# アルタスファイブ通信　2024 年冬号

「AI が進化しても、俺たちヒューマンのモノづくりは無くならない！」という訳のわからない迷言から始めてみました。
アルタスファイブがお贈りする社外報 2024 年冬号です。

今号は「共創」と「チームコラボレーション」をテーマに、この半年間の取り組みを綴りました。

「知恵と工夫を凝らす」ことを使命に掲げて、若手社員の率直な成長の記録とベテラン社員の新たな挑戦とともに、等身大の姿として紹介しています。

この「アルタスファイブ通信」が、皆さまとの絆をより深めることを願っています。

目次

**01**　振り返りに対する意識の変化・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　花井

**02**　成長する組織で生まれる共創とチームワークの力・・・・・・・・・・・・・・・・・・・・・・　大成

**03**　視点の違い・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　川野

**04**　同じ問題への対応の違い・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　藤原

**05**　相手の判断基準を理解する・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　檜山

**06**　AI エディタ Cursor を使ってみて・・・・・・・・・・・・・・・・・・・・・・・・・・・　磯貝

**07**　チーム内レビュー会でレビュー期間を短縮する・・・・・・・・・・・・・・・・・・・・・・　神﨑

**08**　チーム内の知識共有手法について・・・・・・・・・・・・・・・・・・・・・・・・・・・　小林

**09**　自分の仕事を客観的に分析しよう（占い師との決別）・・・・・・・・・・・・・・・・・・　佐藤

**10**　開発で変わったスクラムの認識・・・・・・・・・・・・・・・・・・・・・・・・・・・・　柳

**11**　お菓子紹介・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・　宇野

## 01 振り返りに対する意識の変化

**開発部 花井**

最近、大学 1 年から 7 年ほど使用してきたノートパソコンを買い替えました。あまり性能を気にしていなかった人間なのですが、さすがに IDE を起動してコーディングすることすら快適にできない状態だったので、ついに買い替える決断をしました。先日届いたばかりでまだ新調したノートパソコンの使い心地は分かりませんが、以前よりは快適になっていることを期待しています。初めまして、今回がアルタス通信執筆初回となる、開発部 2 年目の花井です。

今回はチームコラボレーションというテーマに関連して、入社 2 年間で起こった振り返りに対する意識の変化について語ろうと思います。

振り返りをする機会は入社 1 年目からありました。ただ、当時は振り返りの重要性についてあまり理解できていなかったように思います。振り返りの時点では改善点が見つかって意識するのですが、その後ちゃんと改善できているかと問われると「そんな気もするかも…」くらいの曖昧な感覚になっていることも多かったです。

2 年目に入ったタイミングで上司から振り返りに関するアドバイスをいただき、それ以降振り返りに対する意識が変わったように思います。そのアドバイスとは、「Action には 2 週間程度で実行したことを確認できる具体的な目標をあげる」です。これを意識して Action を決めることで、次回の振り返りでどう行動してどう改善されたかが分かりやすくなり、振り返りの効果があったことが実感しやすくなりました。また、目標を達成できなかったとしてもできなかったことが明確になり、次回振り返りでなぜ達成できなかったか、どうすれば達成できるようになるかが議論しやすくなっていたように思います。

実際、この方法を取り入れてからは、振り返りがチーム全体にも良い影響を与えていると感じています。例えば、現在私がチューターとして後輩と一緒に進めている 2 人体制のプロジェクトでは、情報の連携不足という課題になったことがありました。去年の私であれば「より連携を密にする」といった抽象的な努力目標をあげていたかもしれません。この時は「slack の canvas に共有すべき情報を記載する」のような具体的な Action を設定したため、明確に改善された点ができ、情報の連携漏れなどが減りました。また、掛け持ちしている別のプロジェクトではレビュワーの負担が課題になっていましたが、「プルリクエストを分割する」という具体的な案により、負担が大きく軽減されました。

振り返りは、単なる過去の反省ではなく、次の行動を具体化し、チーム全体を前に進めるための重要なプロセスだと実感しています。1 年目には曖昧だった振り返りも、2 年目からは具体的な Action を設定することで、その効果を実感できるようになりました。

まだまだチームとしては未熟だと感じていますが、振り返りを通じてチームとしてより成長していけるようになればいいなと思っています。

**会社からの一言：**

朝のワイドショー番組にある”今日の運勢”で、自分の運勢が、最悪だったとしても CM に入った瞬間に忘れてます。それと同じで、“振り返り”での反省も、終わった途端にキレイさっぱり忘れちゃう。だから、振り返りは、儀式化してしまう。そうならないように、確実にできることを決めることと、定期的に確認するという運用が大事ですね。（開発部 佐藤）

## 02 成長する組織で生まれる共創とチームワークの力

**管理部 大成**

アルタスファイブの社員数は、ついに 24 名となりました。この規模になると、全員が直接会話することが難しくなります。また社長一人が全員を細かく見れなくなる規模にもなってくるため、マネジメント層の必要性がでてきます。そこで昨年、新たに「目標リーダー」を設置しました。目標リーダーは半期ごとの目標設定や達成支援、キャリア支援を行い、社員一人ひとりが成長できる環境を整えていく役割を持ちます。あわせて、定期的に目標リーダーが集まり、どのように社員の成長を支援するのかを話し合う「リーダー会」も始まり、組織の連携をさらに強化しています。

規模が大きくなると避けられない課題があります。それが「コミュニケーションコスト」です。ソフトウェア開発にお

ける「ブルックスの法則」（『人月の神話』より）でも示されるように、追加要員が逆にプロジェクトを遅延させるリスクが高まります。

今年 11 月に開催した社内一日研修では、「チームワーク」をテーマにしました。この研修では、大きく次の 2 つのプログラムで構成しました：

1. 書籍『エンジニアリング組織論への招待』輪読会
2. 論理的思考力を鍛えるグループワーク

**書籍『エンジニアリング組織論への招待』輪読会**

『エンジニアリング組織論への招待』では、エンジニアリング組織が直面する不確実性への対応方法が書かれています。社員が担当パートをスライド 1 枚にまとめ、2 分で発表するという方法で進めました。研修後の事後アンケートでは、短時間で本全体の概要を把握することができたとの声が多く寄せられました。また、他のメンバーの要約や発表を聞くことで、担当外の内容も理解しやすく、興味を持った部分を後から読み返すきっかけになったという意見もありました。この本は、少し難易度の高く、なかなか一人で読むには難しい本ではありますが、こうやってみんなで手分けして学ぶことで、一冊の内容を理解することができます。また、一般的な組織論というよりも、”エンジニア組織論”とまさにアルタスファイブの組織に沿った内容でもありますので、今後、実務でも参照・応用していくことが期待されます。

