アルタスファイブ通信 2022 年冬号

「ちょっとだけ変なことをしている会社＝技術変態アルタスファイブ！」です。

今年も、皆様方からのお仕事のご依頼を賜りまして、堅実な一年を送ることができました。いつも、ご愛顧いただきまして、ありがとうございます。

円安が止まらないですが、私たちの仕事への影響はどれくらいあるのでしょうか？ システム開発は、企業の設備投資ですから、影響が出るとしたら、来年以降でしょうか。でも、生産性が低いと言われている日本ですから、DX 化をとめたらダメですよね。

私たちも日本に貢献するために、しっかり技術を磨いて、1 つでも多くの企業のシステム化を支援していこうと思います。

さて、2022 年の下半期を振り返って、アルタスファイブの仕事っぷりをご紹介させてください。

今号では、普段の仕事の中で、「学び」と「生産性」について、どのようなことを気にかけているか、そのあたりをテーマに記載しています。

目次

01 チーム内の技術差改善の難しさについて

開発部：神崎

そろそろ、PC の排熱が頼もしい季節になってきました。最近、機械学習を使ったサービスを一般の人が使用して、イラストや文章を作るなどができるようになり、ついに DeepLearning が一般化してきたなと感じます。

プログラミングにも機械学習でコメントを書くとプログラムを生成してくれるものができましたが、ライセンス面などの問題で実用としては微妙なものになってしまったのが残念です。私個人としてユニットテストの実装などを支援してくれる AI が欲しいところではあります。まだまだ、機械学習を使ったサービスには、汎用的な面はまだまだな印象(それでも今の機械学習は凄いことではありますが)や法的や倫理的な面でのいろいろな課題などもありますが、AI サービスで世の中が良くなっていくことを願っています。

さて、本題として、今回は学習や生産性がテーマになっています。

少し前に参画していたプロジェクトではスクラム方式の開発

でした。スクラム方式の場合、できるだけタスクの属人化を防ぐために、ドキュメント整備や研修などでチーム内の技術レベルを平坦化していく必要があるそうです。

実際はドキュメントを整備するだけでは、技術レベルの平坦化は非常に難しかったです。ドキュメントを作り慣れていない事もあり、詳細な説明を省いてしまって、知識レベルに差があるとうまく手順通りにできないなどが発生しました。結果的には作業の初回時はすでに作業をしたことがある人と画面共有をしながら、作業を教え、手順書を改善していく方式になりました。作業を既にしたことがある人の作業時間が減るので、時間管理が厳しいスクラムでは作業を既にしたことがある人の分の時間もタスクに含めるようにして対策をしました。

本来であれば、チーム内部でメンバーにまとめて研修のような形で教えるのがベストなのでしょうが、時間が無いプロジェクトではなかなか難しく、後から考えるとそういった技術研修の時間も考慮してスクラムを計画する必要があったのだと思います。

特に技術要件が多いプロジェクトの場合、技術レベルの平坦化は重要でした。プロジェクトが進めば進むほど、技術レベルの差が広がりすぎて、プロジェクトが忙しいとタスクを技術力があるメンバーに一任することになり、作業に偏りが発生してしまうことになります。結果的にはタスクが並列で進みづらいので、遅れが発生してしまう原因にもなっていたように感じます。

結論としては、個人のプライベートの時間を使って学習するというのは難しいので、できることならチーム内で学習時間をいくらか確保したり、外部の研修を受けてもらったりすることがよさそうです。機会があれば試してみたいところです。

クラウド技術を学ぶ

前項のプロジェクトでは技術要件にクラウド系(AWS、GCP)が多くありました。私自身はクラウドサービスに触れたのが初めてでした。手順書があるのでその通りには行うのですが、あまり理解はできなかったので、興味もあり、学ぼうと思いましたが、クラウドサービスは従量課金制が多いので学習するのに一定額ではないので、ちょっと手を出しづらいところがありました。(もちろん、使いこなせていれば安くは済むとは思います)

さらにサンドボックス的に触れられたらいいなと思い、色々探していたところ、Microsoft Azure の公式(https://learn.microsoft.com)に講座付きで条件はありますが無料でクラウドサービスを利用できる研修サイトがありました。

