変更のコミットやプルリクエストの作成を繰り返す

- コミットメッセージを考えるのが面倒
- プルリクの本文を書くのが面倒

→ AI に「コミットして」「プルリク出して」って言えばやってくれるけど...

社内でやってたルールと違うフォーマットだったりする

- コミットメッセージのルール
- プルリクタイトルのルール
- プルリク本文に書く内容や情報の粒度

→ ルール化

余談1：例えばプルリクのタイトルは changelog / 更新履歴のお知らせページ / SNSでの更新通知 などにそのまま反映させたりするのでとても重要

余談2：ルールがあっても人力だと...

- 社内でルールがあっても守らない人がいる
- PRの本文など、雑に書かれると何をチェックすればいいのか意味がわからずレビューに時間がかかる

ルールに沿ってAIにつくってもらった方が作業者によるブレがなくて安定する。

スキルの作成

スキルとは？

- よく行う作業の手順書のようなもの
- コマンドで簡単に呼び出せる

ルールがある上で、

- 普通に「プルリク出して」とAIに伝えても ルールファイルを読み込んでくれなかったりする
- 「pull-request-rules.md を読んでルールに沿ってプルリク出して」と毎回言うのもだるい

→ スキル化します。

スキル例

```
---
name: vk-pr
description: "PR ルールに従い、コミット・changelog 記載・確認手順を含む
---

# /vk-pr スキル

このスキルは `rules/pull-request.md` の PR ルールに従い、**PR 作成まで

～ 以下略  ～
```

（実際の vk-pr/SKILL.md で軽く解説）

ちなみにスキルファイルは手で書いたりしてません。
要望とスキルのコマンド名を指定してAIに作ってもらってます。

./claude/skills/vk-pr/SKILL.md
./claude/skills/vk-add-phpunit/SKILL.md

など、ユーザー直下に `.エージェント名/skills/スキル名` のディレクトリに `SKILL.md` ファイルで保存されます。

人間がボトルネックになる

開発・PR作成をAIがするようになる
→ レビューの数が増える
→ 手動で人間が確認とかめんどくさくてやってられない

→ レビュー用のスキルを作る

※ 標準で code-review ってスキルもあるけどね...

レビュー用のスキルも作る

ルールがあっても実装者が100%それに従うとは限らない。

- タイトルやプルリク本文がルール通りになってるか？
- 各種コーディングルールが守られているか？
- PHPUnit / e2eテスト がルール通りに書かれているか？

ルールチェックに加えて...

- セキュリティチェック / UXチェック
- e2eテストを実施＆スクリーンショット撮影
- 結果報告

→ スキルとして登録しておけば同じ処理を自動でしてくれる！

え？

その作り方を知りたいんだ？

プルリクのURLを投げたら上記項目を自動でチェック・報告してくれるスキルを作成してください。
参照する ◯◯ルールは **.md、××ルールは **.md です

とか投げれば作ってくれますよ！

ルールとスキルの棲み分けが効いてくる

- 実装やプルリクを立てる時
- レビューする時

→ ルールは同じ

実装スキルとレビュースキルで同じ注意事項をそれぞれ書くと...
二重、三重管理になっていく。

- ルールのばらつき
- 片方の変更・追加漏れ

実装のスキルもレビューのスキルも、
同じルールファイルを参照させる。

サブエージェントの作成

さらに自動処理したい

「プルリク作成スキル」「レビュースキル」ができるなら...

- issue に対する実装前の仕様検討・提案
- 実装後にレビューでひっかかった場合の自動修正

→ 組み合わせたスキルを作れば issue に対して
検討 → 実装 → テスト → レビュー（ 必要に応じて修正 ）
まで一気にやってくれる自動化スキル作れそうよね？

1人のAIに全部やらせると...

全部同じAIが、自分で実装して、自分でレビューすることに...