**論理的思考力を鍛えるグループワーク**

論理的思考力を鍛えるセッションでは、論理的思考の概念について学んでもらった後、4 つのグループに分かれ、次の課題に挑戦してもらい、最後に 5 分でプレゼンをしてもらいました：

- 経営目標に直接貢献する施策の特定
- 不足している施策の抽出
- 新たな施策の提案

経営課題をテーマにした理由としては、正解が一意ではないからです。正解がない中で、チームで仕事をしていくには、論理的に説明していく必要があります。チームメンバーひとり一人ひとりが納得した上で仕事をすることで、相乗効果として成果に反映されます。
プレゼンの評価観点としては、「論理的観点」と「経営的視点」の両面から評価しました。「論理的観点」とは、プレゼンの内容が論理的に整理されているか、論理的に説明されているかの観点で評価します。「経営的観点」については、代表の佐藤が経営施策として取り入れたいと思うかどうかの観点で評価しました。論理的思考は、関わる相手を納得することが大事なポイントになります。ですので、経営者である代表の佐藤がな納得できる視点で持って語ることができるかという点を評価軸に取り入れました。

「論理的観点」で最も優れていたプレゼンは沢辺チームでした。経営目標を選んだ理由、不足している施策の特定した根拠、そして新たな施策の提案まで、一気通貫した発表内容でした。

「経営的観点」で最も優れていたプレゼンは小池チームでした。小池チームでは、ウェルビーイングをテーマにいかにして社員がアルタスファイブで働くことの誇りを持てるかを取り上げ、パーパスの浸透が不足していることから、「社員がパーパスの制定に関われる仕組み」を提案しました。この提案に、代表の佐藤からは、面白い施策であると高評価でした。

＊　沢辺チーム：沢辺、神崎、柳、花井、矢島
＊　小池チーム：小池、雲居、西岡、村山、加藤

沢辺チームと小池チームにはシャトレーゼのお菓子が贈られるというささやかなご褒美もあり、楽しみながら学ぶ雰囲気が広がったのではないかなと思います。

**研修を通じて得た学びと気づき**

研修後のアンケート結果かは次のような感想が寄せられました：

- 論理的思考が日々の業務に直結していることを再認識した。
- 自分の行動や上司の指摘を振り返るきっかけになった。
- 他のメンバーの意見に触れることで新しい視点を得た。
- 思考方法だけでなく、伝え方も論理の一部だと気づいた。

全体的なコメントとしては、「新たな気づき」や「他者の視点」といったことへの開放性が高まる場になったようです。

**次へのステップ**

今回の研修を通じて、「共創」と「論理的思考」をテーマにした学びが組織全体に広がり、新たな気づきと成長のきっかけを得ることができました。アルタスファイブはこれからも、一歩ずつ着実に前進し、社員一人ひとりが主体的に活躍できる環境を築いてまいります。

引き続き、私たちの挑戦を温かく見守り、応援していただければ幸いです。

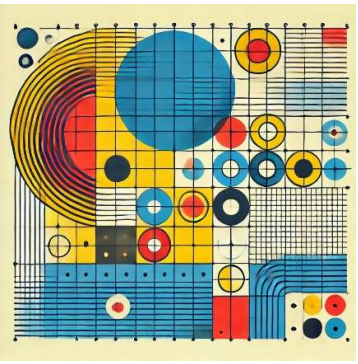
**会社からの一言：**

「エンジニアリング組織論への招待」は、輪読会を聞いていて、初めて内容を知りましたが、難しい本でした。1 人で読むと挫折する類です。グループワークもよかったです。SI 事業は、売上が社員数で決まってしまうようなところがあるの

で、採用を頑張らないといけないということ、利益率を上げるには、単価を上げる、そのためにスキルアップが必須なので、評価制度を導入して必然的な取り組みにしているのだということ、そして、会社が WEB-BEING に取り組んでいる背景には、この会社自体に魅力を作ること、社内定例などでも説明はしているけど聞き流していたことを、あらためて考える機会になったと思うので良かったと思います。（開発部 佐藤）

## 03 視点の違い

**開発部 川野**



テーマに関するアイデアが浮かばなかったため、業務で感じたことについて書かせていただきます。

スプレッドシートアプリはネイティブからクラウドベースに主流が変わりつつありますが、古くから多くのプロジェクトで利用されていると思います。このスプレッドシートの書き方について、自分では思いもしなかった様々なご指摘をいただいてきましたので、そのいくつかを紹介したいと思います。

**1．結合するな**

新卒の頃、設計書を作成する際に見栄えを良くしようとセルを結合したところ、上司から注意を受けました。当時の理由までは覚えてないのですが、計算がやりづらくなるなどの理由だったと思います。セルを結合すると、コピペや関数を使用する際に扱いづらいというのもあり、私は現在でもできるだけ結合を使わないようにしています。

**2．罫線を消してほしい**

デフォルトで表示されている薄い罫線があると見づらいとのご指摘をいただきました。最初はあまりピンときませんでしたが、実際に罫線を消してみると確かにスッキリして見えます。それ以降、要点を書いたシートの罫線はできるだけ消すようにしています。

**3．黄色がチカチカして見づらい**

強調したいセルの背景色を黄色にしていたところ、「チカチカして見づらい」とのご指摘をいただきました。私としては、参考書にラインを引くようなイメージでしたので、気にしたことがなかったのですが、人によってはそう感じることを知り、それ以降は彩度の低いトーンを使用するようになりました。