これを使ってクラウドについて、学んでいってみようと思います。

もし、機会があれば、次回は Azure の研修サイトの使用感も書ければいいなと思います。

あとがき：

今は、いろんなサービスを組み合わせて、システムを作る時代になりました。新しいこと取り込んでいくマインドセットも不可欠ですね。（開発部 佐藤）

02 入力デバイスの最適化

開発部 川野

30 代半ばあたりから、肩こりで右腕が上がらなくなることがしばしば発生して困っていました。入力デバイスの最適化は生産性のアップにもつながるということで、今回はその話をしたいと思います。

ポインティングデバイス

右肩のみの症状だったので、初めはマウスが原因かと考え、以下を試しました。

・トラックボール（Kensington K72327JP）に変える

・右腕でなく左腕で操作する

どれも肩こりは改善しませんでした。ポインティングデバイスは肩こりの原因ではなかったようです。余談ですが、左腕でトラックボールを使うのはマウスより楽に感じたので、現在もこの組み合わせで使っています。

キーボード

次にキーボードを試しました。

・エルゴノミクスキーボードの使用

マイクロソフトのエルゴノミクスキーボード（L5V-00030）を使用してみました。こちらも肩こりには効果ありませんでしたが、打ち心地が好みだったのでボディが割れるまで２年ほど使いました。

・左右分離キーボードの使用

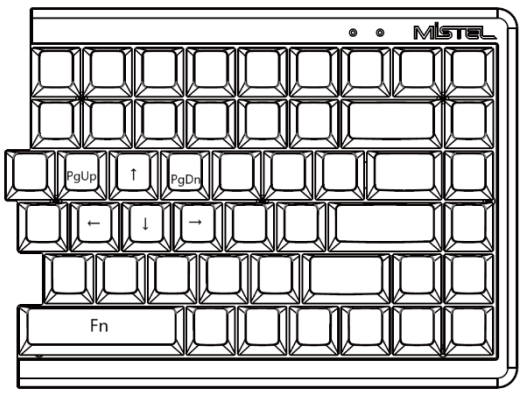
このタイプは、英語配列で特殊配列が多く、値段も高めなのがネックでしたが、１年ほど前に MISTEL MD770 というキーボードを購入しました。こちらは肩こりに改善が見られました。私の場合、両腕を中央に揃え続けるのが肩こりの原因だったようで、このキーボードを利用してから腕が上がらなくなることがなくなりました。

ハードウェアマクロ機能による入力の効率化

肩こりは解消できたのですが、MISTEL MD770 はやや特殊配列でカーソルキーがホームポジションからズレているのが使いにくいと感じていました。この問題はハードウェアマクロを利用することで対応できましたので、一例としてご紹介します。

左右分離キーボードはその構造ゆえ、スペースバーが２つ存在します。そのため片方を修飾キーに割り当てても特に不便はありません。私は右のスペースバーを Fn に割り当て、Fn＋任意キーをカーソル移動となるようにマクロを設定しています。これでほぼホームポジションを移動することなくカーソル移動が可能になります。

Fn 押下時のイメージ



また、このキーボードは英語配列なので、日本語切替は Alt+`（メンドクサイ）ですが、CapsLock キーにそれを割り当てることで１キー入力で切替できるようにしています。

ソフトウェアでもこういったことはできそうですが、OS にインストールが必要なのと、スペースバーの左／右で別々のキーが設定できるか？などを考えるとハードウェアであるに越したことはないように思います。