→ 役割を持ったサブエージェントを作る

- ディレクター
- WordPress実装担当
- UI / UX担当
- ブラウザテスト担当
- コード品質・セキュリティ担当

役割ごとに、参照するルール・使うスキル・完了条件をもった
サブエージェントを定義する

役割を分けるメリット

コンテキストを分離できる

それぞれ必要な専門知識だけを読み込ませられる

並列で作業進められる

特定の機能追加について、実装担当と UI / UX 担当が同時に検討してそれぞれの見解を issue にコメントするなど。

チェックの独立性が上がる

レビュー担当は完成した差分を第三者視点で確認。
「自分が書いたコードだから大丈夫」という前提を持たずに確認できる。

エージェントやモデルを変えられる

- 設計には性能のいいモデル、単純作業にはコストの低いモデルを割り当てたりできる
- 実装エージェントは Claude / レビューエージェントには Codex などを割り振る

作成したエージェント

名前	主な役割
司	全体の進行管理・担当者への割り振り
和田	主に WordPress 関連などプログラムの実装
植草	UI設計・UX・アクセシビリティ
麗美	Playwright によるブラウザ動作テスト
安藤	コード品質・セキュリティの最終レビュー

え？

その作り方を知りたいんだ？

`ディレクター / プログラマー / UI・UXデザイナー / UIテスター / セキュリティーレビュワー のサブエージェントを作りたいです`

とか

`GitHub の issue を解決してプルリクを出すためのサブエージェントを作りたいです`

とか投げれば相談にのってくれますよ！

エージェントは「名前をつけたAI」ではない

必要なのはキャラクター設定よりも、責任範囲の明文化。

- 何を担当するのか
- 何を判断してよいのか
- どのルールを読むのか
- どこまで作業したら完了なのか
- 誰に、何を報告するのか

ってこの部分の原稿はAIが提案したのですが...

わかってない

┐(´д｀)┌

エージェントに必要なのは...

＿人人人人人人人人人人人人人＿
＞　キャラクター設定だ！　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣

(実際の vk-agents-personas ファイルを軽く紹介)

エージェントの設定ができたので
issue を自動処理するスキルを作ります！

やりたい事

- issue のURLを投げる
- 司さん（ディレクター）が中身を確認
  - 和田くん（開発担当） / 植草君（UX担当） /
    安藤さん（セキュリティー担当）など、
    必要なサブエージェントを起動して仕様を提案させる

- 明確な不具合修正や issue で既に仕様がしっかり固まっているものなど、人間の確認が不要
  → 実装開始
- 議論の結果人間に方針確認が必要
  → issue に仕様提案をコメントして指示を待機

- 実装完了したら
  - PHPUnitテスト実行
  - 各種レビュー
    - 安藤さんのセキュリティレビュー
    - 植草君の UX レビュー
    - 麗美ちゃんの e2e テスト
  - 問題があれば差し戻して和田君が修正 → 再レビュー

- 一通り完了したら /vk-pr スキルでプルリク作成
- プルリクの内容がプルリクのルールに沿っているかを司さんが確認
  - 問題があれば修正
- 司さんがプルリクのコメントで人間に最終報告

この一連の処理を一気に実行するスキルを作ります。

え？

その作り方？

こういう流れで処理したいんだけどスキル作って！

とか投げれば相談にのってくれますよ！

って事で...

issue 自動処理スキル完成！

GitHub の issue の URL を投げれば自動で各種レビュー済みのプルリクまで出してくれる

`/vk-kore {url}` スキル（これやっといてスキル）爆誕！

定期的にチューニングは必要...

スキルができると

「うひょー！これで超ラクだぜ！ヒャッハー！」

と思いがちですが、そんな事はない...。

意図せず途中で止まったり精度に問題があったりするので、
都度ブラッシュアップの繰り返していきます。

ちゃんと自動処理してもらうために

issue に完了条件を書いておくのは超重要

脳内で「これ対応してもらうはずだったのに」と思っても
AIにとっては「そんなん知らんがな。書いとけや。」となる。

無駄な差し戻しや事前につぶしておきましょう。

チームでの共有

ルールやスキルなどのリポジトリで管理

以下のような構成のGitのリポジトリを作りつつ...

- agents
- rules
- skills
- scripts

./Users/ユーザー名/.claude/ ディレクトリに複製するスクリプトを書いて貰う

スキルのアップデート用のスキルを作る

「スキルをアップデートして」と言ったら

- リポジトリからメインブランチを pull して最新版を取得
- .claude ディレクトリへの複製（展開）スクリプトを走らせる

という事をしてくれるスキルを作成

→ チーム全員が簡単にアップデートできる

共有スキルの注意点

各メンバーのPCのグローバルな .claude/ ディレクトリに展開

→ メンバー独自で作ったり入れたりしてるスキルと名前が被ってると上書きされる。

→ vk- などプリフィックスをつけている

並列実行

自動処理はできるようになったが...