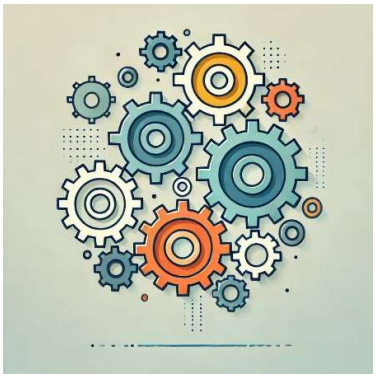
最近、客先でこの「黄色の背景色」の話をした際、色弱の方は彩度の低い色の判別が難しく、逆に見づらく感じることがあると聞きました。これまで 1〜3 の対応をしていれば問題ないと考えていたのですが、バリアフリーの観点を考慮すると、必ずしもそうとは限らないようです。資料を提供する相手に合わせて、柔軟に表現方法を調整することが重要だと改めて感じました。

**会社からの一言：**

バリアフリーというかアクセシビリティのことだと思いますが、私は老眼なので、これをどうにかしてほしいです。コントロールパネルに拡大鏡がありますが、目線移動で混乱するので、正直なところ使えません。（開発部 佐藤）

## 04 同じ問題への対応の違い

**開発部 藤原**



私にはテーマの言葉の意味がふわっとしかわからなかったので、chatGPT に詳しく言葉の意味を聞いてみました。

「共創」とは単なる共同作業ではなく、多様な背景やスキルを持つメンバー同士が互いの知識や経験を活かして新たな価値を生み出すプロセスだそうです。

普段の仕事を思い返しながら、この解釈を考えていきたいと思います。

私は普段は私よりずっとスキルの高い人と二人でプロジェクトをしているのですが、そのためバグ対応をするときにもほとんどのバグに対して原因究明が早いです。

普通はそういうものだと思いますが、ここでは単なる共同作業ではなく互いの知識や経験を生かすという部分に注目し、私のほうが早かったケースを思い出していきます。

バグ対応は慣れるとバグの内容を把握した直後に大雑把にはどこで何が起きているか把握できるものなのですが、私のほうが早かったケースでは何が起きているかパッと見ではわからないような、いわゆる変な状態になったときでした。

変なバグの原因になっている原因を見つけて報告したときには「よくこんなの見つけましたね」と言われました。ただ、これは自分としてはいつもの方法でバグの原因となる箇所を探していたら見つけたものだったのです。

他にももう一つケースを思い出します。それは git の使い方です。

私は普段 Sourcetree を使って git のローカルを管理しているのですが、もう一人のほうがコマンドを直接叩く派という違いがあります。

その影響とは限らないのですが、私が知っているコマンドをもう一人が知らなかったことがあります。

「偶然そのコマンドを使う機会がなかっただけ」などと言ってしまえばそれまでですが、これを多様な背景やスキルと捉えることもできるかもしれません。

以上 2 つのケースを考えると、使っているソフトや思考形態までも多様な背景やスキルとして捉えれば、チームとして動いて同じ問題に対応するときにそれぞれが強みを生かして対応するということの解釈が広がって新たな価値を生み出しやすくなるのではないか？とそう考えることもできそうですね。

「チームコラボレーション」とは、メンバー全員が一丸となり、共通の目標を達成するために力を合わせるプロセスだそうです。

今までの「共創」の話でだいたい終わってしまったような気がします。メンバー全員がそれぞれの強みを活かしていくことができればこれはできていると言えるでしょう。

余談になりますが、私の参加しているプロジェクトでは定期的に「振り返り」というものを行っています。

簡単に言うとタスクをひとつひとつ分割してよかった点、悪かった点、改善する点を挙げていくというものなのですが、これをうまく使ってお互いの強みを相互に把握することはチームコラボレーションや共創の観点からもかなり有効なのだなと思いました。

**会社からの一言：**

変なバグを見つけたときに、なぜ見つけられたのかを考察していたら、それがナレッジになるのだけどね。そこまでは出来ていないようなので、もう一段階、掘り下げて、”なぜ”を究明することを癖にするとよいと思います。（開発部 佐藤）

## 05 相手の判断基準を理解する

**開発部 檜山**

今回のテーマは、「共創・チームコラボレーション」ということで、私は「判断基準を理解する」についてお話したいと思います。私自身まだまだ学習中の分野ではあるので、生暖かい目で見ていただければ幸いです。

仕事を進めていく中で、上司やお客様（あるいはメンバー間同士でも）に承認をいただく場面があると思います。その際の判断基準は、コスト重視だったり、スピード重視だったりと人それぞれ違っていると思います。

この判断基準を明確にしないまま仕事を進めてしまうと、的外れな提案をしてしまったり、必要のない調査に時間と労力を使ってしまったりして、生産性を下げてしまうといったことがあると思います。

私自身の失敗談なのですが、ある機能の実装方式を検討・調査した際に

- A 案：少しダサいが既存実装を大きく変えずに実装できる
- B 案：スマートだが既存実装を大きく変えなければならない（スレッド処理を自前で実装しなければならない）

という 2 案が考えられ、その 2 案の実装方法、それにかかる工数、懸念事項まで詳細に調査し、資料にまとめて共有・相談しました。

しかし、ここでのお客様の判断基準は、コスト、スケジュール、パフォーマンス、メンテナンス性の 4 つであり

その観点で考えると、B 案はコスト、メンテナンス性に問題があり、明らかに NG でした。

（スレッド処理を自前で実装すること自体が NG というわけではないのですが、そのシステムでは並列処理をライブラリ側で制御しているので、その制御から外れてしまうとどんな不具合を生むかわからないという点が NG でした。）

その判断基準を事前に明確にしていなかったため、要点を絞れず、明らかに NG な実装案についても詳細を説明し、説明に時間がかかり、聞く側にも負担を強いてしまいました。

下記のように話していれば、５分で終わる話だったと思います。

「B 案は自前でスレッド処理を実装する必要があるのでコストがかかり、メンテナンス性も悪くなるので、A 案で行こうと思いますが、いかがでしょうか？」

相手の考え方や判断基準を明確に把握しておくことの重要性を実感させられた一件でした。

とはいえ、相手の判断基準を明確にするというのは、時に難しいこともあると思います。その１つの手段として、お客様からお聞きして、私がなるほど～と思った方法を共有させていただきたいと思います。