肩こりに悩んでいる方は、ハードウェアマクロ機能付きの左右分離キーボードをおすすめします。

あとがき：

私も、東プレのキーボードを使っているので、同意です。

前号と読み比べると、同じような記事の刷り直しになっちゃってるところがあるので、こういう記事もいいですね。図や写真もあって、いいです。（開発部 佐藤）

03 エンジニアの生産性について

開発部 板子

生産性とは何か

いろいろな考え方があるかとは思いますが、一般的には労働者１人あたり又は１時間に生産できる成果を数値で表したものになると思います。

物理的労働生産性と付加価値労働生産性

物理的労働生産性

物的労働生産性＝生産量/労働量で計算します

100 のモジュールを 5 人で作業したとします。

物的労働生産性（個／人）＝100 モジュール／5 人となり

この場合の物的労働生産性は 20 モジュール／人です。

5 人で 200 のモジュールを作成した場合、1 人当たりは 40 となり、同じ条件でたくさん作れれば生産性は高いとなります。

付加価値生産性

付加価値労働生産性＝付加価値額（売上-諸経費[燃料費、材料費、運送費など]）/労働量で計算します。

物理的労働生産性で例とした、100 のモジュールの金額が 100

万円とすると

付加価値労働生産性（円／人）＝100 万円／5 人

付加価値労働生産性は、20 万円／人となります。

諸経費についてはここでは考えないことにします。

向上のメリット

少ない人材で利益が上がることになりますし

数時間の短縮につながれば、労働時間が減らせますのでライフワークバランスの向上に繋がります。

短縮できた時間でスキルアップへの勉強もできるようになります。

逆に低いと残業が増えるなど、悲しいことになります。

労働生産性の向上に向けて

私がリーダーとして係わっているプロジェクトにサイト運営の保守があります。

保守ももちろんですが、どちらかと言えば小規模な改修の繰り返しが多いです。

一般的な生産性向上へのアプローチを、こちらのプロジェクトでの試み

（というよりできていないことがほとんどですし、目新しい話はないかもしれませんが。。。）

交えながら進めたいと思います。

現在の物理的労働生産性の把握

前述の例でたとえると、1 人でどれだけのモジュールを作れるのか

という現状の把握が必要になりそうです。

これが出来ないと生産性が向上したのかがわかりません。

プロジェクトでは、予定工数と実績を管理している程度です。

業績評価の指標を作る

KPI ですね、これ設定することで目標が数値化され、全員が共通の指標を把握できるためメンバー全員の意思統一も容易になるというメリットがあります。

プロジェクトでは、目標とするべき指標を模索している状態です。

一般的には SMART(Specific：明確性、Measurable：計量性、Achievable：達成可能性、Relevant：関連性、Time-bound：期限)

等を踏まえて設定すると良いようです。

ボトルネックの洗い出し

効率の悪い部分を探します。

・よく遅延する作業

・発生頻度の高い作業

・同じ失敗が起こる作業

・たくさん時間を使っている作業

これらを把握することで作業を改善していきます。

プロジェクトでは、これについては「ふりかえり」を行うことで積極的に行えています。

上記の作業を洗い出しています。

何が問題だったかを話し合い必要であればナレッジとして Wiki などで共有していたり

お客様とのコミュニケーションの必要があれば共有、相談し改善するようしています。

今後取り組み

取り組んでいたいこととして、上記、物理的労働生産性の把握・KPI ももちろんですが

メンバー個々のスキルも異なることの改善にも取り組んで行きたいと考えています。

たとえば

・画面周りの作業は問題ないが、バッチ処理は作業したことがない

・PHP やバックエンドは得意だけど、JavaScript やスタイルシートは苦手

なんて感じです。

これについては、満遍なくメンバーに作業を割当て知見のあるメンバーのサポートを受けながら慣れてもらい

プロジェクト固有のスキルマップなんて物を作っていきたい模索しています。

改めて振り返るとできていないことばかりですが

できることから取り組んでいき、メンバーが「向上のメリット」である

スキルが上がり、エンドユーザーにも安心を与え

メンバーのライフワークバランスが良い

そんなプロジェクトにしてきたいです。

あとがき：

SCRUM のベロシティで、計測するのも 1 つのやり方ですね。

そういえば、プロジェクトの始まり時期と、終り頃の生産性は、違うハズだけど、スケジュールを立てるときは、生産性は通期で同じ前提にしてますね。プロジェクトが始まってすぐに遅延が出て、“キャッチアップします“地獄に陥ることがあるけど、原因はコレかも。（開発部 佐藤）

04 データウェアハウスの特徴とデータマートとの違いについて

開発部 檜山

リモート勤務での、運動不足、特に筋力低下を解消するため、ここ 1 か月ほど、Switch のリングフィット/Fit Boxing2 をほぼ毎日やっていますが、特に、腹筋がなかなか付かず、筋肉ってどうやったら付くんだっけ？と心が折れかけています。
開発部の檜山です。

現在参画させていただいているデータ分析基盤構築の案件で、データウェアハウスのデータを作成することになり、その特徴やデータマートとの違いを学習していますので、初歩的なレベルではありますが、紹介させていただきます。
※まだ勉強途中であり、データウェアハウスとデータマートの境界線は曖昧な部分があるということなので、不正確な部分もあるかもしれませんがご了承ください。