待ち時間が長い。

- 実装待ち
- テスト実行待ち
- 外部レビュー待ち
- 修正・修正後の再確認待ち

1件が終わるまで待ってから次へ進むのはもったいない。

複数のタスクを同時に動かす

VS Code や Cursol などのエディタで実行するのではなく、
ターミナルを複数開いて同時に処理させる

```
タスク A — 実装 — テスト — レビュー — PR
タスク B — 実装 — テスト — レビュー — PR
タスク C — 実装 — テスト — レビュー — PR
```

標準ターミナルの場合

並列処理自体はできるが数が増えると...

- どのターミナルで、どの issue 処理してるかわからない
- どれが実行中で、どれが入力待ちかわからない
- ウィンドウを順番に開いて確認する必要がある
- 終了したセッションを見落とす
- ターミナルを並べ直すだけでも面倒

AIの処理速度より人間の監視がボトルネックに

並列数を増やすほど、
人間がターミナルを見回る仕事が増えていく。

→ 複数セッションの状態を一覧できるUIが必要。

統合UIの作成

VK Terminals

複数のターミナルを自動折返しグリッドで表示する
Electron 製デスクトップアプリを作成しました。

https://github.com/vektor-inc/vk-terminals/

※tmux 使えよという説もありますが独自機能つけたかったし...

複数のClaudeを一画面で操作

- ペインごとに Claude Code を起動
- ペインを追加・並べ替え・リサイズ
- 不要なペインはサイドバーへ格納
- Claude の使用量を表示

統合UIで「見やすく」はなった

- ターミナルウィンドウがばらつくよりはかなり見やすい
- 入力待ちの場合にラベルがつくようにした

けれど...

- どのペイン（ターミナル）が、どのタスクかわかりにくい
  issue やプルリクの URL が探しにくい。ブラウザのタブが散らかる

加えて、まだ人間がやる事が多い

- issue を確認する
- ペインを開いて Claude に issue を投入する
- GitHub の issue や PR のページにアクセスして
  コメントや状態を確認する
- 完了・失敗を判断する
- 次のタスクを投入する

人間がディスパッチャーになっている

一部はAIに作業を任せたが、
人間がAIへ仕事を配り続けている。
監視確認も面倒。

→ タスクの割り当てと状態遷移も自動化する。

統合制御システム

VK Orchestrator

Vektor,Inc.
Web Solutions

並列実行を円滑にするために

- 進行中の issue や 実行待ちの issue を見える化
- 人間の入力が必要な状態の見える化
- 後で処理したい issue などをタスクキューとして登録
  - 他の作業の完了後に実行したい
  - トークンリミットの都合で今走らせられない

VK Orchestrator の構成

- 作業を実行する AI のルールやスキルなど各種設定
  → VK Agents
- 作業を行う実行GUI
  → VK Terminals
- 進行を制御するシステム
  → VK Orchestrator

やりたいイメージ

- 作業させる issue をタスクキューとして登録
- そのタスクキューをオーケストレーターが定期巡回
- 実行待ち issue 情報を取得して VK Terminals の api に投げる
  - issue のタイトル
  - issue のURL
  - Claude に投げるスキル（ vk-kore ） など

- VK Terminals が受け取って自動的に処理用のペインを開く
- VK Terminals のペイン内の Claude へ自動投入、
  自動処理スキル（ vk-kore ）で プルリクまで作成
  - VK Terminals で状態確認ができる

専用リポジトリの issue を受付票にする

× 各プロジェクトに issue が存在しているので横断して見にくい

- 自動処理の対象の issue を管理する専用のタスクキューリポジトリを作成し、そのリポジトリの issue に 自動処理する issue を登録していく。

1. 各プロジェクトの issue にタスクキュー登録用のラベル付与
   ※ チームで運用の場合はアサイン（担当ユーザー）を自分に
2. オーケストレーターが定期巡回、タスクキューラベルを検出
3. 専用のリポジトリの issue に同名の issue を起票
   中身は元 issue へのリンク

タスクキューリポジトリの issue の情報

AIが作業を始めるために必要な情報を持たせる

- アサインユーザー
- 作業対象 issue の url
- 作業ステータス
- 並列でよいか、順番に実行するか
- 優先度
- 自動マージしてよいか

ラベルを状態管理に使う

タスクキューリポジトリに登録されても即時実行はされない

例：

```
status:waiting-approval
    ↓
status:ready
    ↓
status:in-progress
    ↓
status:waiting-merge
    ↓
status:done
```