「相手の判断基準や、考え方・思考法を理解するために"あえて"質問する」という方法です。

この「あえて」という言葉には

- 論理的に考えればこっちだよねということが明確であっても「あえて」質問する
- 相手の考えを理解できていたとしても「あえて」質問する

という意味が含まれていると理解しています。一見すると無駄とも思える質問ですが、こういった質問をすることで

「この方は、こういう思考をして、こう判断したんだな」と理解することができ、色々なケースで相手の考え方を聞くことで、考え方を立体的に理解し、共通する考え方や軸となる部分を理解することができます。

もう一つ重要な点として、ご本人に自分の考え方を自覚して頂けるという点もあります。

特に無意識レベルで行っている判断基準については、ご本人も自覚していないことがあるので

自分の口で話していただくことで、言語化され、自分はこういう判断基準で判断しているのだと認識していただくことができます。

まだまだ試行錯誤している段階でありますが、相手の判断基準を明確に把握し、相手に合わせた提案・説明をすることで、生産性の高い仕事をしていていきたいと思います。ありがとうございました。

**会社からの一言：**

考え方は、一貫性を持っている必要があるけれど、判断基準は、柔軟に変化するものだと思います。昨日言ってたことと今日言ってることが違うじゃないか！ということがあっても、理不尽だと考えない方が賢明です。作成依頼のあった種類を作成して提出したら、ここが違うとか、あれはこうだと、後から言われることがあります。でも、提出した資料という、判断材料が加わったことで、3 つだった判断材料が 4 つに増えたわけだから、判断が変わることがあっても、その方が当たり前とも言えます。（開発部 佐藤）

## 06 AI エディタ Cursor を使ってみて

**開発部 磯貝**

皆さんこんにちは！最近始めたテニスにハマり、週末はテニスプレイヤーと化している開発部の磯貝です。

今回は、話題（？）の AI 搭載コードエディタ「Cursor（カーソル）[i]」を試してみたので、簡単に紹介しつつ、使用感についてお伝えします。（今回のテーマが微妙にずれている気もしますが、「AI との共創」という観点でご容赦ください！）

Cursor は、AI 支援機能を搭載したコードエディタで、VSCode をベースに開発されています[ii]。そのため、基本的な UI やショートカットは VSCode と同じで、拡張機能もそのまま利用可能です。Free プランと Pro プラン（および Business プラン）が提供されており、Free プランは無料で使用できますが、AI 利用回数に制限があります。Pro プラン（$20/月）では無制限で AI を活用可能です[iii]。また、OpenAI や Anthropic の API キーを登録すれば、プランに関係なく自身の API で AI を利用することもできます。

実際に使ってみて特に便利だと感じたポイントを以下にまとめました。

**コードに基づいた回答**

Cursor では、AI とのチャットで'@ファイル名'や'@フォルダ名'を指定すると、そのファイルやフォルダをコンテキストに含めて回答が得られます。さらに、'@Codebase'を指定すればプロジェクト全体が対象になります。

例えば、以下のような質問が可能です：

・ 「@Codebase 税率を計算している関数はどこにありますか？」

・ 「@Codebase プロジェクト全体の構造を説明してください。」

コードリーディングや調査の際に非常に役立つ機能だと感じました。

**外部ドキュメントの読み込みが可能**

Cursor には Docs 機能があり、外部 URL を登録することで、'@Docs 名'のように指定してその内容を参照した回答を得ることができます。

例えば、ライブラリの公式ドキュメントを登録しておけば、わざわざブラウザで調べなくても、直接チャットで質問して解決できます。また、'@Web'と指定すると、AI が関連情報を自動で検索して回答してくれる機能もあります。

**AI によるターミナル操作**

Cursor はターミナルにも AI 機能を統合しています。例えば、必要なファイルの作成、ライブラリのインストール、アプリケーションの実行などを AI に任せることができます。提案されたコマンドは、実行ボタンを押すことで実行されるため、誤って意図しないコマンドが実行される心配もありません。プロジェクトのセットアップなどに非常に便利です。

上記以外にも、テストコードの生成やコードレビューの依頼も可能です。これにより、より効率的な開発が実現します。

便利な Cursor ですが、AI が万能というわけではありません。コンテキストが多すぎたり、質問が複雑だったりすると、期待通りの回答が得られないこともあります。その際は追加の質問をするなどして対応が必要です。ただ、最近の AI 技術の進化を考えると、自然言語だけでコーディングができる時代もそう遠くないのではないでしょうか。

Cursor は、AI 支援を活用してスピーディーに開発を進めることができる非常に有用なツールです。また、VSCode ベースの直感的な UI でストレスなく操作できるのも魅力的でした。今後、AI による開発がさらに進化していく中で、私たちエンジニアもその変化に適応していく必要があります。引き続きアンテナを張りながら、新しい技術をキャッチアップしていきたいと思います！

i. https://www.cursor.com
ii. VSCode 自体にも似たような拡張機能として GitHub Copilot (Chat)があります。また、同じく VSCode を fork した AI エディタで、Windsurf というものも注目されているみたいです。
iii. 一部上位モデル（o1-preview, o1-mini）は回数制限あり。

**会社からの一言：**

AI はツールとして取り入れないと生産性が出せない時代になってきましたかね。つい先日、ChatGPT Pro が出ました。月$200 って、中小企業には使えるわけないじゃん！とあきらめてたのだけど、中小企業には補助金という支援策があるので、中小だったおかげで、全社員分は無理かもだけど、アカウントを準備できそうな兆しが見えてます。月$200 の性能って、どんなものなにか、楽しみです。（開発部 佐藤）