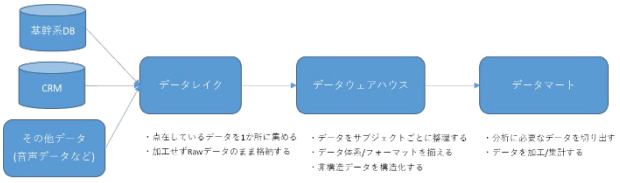
◆そもそも、データ分析基盤構築とは？

データ分析基盤構築とは、分析者がデータを分析するための基盤(環境/システム)を提供/構築することです。(名前のままですね)
そのため、直接データを分析して規則性や法則、事業の課題解決をしていくというよりは、そのために有用なデータを作成したり、インターフェースを提供することが仕事になります。

ただ、もちろん、そういった有用なデータを提供するためには、分析者が欲しいデータや、分析したくなるようなデータを検討して、設計/構築する必要があるため、分析する力があった方がより良いのは言わずもがなです。

◆データ分析の流れ

データ分析のおおまかな流れは下記の図の通りです。
※あくまで一例です。



以降、データウェアハウスは DWH と表記します。

まずはデータを収集します。各所に点在しているデータをデータレイクに集め一元管理します。この時点では基本的にデータの加工はせず Raw データのまま格納します。

集めたデータは、データ体系が異なっていたり(例えば、同じ商品でもシステムによってコード体系が異なっているなど)、非構造データも含んでいたりします。そのため、フォーマットを揃えたり、非構造データを構造化したり
目的(サブジェクト)ごとにデータを整理したりすることで、分析しやすい形に加工/整形し、DWH に格納します。

加工/整形した DWH のデータをもとに、分析を行い、必要なデータのみを抽出したり、データどうしをかけ合わせたり、集計したりして、データマートを作成します。データマートのデータをモニタリングし、次の施策を検討したり、施策結果の評価をしたりします。

※実際には、データレイクのデータで分析することや、DWH を作らずにデータレイクからデータマートを作ったりすることもあるので、あくまでデータ分析の流れの一例です。

◆3 層構造

データ分析基盤では、データ分析の流れに合わせ、データを 3 層構造で設計することが一般的なようです。

・データレイク
・データウェアハウス
・データマート

小規模な分析ではデータレイクとデータマートの 2 層だけで十分なこともありますが
企業が蓄積しているデータが増え、必要なデータを素早く見つけ、活用するために、DWH の活用が注目されているようで

す。(DWH 自体は最近提唱されたものでは無く、データレイクより前に提唱されたもののようです)

データレイクとは？

・データの湖。
・各所に点在しているデータを 1 か所に集め一元管理する。
・データベースのような構造データだけでなく、ログや、画像、音声などの非構造化データでも OK。
・データは加工せず Raw データのまま保存する。(最低限の加工をする場合もあります)

データレイクのデータは加工せず、Raw データのまま保存しておくことが特徴です。これにより、バックアップとしての機能や、利用者の汎用性/柔軟性(利用用途に合わせて自由に加工/利用できる)を持たせることができます。

データウェアハウスとは？

・データの倉庫。
・データレイクのデータを目的別に整理したデータ。
・提唱者の William H.Inmon 氏によると、「意思決定するために、目的別に編成、統合された時系列で、削除や更新を行わないデータの集合体」と定義されています。

整理すると下記 4 つの要件に分けられるようです。
・サブジェクト志向(subject oriented)
・統合(integrated)
・時系列(time-variant)
・不揮発性(non-volatile)

これらの 4 つの要件を 1 つ 1 つ見ていくと DWH の特徴/どういう設計が適しているかが見えてきそうです。(サブジェクト志向と統合は独立した要件というよりは一緒に満たされる要件に思いますので 1 つにまとめています)

・サブジェクト志向(subject oriented) / 統合(integrated)
　注文や商品、顧客などのサブジェクトごとにデータを整理します。
　異なるデータソースから集めたデータはコード体系が異なっていたり、単位が異なっていたりして統合的な分析ができない可能性があります。そこで、データ体系やフォーマットを揃え、例えば、「商品」であれば、異なるデータソースから取得したデータでも同じ「商品」データ群として扱うことができるようになり、分析しやすくします。また、複数システムで同じデータを保存していることもあるので重複を排除することも必要です。(例：アプリ側と CRM で同じ顧客データを別々に保持している場合など)
　純粋に「商品」というデータを扱うためにはどんなカラムが必要なのかを検討/精査する必要もあります。

・時系列(time-variant)
　通常のデータベースで必要なデータは、基本的に最新のデータになるため、過去のデータはあまり利用しません。(例：住所といえば、最新の住所になる)
　一方で、データ分析では、過去のデータも分析対象とすることで、最新のデータと過去のデータを比較したり、データの変遷を分析することもできます。そのため、変更がされた時点や、日毎のデータを積んでいき、時系列データとして保存していくことが必要になります。