エージェントとオーケストレーターの責務

オーケストレーター

- タスクを選ぶ
- 実行場所を割り当てる
- 状態を監視する
- 完了条件を判定する
- 次のタスクを流す

エージェント

- 仕様を理解する
- 実装・テスト・レビューを行う
- PRを作成する
- 結果を報告する

例外設計

長時間動かすと必ず例外が起きる。

- ペインが消えた
- Claude Codeの起動に失敗
- 指示本文が届かなかった
- PR作成前にセッションが止まった
- 一時的にGitHub APIへ接続できない

「成功経路」だけでなく再試行・人間への引き継ぎまで対策

と書くと聞こえがいいですが、
トラブルの都度 issue 起票 → 改善
の繰り返しです。

オーケストレーターで出来るようになった事

VK Terminals との連携により...

- タスクキューの見える化
- 処理中のペイン（ターミナル）で
  - 処理してる issue タイトルがわかる
  - 処理してる issue のページを簡単に開ける
  - プルリクの URL を簡単に開ける

- 入力待ち状態かどうか
- 人間の確認が不要そうなものは自動マージ
- タスクキューの issue を順番に自動実行

など

え？

その作り方を知りたいんだ？

- タスクキューの issue をサイドバーに表示したいです
- 各ペインの状態をラベルで表示したいです
- 各ペインで処理中の issue タイトルをペインの上部に表示してリンクしたいです
- プルリクが出たら、ラベルを表示してリンクしたいです

とか投げてブラッシュアップしまくればよろし。

スマホからの制御

遊びに行ってる時の罪悪感を減らしたい

わたし…


SUPが大好きです

SUPの良くないところ

- 海や川など移動に時間がかかる
- 準備・片付けにも時間がかかる

＿人人人人人人人人人人人人人人人人人人人＿
＞　儲かってないのに遊んでて大丈夫！？　＜
￣Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y^Y￣

移動や遊んでる間もAIが稼働すれば罪悪感が軽減できるのでは...

スマートフォン用のWeb画面を追加

ブラウザから全ペインの状態を確認。

- タスクリスト
- ステータス
- プルリクへのリンク
- ターミナル操作

など。アプリのインストールは不要。

スマートフォンから簡単な応答もできる

- 任意のテキスト送信
- ペイン（ターミナル）の追加・終了

GitHub のスマホアプリから新規 issue を登録して、
実行させる事もできる。

スリープ対策

mac にはもともとスリープ制御する caffeinate というコマンドがある

オーケストレーターは caffeinate コマンドと併せてペインの Claude Code を起動してるので、オーケストレーターが起動してれば mac 本体はスリープしない。
→ 外出からでも操作可能

モバイルからのアクセス

VK Terminals のモバイルページと HTTP API は認証なし。

- 信頼できる自宅・社内LANで使う
- 外出先からは Tailscale でプライベートネットワーク経由
- 不特定多数が接続できるネットワークへ公開しない

便利さと同時に、操作経路の保護が必要。

え？

その作り方を知りたいんだ？

モバイルからもアクセスしたいっす。よしなにたのむ！
ただし、スリープしないようによろしく！

とかAIに投げてブラッシュアップすればいけますよ！

そんな感じで....

最初から全自動を作ったわけではない

1. 不満をルールへ変える
2. 繰り返す作業をスキルへ変える
3. 役割をエージェントへ分けて工程の多いタスクを自動処理できるようにする
4. 複数タスクを並列で動かす
5. 見づらいので統合UIを作る
6. 人間の監視・投入もオーケストレーターへ移す

困りごとを1段ずつ仕組みに変えた結果。

注意点

- 調子に乗って並列で回すとめちゃめちゃ消費する
- ここまで作るのにもめちゃめちゃトライ&エラーの繰り返し

ちなみにこの VK Orchestrator ...

Vektor Passport のユーザーは
お試しでマイアカウントページからダウンロードできます。

まだドキュメントも使い方ガイドも動作保証もしないので、
このために Vektor Passport 購入はおすすめしません。

まぁ動かなくてもルールとかスキルとかどう書いてるのかは
参考になるとは思います。

YouTube チャンネル登録してね

https://www.youtube.com/@VektorInc

ありがとうございました