## 07 チーム内レビュー会でレビュー期間を短縮する

**開発部 神﨑**

今年の夏は、外に出るとやる気が削がれるような暑さでした。冬も急に寒くなりましたので健康管理に気を付けたいです。

生成 AI の進歩が早く、いろいろ問題が出つつも便利になってきていますね。最近は生成 AI をできるだけ動作に必要な性能が低くして、オフラインのスマホや PC でも使える方向に進んでいて、そろそろ個人に最適化された学習を行う生成 AI が出てくるのではないでしょうか、生成 AI 技術のこれからが楽しみです。

さて皆さんはこの一年、どのような一年でしたでしょうか？来年がより良い一年になることを願っています。

今回のテーマは「共創、チームワーク」ということなので、参画しているプロジェクトで行っていた開発チームでのレビュー会について書いてきたいと思います。

レビュー会を開催するようになった経緯はレビューからリリースまでの間の期間が長く、また、レビュー者が少ないためのレビュー者の負担が大きいのを改善するための施策として始めました。

チーム内レビューで行っていることとしては、設計やコードを Zoom 等の Web 会議アプリで画面共有をしつつ、レビュー対象者がチーム全員に説明を行います。レビュー対象者以外のチームメンバーは口頭で説明を聞き、質問やレビュー指摘を行います。特にレビュー対象物(詳細設計書やコード)以外の資料作成などはせず、15 分~30 分ほどで終わるように行っています。

レビュー指摘等が無ければ、そのまま、レビュー承認されることもあります。

レビュー会のメリットとしては、レビュー者の時間がとれず、なかなかレビューがされないということを防げることや口頭であるためレビューでのやり取りのレスポンスが早い、チームメンバー全員で行うので、チーム内全体で知識共有されるなどがあります。

デメリットしては、レビュー会の間はチーム全員が参加するため時間をかけすぎるとその分の時間が作業できない事、うまくいかないとただレビュー会をしているだけになってしまうことがあります。

デメリットも工夫次第では改善ができると思います。

チームでは、レビュー時間が長いのはそもそもレビュー対象物が大きすぎるという点もあるので適切な分割を行うやレビュー対象者が若手であれば実装の考えを聞く、熟練者がコードのテクニックなどをレビュー会で伝えるなど、ただのレビュー会ではなく若手の教育も兼ねたレビュー会を行っていました。

個人的には熟練者の方々がこういった観点でレビューしているということも知れるので勉強になります。

現在の開発チームでもレビュー会を始めたばかりということもあり、改善するところはありますが、レビューがなかなか進まないというチームは行ってみてはいかがでしょうか？

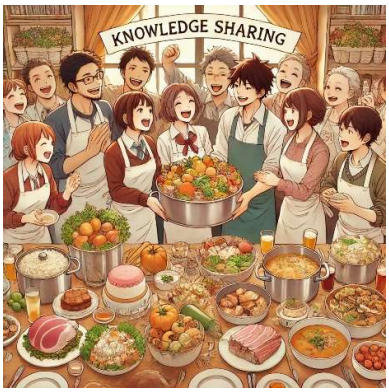
おまけの話として、関わっているプロジェクトでは AI がコードレビューを行う機能がありました。まだまだ、複雑な指摘などは難しそうですが、単純な変数の命名ミスやコメントのミスを指摘してくれたので、人間のレビュー前 に AI でのレビューを行うことでレビュー者が単純なミスの指摘をすることが減り、負担が少なくなっている印象がありました。

**会社からの一言：**

15 分から 30 分のレビューで、しっかりレビューできるのかという疑問はあるけど、内部レビューしてるとか、レビュー会での確認ポイントを絞り込んでるとか、事前にやるべきこを終わらせてるという前提なんでしょうか。
事例として、社内にも展開してほしいですね。私もレビューする機会がありますが、レビューの中でも、テストケースが足りているかをチェックするのが、とにかく大変で、疲れます。C0/C1 のカバレッジ計測に委ねて、テストの網羅性はチェックしないというのも選択肢ですね。（開発部 佐藤）

## 08 チーム内の知識共有手法について

**開発部 小林**




開発部 4 年目の小林です。

今回は、所属プロジェクトで取り組んでいる知識共有の仕組みについて書いていきます。

所属プロジェクトのチームは社外の方を中心に構成されており、そこに altus のメンバーが数名加わるという形になっています。altus メンバーは定期的に入れ替えが発生しており、メンバーによって加入時期が異なります。

ある日、加入直後の altus メンバーが社外メンバーに仕様関係の質問をしたところ、質問への返答に加えて「以前 altus チーム向けに教えた内容は、altus チーム内で共有してほしい。同じ質問に何度も同じ答えを返すのは避けたい」とコメントされました。このコメントを受けて、「加入時期のずれが原因で、altus メンバーの知識レベルにずれがある」「自分以外の altus メンバーが担当した修正内容を把握しきれていない」などの課題を解決するために、知識共有のためいくつかの取り組みを始めました。

１つ目の取り組みは、Tips 集の作成です。これは開発中に気づいたことを置いておく場で、altus メンバーがいつでも閲覧できる場所に整備しています。システムの仕様に関するメモ、ローカル環境での確認手順、エラーの発生内容とその原因など、いくつかのジャンルに分けて閲覧しやすくしています。社外メンバーも閲覧できる開発 wiki と比べると、Tips 集は altus メンバー向けであり、小さな話題も書き残せる場となっています。

こうした知見置き場を用意することで、過去に蓄積された知見にアクセスしやすくなりました。また、アウトプットに対するハードルが下がったことで、積極的に知見を共有するようになりました。

２つ目の取り組みは、altus メンバーでの夕会の実施です。これはメンバーの進捗状況を共有する場で、基本的に毎日実施しています。各メンバーが作業内容・成果・反省点・明日の計画などを日報にまとめ、それを全員で読み合わせるという形式をとっています。私は夕会の進行役を務めており、各メンバーの状況や作業内容などについて、積極的に質問を投げかけるよう意識しています。

こうした報告の場を設けることで、進行中の作業に関する知識を共有しやすくなりました。また、他メンバーが取り組んでいる作業への理解が進んだことで、「以前自分が担当したこの実装が参考になるかもしれない」といった形の意見交換ができるようになりました。