・不揮発性(non-volatile)
　時系列で述べた通り、過去の履歴をすべて記録し、そのデータを活用するので、削除や更新は行わず、積んだデータをそのまま残しておく必要があります。

DWH ではこれらの要件を満たした上で、どういうデータ(中身)を作るかを検討する必要があります。「どういうデータを作るか」は、「どういう分析をしたいか」によって決まるので、事前にどういう分析をしたいかを検討しておく必要があります。

データマート

・データの小売店。
・DWH のデータを一部切り出したデータ。
・DWH のデータどうしをかけ合わせたり、集計したりしたデータ。
・DWH よりも特定の部署/サービス/目的に特化したデータのため、比較的構築が容易で、分析もしやすいデータになる傾向があるようです。

◆DWH とデータマートの違い

DWH とデータマートの特徴から、その違いを表にまとめました。

	DWH	データマート
特徴	データレイクのデータを加工/整形/クレンジングして分析しやすくしたデータ。 サブジェクトごとに整理したデータ。	DWH のデータの一部を切り出したデータ。 特定の部署/サービス/目的に特化したデータ。

	DWH	データマート
分析対象	広い	狭い
開発工数/期間/難易度	そもそもどういうデータを作ればよいか(=分析者が使いたいデータ)の検討/設計が難しい	どういうデータを作れば良いかが明確なため、比較的早く構築できる
データ集計	集計せず、ログをそのまま入れてあげるのが望ましい	月別売上集計など集計したいりする(集計しない場合もある)
分析難易度	横断的な分析が可能で、より高度な経営分析が可能(その分、分析は難)	必要な情報のみを取り出しているため分析が比較的容易

データマートは特定の利用用途に特化したデータであるのに対し、DWH はサブジェクトごとに整理したデータでより汎用的で、範囲が広いなど、データに持たせる「役割」が異なっているようです。
ただ、元々は明確に分けずに使っていた部分もあり、明確な境界線は無くグレーゾーンな部分もあるようです。

◆まとめ

今回は DWH の特徴と、データマートとの違いについてまとめてみました。
企業が保有するデータ量が多くなり、必要なデータのみを素早く抽出したり、気づきを得るためには今後も必要になってくるものだと感じました。

DWH の満たすべき要件(サブジェクト志向、統合、時系列、不揮発性)はそれほど難しく無さそうですが、どういうデータ(中身)を作るか？に関しては、そもそもどういう分析をしたいかを考えなければならず、そのためには統計学などのデータを見たり、分析する力や、データソース/業務理解なども必要になると思われるので、こういった力も養っていきたいと思いました。まだ案件内では DWH のデータを作っている途中で、これから活用していこうという段階なので、その際にまたノウハウや知見が得られればご紹介させていただければと思います。ありがとうございました。

参考

- ［エンジニアのための］データ分析基盤入門　データ活用を促進する！ プラットフォーム＆データ品質の考え方 斎藤友樹著
- IT トレンド データマートとは？データウェアハウス・データレイクとの違いを解説
  https://it-trend.jp/dwh/article/149-0005
- データのじかん データマートとは？
- データウェアハウスとの比較から話題のデータレイクも含めて解説！
  https://data.wingarc.com/what-is-data-mart-12075
- IT トレンド DWH（データウェアハウス）とは？要件・活用事例まで詳しく解説
  https://it-trend.jp/dwh/article/149-0001
- データウェアハウス設計とは？DWH の特徴や分析の流れについても解説
  https://www.cloud-for-all.com/blog/what-is-data-warehouse-design
- データマートと DWH 違い
  http://www.mh.rgr.jp/memo/sv0038.htm
- RECRUIT Member's Blog 分析者から見た使いにくいデータ基盤の話
  https://blog.recruit.co.jp/rtc/2018/12/25/datakiban_-risou/
- Think it データレイクとストリームデータ処理を理解する
  https://thinkit.co.jp/article/17908
- DWH（データウェアハウス）とは？データベースとの違いや特徴も解説！
  https://www.dal.co.jp/column/b-dwh/
- データウェアハウス(DWH)の 4 つの要件について
  https://dev.classmethod.jp/articles/data-warehouse-basics-02/
- DWH（データウェアハウス）とは？DB や BI との違いから基本を理解しよう
  https://www.pasonatech.co.jp/workstyle/column/detail.html?p=9085