チームとしての取り組みはこの 2 つですが、私はこれらに加えて、altus チーム内で質問しやすい雰囲気を醸成する事にも力を入れています。なぜなら、知識を共有するためには知識を得る必要があり、知識を得るためには質問する必要があるからです。

冒頭で書いた社外メンバーからのコメントは、同じ質問を何度もすることを避けてほしいと言っていますが、質問すること自体を否定しているわけではありません。実際に、作業に関する質問を社外メンバーの方にすると、親切に答えていただけることが多いです。

私は、この「質問しても良い、質問することは悪いことではない」という雰囲気を altus メンバーの間でも構築したいと考えています。そのために、他メンバーに質問や相談をするように促すだけではなく、私自身も積極的にわからないことを他メンバーに質問するようにしています。

これらの取り組みにより、altus チーム内の知識共有は以前と比べて大きく進みました。メンバー間での質問も活発になり、疑問や不明点などを積極的に質問するようになってきています。取り組みを始める前と比べると、altus チーム内で良い雰囲気が醸成されていると感じています。今後もこの取り組みを継続しつつ、心理的安全性などチーム開発をより円滑に進めるための知識や技術を学んでいきたいと思います。

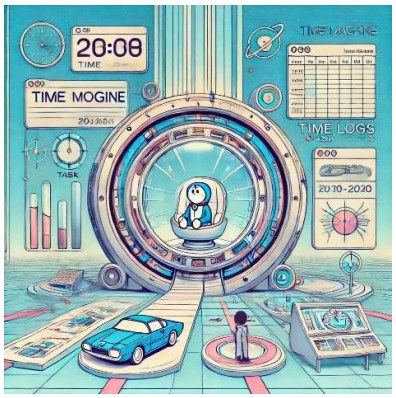
**会社からの一言：**

朝会で、「昨日〇〇やって、今日は××をやります」という、他の人が発言している内容を、ほとんど聞いてないってことありませんか？自分の番に回ってきて、「えーと、なんだっけ・・・」となってませんか？司会進行役をやっているときに、こんな状態の人を見ると、少し悲しくなりますね。

弊社では、夕会をやっているチームでは、短くてよいので、日報を書くことを推奨してます。そうすることで少なくとも無意味な夕会ではなくなるし、報告に対して、何ならのレスをすることを心がけると、思わぬ意見を聞けることもあったりします。（開発部 佐藤）

## 09 自分の仕事を客観的に分析しよう（占い師との決別）

**開発部 佐藤**




サブカルが好きです。”まんだらけ”にときどき行きます。みうらじゅん氏は、心の師です。好きな漫画家は和田ラヂヲ氏なのだけど、ラヂヲ氏の漫画に”とんち”ネタがあるんですが、なんでも”とんち”で解決しようする輩が出てきて場を壊すネタです。例えば、私らの業種だと、システムを”とんち”で作ろうとするわけです。論理的な思考ではなくて”とんち”で開発を図るんです。「屏風の前で虎を出してください」のようなことをお客様に言っちゃうわけですけど、依頼主は、やるせないですよね。視座が低い高いとか、そういうこと以前の話です。ようこそサブカルの世界へ。今度、漫画お貸しするので、読んでみてください。(笑)

さて、カイゼンしてますか？ソフトウェアの開発におけるカイゼンは、その代表格が、”振り返り”だと思います。今号でも、”振り返り”話題がいくつも出て来ています。全部の記事を読んでいただいている方には、食傷気味かもしれません。
私の記事は、少し視点を変えて、カイゼンをハックするためのツールを開発してます、というお話です。

仕事のやり方を改善していくために、まずやらなくてはいけないことは、問題点の炙り出しです。弊社でもほとんどのプロジェクトで、”振り返り”をしています。問題点を炙り出しまくってます。

- PR のレビューに時間がかかってるんです
- 要件の調査に時間がかかったんです
- ユニットテストのメンテナンスに手間がかかりました
- 朝会＋夕会＋個別の MTG と、アジャイル開発は小さい MTG が多いんですよね
- ・・・

とりあえず、”振り返り”機会に、上記のどれかを言っておけば、だいたいのプロジェクトで、同意を得られるのではないでしょうか？
これって、どこかにあったような・・・、そう！ ”占い” です。占い師が言う誰にでも当てはまる事の理を、自分のことを言い当てたと思い込んでしまうアレです。

これを、一種の認知のゆがみと捉えるならば、以下が該当すると、 ChatGPT が申してます。

- 確証バイアス（Confirmation Bias）

自分の信念や期待に合致する情報のみを選択的に受け入れ、反する情報を無視する傾向。占いの一般的なアドバイスを自分に特有のものとして解釈することで、既存の信念を強化してしまう。

- フォーレ効果（Forer Effect）

  一般的で曖昧な記述を自分に特有のものとして受け入れる現象。占いの多くのアドバイスは広範な人々に当てはまるように設計されており、個々人がそれを自分専用に解釈しがち。

- 結論飛躍（Jumping to Conclusions）

  十分な証拠なしに結論を急いでしまう思考パターンで、ここでは占いの一般的なアドバイスを自分に直接適用することが該当する。

開発の現場には、占い師はいないけど、”シッタカ” がいます。どこかで聞きかじったような浅知恵を、まるで持論かのように語ってしまう鼻につくやつです。シッタカ野郎は可愛がってもらえないので、私は、シッタカを見かけたら、slackで、こっそり「シッタカ出てたぞ、酸っぱいから気をつけろ！」言ってあげるようにしてます。(笑)

認知のゆがみを誘発してしまうものは、他にもあるでしょう。これには、どのような対処をしたらよいのでしょうか？

ChatGPT は斯様に申してます。

- 客観的な視点を持つ:

  占いのアドバイスがどれほど一般的なものであるかを認識し、それが自分に特有のものかどうかを冷静に判断する。

- 証拠を検討する:

  占いの内容が具体的に自分に当てはまる証拠があるかを確認し、根拠のない部分は疑問視する。

- 他の視点を取り入れる:

  友人や専門家と相談し、異なる視点から物事を考えることで、バランスの取れた判断を促す。

- 自己認識を高める:

  自分自身の思考パターンや信念を理解し、どのようにして特定の情報を解釈しているかを意識する。

そろそろ本題に入りますが、上記のうち、「客観的な視点を持つ」というのは、実は、とても大事で、効果が大きいと言われています。時間管理術の書籍や WEB の記事は検索すると大量に出てきます。（メモしてないので、URL を載せることができませんが）多くの時間管理術では、まずは、自己を知ることから始めましょうとあります。ノートや Excel を用意して、作業の始まりと終わりの時間を書いて、どんな作業に何分かかったのかを、正確に知って、それを客観的根拠として、問題分析しないとダメだとあります。バイアスのかかった時間管理では、見当違いな改善策が発案されて効果が出ないことが多いのだと。

ゆえに、時間の記録は、1 日の終わりとか、翌日の朝一に、まとめてつけるのではなく、作業の切り替わりのタイミングで、即時つけるようにしないといけません。

作業の丸め方も、なんらかのルールを基にして、一貫性を持たせないとブレます。

でも、エンジニア職は、横着者が多いので、こまめに時間を付けるということが苦行です。続かないことは、目に見えてます。

だから、これを自動化しようと、時間計測用のツール開発に着手しました。

「MINR desktop」

仕組みは単純で、デスクトップに常駐して、PC 上で起動している各種アプリの Window がアクティブになったタイミングで、アプリ名と Window タイトルを記録していくツールになっています。

Windows の標準機能として搭載予定の「Recall」も PC 上の操作履歴を記録するものになります。標準機能にあるなら、それを使えばいいじゃないかという短絡的な話ではありません。データは手元で自由に使えてこそ価値があります。そして「Recall」が標準機能に搭載されるということは、ここに需要があるということだと捉えています。
MINR desktop は Electron をベースにしているので、Mac や Linux でも動くことを目指しています。単に Window タイトルを記録するだけでも、その Window タイトルを正規表現で、パターンマッチングすることで、以下のような分類をすることが可能です。

- **アプリ名=Excel.exe && Window タイトル=/.*仕様書.xlsx/**

  **→ 作業分類: 設計**

- **アプリ名=Code.exe （VSCode）**

  **→ 作業分類: 製造**

- **アプリ名=Code.exe && Window タイトル=/ .*.test.ts$/**

  **→ ラベル: ユニットテスト実装**

・・・

このように、アプリ名と Window タイトルだけでも、わかることがたくさんあります。

ここで、現在実装中の機能を少しご紹介します。

- プロジェクト、タスク、カテゴリ、ラベルの登録機能

  手動で登録する機能は実装済で、このあと、GitHub のリポジトリや Issue と同期する機能を作成予定です

- 予定と実績の登録機能

  こちらも手動で登録する機能は作成済です。

  予定については、Google カレンダーとの同期機能は作成済です。

- 実績の自動登録

  アクティビティログ(*1)をパターンマッチングしてプロジェクト、タスクを自動分類しながら実績として仮登録す

る機能です。同じく、自動的なラベル付けもします。

*1: Window タイトルの保存をアクティビティの保存と呼ぶ

- 予定の自動登録

  未完了タスクを自動的に予定として仮登録する機能です。

- その他

  まだ十分なものにはなっていませんが、分析のためのグラフ表示の機能と、CSV 出力の機能などが作成されています。ポモドーロタイマーもあります。

このツールを使うことで、「己を知る」ための客観的なデータが取得できます。”振り返り”のときには、占い師のアドバイスではなく、客観的な分析に基づいた改善策を出せるようになるのではと思います。

MINR desktop は、当初は、SaaS で展開したいと計画していたのですが、このツールが扱うデータには、個人情報や企業秘密などセキュリティリスクがあります。弊社の企業レベルというか開発体制では、まだクラウド上での共有をしない方がよいと思ったので、スタンドアロンで動くように方針を改めました。そのことが、逆に顧客企業からの貸与 PC にインストールして使うことのハードルが下がったと思います。

マネタイズは、SaaS での展開をしない代わりに、このツールを企業内限定アプリとして制作する SI 案件化で展開することを考えています。ご要望のある企業様がいらっしゃいましたら、ぜひ、お声がけください。

とはいえ、実は、マネタイズは、あまり期待していません。OSS で公開しているということもあり、エンジニアの方々に、自由に使ってもらいたいと思っています。このツールに込めた弊社の狙いは「アルタスファイブを知ってもらうこと」です。

初版のリリースは、年内を予定しています。
準備整い次第、別途ご案内させてください。

# 10 開発で変わったスクラムの認識

**開発部 柳**

リモートワークが社内で主流になり、あまり人と対面で会わないことに危機感を感じ、最近はボルダリングやボードゲームカフェのイベントに参加など、人と接する機会を増やしてコミュニケーション能力を養っています。

開発部の柳です。

今年はプライベートで色々な挑戦をして自分磨きに力を入れていた↑と書きましたが、業務面でも今まで経験してこなかったことが多く勉強になる年でした。特に印象的だったのは、9 月頃から参画した自社開発で、アジャイル開発手法の 1 つである『スクラム』の開発でいろいろな発見をしました。

今回は、自社開発で行っていた『スクラム』について、私が学んだことを書かせていただきます。

**【スクラムについて】**

アジャイル開発におけるスクラムは、動くプログラムを 1 週間~4 週間のスプリントと呼ばれる期間で開発を行い、作成されたプログラムを評価する開発手法です。開発では、スプリントの回数を重ねてより良いソフトウェアを作ります。ここで、重要になるのがスプリント終わりの評価です。評価は開発しているソフトウェア以外に、スケジュール通りに進んでいるか、メンバーの生産性に問題ないかなど、チーム全体の評価も行います。つまり、チーム内でふりかえりを行うということです。