あとがき

分析系のシステムも、実績が増えてきました。得意分野の 1 つと言ってもよいかもしれません。ただ、今までは、基盤を作るところや、ETL、データ抽出処理の実装などを担当する役割が多かったので、データサイエンスの一端を経験できるとよいですね。（開発部 佐藤）

05 「振り返り」とその手法について

開発部 藤原

一言で生産性を上げるって言っても、具体的にどうすればいいのかは難しいものです。

最近その生産性を上げるための取り組みとして、定期的にプロジェクトメンバー全員で業務を振り返るというものをしています。

この振り返りの一種として有名なのが「KPT」です。Keep（継続すべきこと）P（課題となること）Try（取り組むこと）に行動をひとつひとつ分類して整理する手法ですが、この手法はすでに実践しているところも多いと思います。

そこをさらに深堀りしてより効率よく振り返りを行うにはど

うすればよいかを工夫することで、さらに生産性を向上させていきます。

その際に**アジャイルなチームをつくる ふりかえりガイドブック**という本を参考にしていろいろ試してみました。その中には様々な振り返りを改善する方法が載っていて、書かれている全ての手法を試したわけではありませんがその中で自分たちで試した一部を KPT ベースで考察したいと思います。

SMART な目標

Specific Measurable Achievable Relevant Timely／Time-bounded という単語の頭文字をそれぞれとった略であるらしいですが意味だけを略すと「具体的で現実的な目標」ということで、KPT では T の部分が改善されることになります。

たとえば「残業を減らす」という目標よりも「ノー残業デーを作る」という目標のほうが達成しやすく、達成できたかどうかも確認しやすいという話です。

これをすると Try がすごく管理しやすくなるので、KPT をずっと続けてきたのはいいが Try することを多くしすぎて手が回らない、という状況のときにこれをするとより有効に業務を改善していくことが可能になります。

ドット投票

これは Try の中から振り返りに参加してる人たちで投票を行い、どの Try に対して優先して取り組むかを決める手法です。

これも SMART な目標と同じく、Try の数を増やしすぎて手が回らなそうな場合に効率よく消化していく手法になります。目標ごとに重みをつけることで優先順位が明らかになり、ひとつひとつ消化していくことができるようになります。

Try を消化する方法が多いのはそれだけ KPT のキモは Try の部分にあり、これをしていくこと自体が重要だということだと思います。

＋／△

振り返りの振り返りを行うことで、振り返り自体をさらに効率化していくという手法です。

振り返りのあとに参加した全員で意見を出し合ってどの手法がやりやすかったかなどを話し合い、必要だった手法と不必要だった手法を選別していきます。

各々がなにか思っていても普段言わないこと言う場を作ることでいろいろな手法を試して、プロジェクトチームに合った振り返り方法にたどりつくまでの時間が少なくすみます。

KPT による振り返りを行うほかにも生産性を上げるための取り組みとして一般的なものはペーパーレス化などが挙げられますが、そういった全体的な管理よりも一人単位での業務改善案を考えることができるので個人的には自分に合った改善案を探すという意味でこちらのほう良いと感じました。

あとがき

「アジャイルなチームをつくる ふりかえりガイドブック」という本は、ふりかえりを実施しているチームには、読み合わせをしてもらって、改善してもらいました。私自身も、その改善による変化を感じています。正しく運用しないと、ただの儀式になってしまうので、こういう活動も時々必要だと思います。（開発部 佐藤）

06 ツールの使い方と考え方

開発部 柳

入社 3 年目にして、二回目のアルタス通信の執筆をさせていただきました開発部の柳です。久しぶりに執筆してみましたが、書き方が思い出せず論文を書いていた学生時代のノリで書いていたら少し硬い文章になってしまいました。最近は試験勉強や読書など、文字に触れる機会は多いのですが、書くのと読むのは全く違うものだと感じます。今回は作業の効率化について、私個人の考えを書いてみたので楽しんでいただけたら幸いです。