ふりかえりを行うには、目標設定が必要です。ここでの、目標設定とは開発におけるゴールで、どのようなソフトウェアにするのか、開発はどれぐらいの期間で終わらせるのかなどを指します。私が携わっている開発では、目標を明確にかつチーム内で共有するために、『インセプションデッキ』という手法で設定を行いました。『インセプションデッキ』とは、開発における 10 個の設問に答えることで目標を設定する手法で、チームの立ち上げなど初めに行うことが推奨されています。ここで重要なのは、チームの全てのメンバーが目標について理解することにあります。目標についてチームの共通認識を作ることで、ふりかえりを行う際に、目標に対してどれだけ進んでいるかを評価できるだけではなく、問題点についてどのように解決すればいいかを考える指標にもなりま

す。

**【スクラムを行って気づいたこと】**

スクラムは『アジャイル開発の代表的な手法』と呼ばれるように、わかりやすくシンプルなフレームワークという認識でしたが、実際に実施してみるとチームでのふりかえりの印象が強く、チーム間のコミュニケーションの起点であり、チームの成長に優れたフレームワークという認識に変わりました。

ふりかえりでは、チームメンバーが全員参加して行うためチームでの課題について考えることが望ましいと認識していましたが、個々の課題についてもその場で共有し、問題解決のための意見やチームでの改善方法を考えるなど、チームから個人まで幅広い課題について考えることができました。

スクラムは『アジャイル開発の代表的な手法』と呼ばれるのが一般的ですが、その本質はチーム内の不安に思っている課題点についてチームで解決し、チーム全体の成長とコミュニケーションを円滑にするためのフレームワークであると私は考えます。

**【課題点と今後の展望】**

スクラムのふりかえりは、チームで行うことが望ましいですが初めのスプリントではそれが難しいと感じました。初回のふりかえりでは、出会って間もないメンバー同士で行うため、どうしても意見が出にくい環境になってしまいます。個人の課題点であれば尚更で、どうしても外に出そうとは思いません。また、最初はインセプションデッキなど目標を設定しているので、チームでの改善点をどうにか考えないと思ってしまい、幅広い意見がどうしても出ないように感じました。

そのため、ふりかえりを行う際に、「自分から不安に思っていることを吐き出す」、「些細な問題点でも共有する」という意識をもつことが大切であると思います。スクラムのふりかえりは回数を重ねるため徐々に自分から問題点を表に出すようになりますが、早期に表に出して発言できるようになれば、それを起点にその後のふりかえりでより良い改善策やチームでの行動ができるようになると考えます。

最後に、私はスクラムにおける最も重要なことはふりかえりを有意義なものにすることにあると思います。開発では１人で解決できないことが多々あり、それを解決するためにふりかえりを行い改善します。リモートワークが一般化し、対面や会話での意見交換が少なくなりつつある現在、人と話すことが苦手になっている人は多くなっていると思います。他人の苦手を解決するのは個人の問題なのでどうこうすることは難しいと思いますが、そういった人を議論の場に誘い込むことはできると考えています。そのために必要なことは、コミュニケーション能力であり、どれだけ開発を熱狂させることができるかが重要になると思いました。

**会社からの一言：**

自分の実力やスキルをチーム全員に共有することで、得意不得意が把握できて、適材適所で仕事をすみ分けることができるので、そういった自己開示はした方がよいですね。

いつもの顔ぶれの中で仕事をしているので、ある程度の心理的安全性が確保できていることを前提にしてよいと思います。そのために、持ち帰り型の仕事になるようにしているので。（開発部 佐藤）

## 10 お菓子紹介

**管理部　宇野**

毎度のことながら、新たなお菓子を探すべく、インターネットの海を奔走する宇野です。

今期も、日ごろの感謝の気持ちを込めまして、社内にて選りすぐったお菓子を紹介させていただきます。

今期はシンプルです。マカロンをたくさん探して、たくさん食べました！マカロンはひとつひとつはサイズが小さいので、ぺろりと食べられる印象でしたが、そんなわけにもいかず。基本的にはメレンゲと砂糖とアーモンドを混ぜ合わせた焼き菓子だそうで、あの小さな見た目に甘さ・製造の難しさ・見た目の美しさ・繊細さがぎゅぎゅぎゅっと詰め込まれていて、2 個…頑張れば 3 個食べられるかなという満足感の高さでした。

そして、難しいところは、気を抜くと見た目が似ている種類はどれがどこのお店のものかわからなくなってしまうところです！社内で配るときにも付箋をはり、間違えないように呼び掛けながら試食をしました。

ですが、選ばれたものは、見た目で判別がつくかわいらしい 2 種類になりました。

**▼JOUVAUD：プロヴァンスマカロン**

こちらは上記で説明したアーモンドではなく、ヘーゼルナッツが使用されています。砕いたものが混ぜられているのでごつごつとした見た目の一味変わったマカロンです。食べた印象もごつごつとしたヘーゼルナッツの主張が強く面白い食感です。見た目は全部同じなんですが 4 種の味があり、しっかり裏を見て味を確認しないと判別ができません！どの味もナッツの風味がしっかりとするので、ナッツ系の味や香りが好きな方には刺さるおやつだと思います。

**▼クリオロ：ヨーヨーマカロン**

こちらは「サクふわ食感」にこだわった、まるでおもちゃのヨーヨーのようなコロンとした見た目のマカロンです。通常のマカロンと材料は大きくは変わらないらしいですが、混ぜ方にもこだわりがあり、パティシエの繊細な技術がきらりと光る一品です。口の中でほろほろりとほどける食感は宇野の一押しです。味は 5 種ほどあるようですが、弊社佐藤の推しはショコラ一択！ということで、ショコラにのみ絞りました。都内に 3 店舗あるそうなので、是非調べてみてください。

マカロンは贈答品にするには、つぶれやすいことや日持ちの点から選ぶのが少し難しいお菓子かもかもしれません。しかし、おいしいならば、食べてみよう、です。

おいしいという噂を聞けば、レッツチャレンジあるのみです。

**会社からの一言：**

新宿高島屋の催事場でショコラフェアをやっていたときに、ちょっと寄って、何個か買ってきたんですけど、そのときにヨーヨーマカロンを食べました。おいしかったので、宇野にも奨めたんですが、あらためて食べてみて、ヨーヨーマカロンを”俺のオヤツ”の殿堂入りにしました。

（オヤツ食いの自称プロ 佐藤）

**次号につづく**