今年の開発では去年の作業よりも比較的に調査を行うことが多くあり、その中で効率化という点で linux にて grep を利用した調査が印象的に感じました。grep は linux で使用できる検索コマンドで、開発に携わってきた方々から見れば特に話題にするようなものではありませんが、今年で開発 3 年目になる身としては、効率化という点で特に役立ったツールです。とある開発にて、ソースファイルから特定の文字列を使用している箇所を調査することがあり、その際に grep は大いに役に立ちました。調査の始めはエディターツールの検索機能を利用して調査を進めていましたが、対象の文字列が膨大かつ検索結果を一つ一つ記載しなければならないという、非常に効率が悪い状態に陥ってしまいました。その日の夕会にて grep を使用すればいいという指摘を受け、コマンドを作成し調査を行ったところ、検索結果をテキストファイルに残しつつ、さらに結果を利用して必要なデータを作成することが出来たため、その後の作業で作業効率は大きく改善されました。この話で重要なのは、grep を使用すれば効率的に調査ができるという点ではなく、ツールを正しく理解し使用することにより効率的に作業をすることが出来るという点が最も重要だと考えます。

grep は特定の文字列を調べるという点で、正規表現により様々なパターンの文字列を調べることが可能で、結果をテキストファイルに保存できるなど便利な点が多くあります。しかし、grep では特定のソースコードを探すことはできても、そのコードと関連するクラスやメソッドを調べることはできません。ソースコード同士の関係性を調べるのであれば、メソッド間の関係性を検索することができるエディターツールを使用した方が効率的に調査することができます。このように、ツールにはそれぞれ向き不向きがあり、開発ではそれらをうまく使い分けることで効率的に作業することが、エンジニアに求められる技術の一つだと思います。

しかし、ツールを上手く使うにはどうすればいいか考える必要があります。1 案として、様々なツールの知らない機能や新しい使い方を模索するなど、ツールを使う訓練をすることは上手く使うことの第一歩として闇雲であっても効果はあると考えます。知っている、分かっているツールでも、それを実践しないで後になって指摘を受けることは、非常にもったいなく、コロンブスのタマゴのように答えを聴くいて知ったかぶるより、その前に思いつく人材になるべきだと思います。そして、使い方がわからないときは人に聞くのも、ツールを上手く使う上で早期の成長が見込めると考えます。一人で様々な機能を試すことは難しく、ベテランの方でも長い間使用したツールについて他者から便利な機能を知ることは、若者目線から見てもよく見かけるので、聞くことはどんな立場であれ必要な心がけだと思います。

作業の効率化は、ツールだけではなく作業手順の変更や、資料のテンプレート作成、反省会など様々な要因で改善することがあります。その中でツールの使用方法を共有することは比較的簡単に実践できるだけではなく、提案者が効果を実感できているのであれば共有することで他のメンバーも同じく作業効率が改善される可能性があると考えます。ツールの新しい使い方をことあるたびに説明するのは効率的ではありませんが、雑談などのちょっとした時間に話のネタとして、会話に混ぜてみるのも面白いかもしれません。

あとがき：

たかが grep、されど grep というところでしょうか。

学びは自分で獲得するものだと、それを意識した年になったようですね。来年は、さらに多方面で学びを深めていってください。（開発部 佐藤）

07 Web サイトの保守業務に関わって学んだこと

開発部 瓦谷

開発部二年目の瓦谷です。

前回の初執筆からちょうど一年が経ちました。

OJT から携わっていたプロジェクトを離れた後は別のプロジェクトの業務に従事していましたが、そのプロジェクトからも異動し、4 月からとある Web サイトの保守案件に関わっており現在に至っています。

今回は現在関わっている Web サイトの保守案件で、私が学んだこと、生産性を向上させるために取り組んだことを書いていこうと思います。

スクラム内でのタスクの進め方

本案件では、2 週間単位のイテレーションでサイクルを回すスクラム方式となっています。2 週間の期間で Web サイトの改修、概算見積り、調査、保守を行います。

Web サイトの改修では、事前に修正内容などの調査・見積りが行われているものから改修していくのですが、中には見積り通りに改修しても上手く動かないものや、改修時のテストで見積り時には想定していなかったバグや動作が発生することがあります。

私は最初、そういった事態が起きた時は自力で解決方法を探していたのですが、タスクには予定工数があらかじめ決められているので、詰まってしまった場合にそれでは上手くいかないことを学びました。

現在はなるべくチーム内で問題の共有を行い、必要であればお客様に説明した上で問題解決をするお時間を頂戴するようにしています。

概算見積りでの工数の出し方

依頼されたタスクを受け改修を行う際は、必ず事前に調査を行い改修にどのくらい工数がかかるか、概算見積りを作成しています。

調査・概算見積りは 1、2 人日で行うのですが、タスクの中には改修範囲が膨大なものや、改修による影響範囲が広いものが存在しています。

そういったタスクの工数を算出する際、最初は自分の改修経験から時間を計算したのですが、初めて行う改修や影響範囲が広いものについての算出方法が分かりませんでした。

また自分はこれくらいの時間でできるものでも、他の人が改修することになった場合の要件理解にかかる時間も考えないといけないことを学びました。

現在では、不測の事態に備えてある程度バッファを持たせて工数を算出するように心掛けています。

タスク要件についての問合せ

改修、見積り、調査を行っている際に、お客様に問合せしないと不明な仕様やデータが存在していることが時々あります。

特に改修中にそういった仕様やデータが見つかった場合は、速やかにお客様に問合せを行わないと、タスクが予定通り終わらないことがありました。

なので私は不明点が出た時は、一旦チーム内でお客様への問合せ内容の相談を行っています。この際、何が不明であるかを明確にするように心がけています。

以上が、私が本案件に関わってから学んだ、スクラムでの生産性を高めるためのタスクの進め方となります。

今後も継続的に改善を行い、より良い仕事ができるように取り組んでいきたいと思います。

あとがき：

システム開発の仕事の中で、怠ってはいけないことが、いくつかあります。その 1 つは、コミュニケーションです。開発側の勝手判断で、間違った作業をすると、お客様の工数を無駄に消費してしまうことになるので、怒られます。引き続き、しっかり、コミュニケーションを図っていきましょう。（開発部 佐藤）

08 OJT として案件に関わる中で重要だと学んだこと

開発部 村山

大学 3 年の春、緊急事態宣言と共に講義や研究など様々な物がリモート中心になり困惑しながらも適応しながら過ごし、リモートワークにそのまま突入することとなりそろそろ慣れてきたかと思いつつも要所要所で不便さを感じています。

初めまして、開発部 1 年目の村山です。

今回、アルタス通信を書くにあたって何を書こうかと少し考えてみましたが、やはり無難に OJT として案件に関わらせていただく中で仕事での大切なものやリモートワークにとって重要だと学んだことについて書かせていただくこととしました。

開発するシステムの仕様に対する理解の重要性

学生時代にプログラミングを履修していてプログラミングをするということ自体は経験してきました。しかしながら、入社し実際に仕事に関わらせていただく際には自分で 1 から作るのではなく既に存在しているシステムに手を加えていくという観点において自分勝手なコードは当たり前ですが NG であり、仕様理解に少しでもズレが生じてしまうと作成物が意味のない物になってしまいます。また、仕様を理解していく中で自分の知識や技術となっていくのだと感じました。

そういった面でも仕様理解という物は重要であると学びました。

リモートワークにおけるコミュニケーションの重要性

リモートワークを行う中でどうしても疎かになってしまう物がコミュニケーションです。作業をする中で疑問点が出た場合に先輩方にお聞きしたいとなると slack や zoom といったコミュニケーションツールを使用すればそこまで支障は出ないと感じていましたが細かな進捗の確認や仕様に対して誤解をしてしまっている可能性がある場合等些細なことで zoom を繋いで…となるとやはり少し気が引けてしまうことがあります。しかしながら、そこできちんと確認をしないことで作成物が無駄になってしまい、時間の無駄に繋がってしまいます。現に私はコミュニケーションを怠ってしまい作業時間を無駄に多くとってしまったことが多々ありました。

また、業務上のコミュニケーションだけでなくもう少し砕けた雑談も重要ではないかと感じました。OJT の一環として毎日メンタリング面談を先輩と 20 分程度の面談を行っていましたが、これがあることでスイッチのオンオフの切り替えや業務上の疑問点の解消が出来ました。

また、何より毎日人と話す機会を設けていただくことでメンタル面での安定は保たれたと感じました。

このような点からもリモートワークにおけるコミュニケーションはより重要であると学びました。

コーディングの癖を直す

最後に実際に仕事としてプログラミングをする中で今現在も苦戦していることを書いて終わりにしたいと思います。

前述の通り学生の頃からプログラミングはしてきていますがやはりどうしても大学の課題でプログラミングをする…となると自分が読めれば良い、動けば良いという意識は持ってしまうもので自分勝手なコードになってしまうんですね。

それを 4 年間続けてしまったもので癖になってしまっている部分があり、コーディング規約を確認して、実際のコードを見ながら作業をしないと自分が記述した部分だけ違和感が…という問題が起きてしまうことが少々発生してしまいました。

コーディング規約を確認しながら癖も直しつつ様々なコードを触って自分の知識として蓄えられるようにしていきたいと

思います。

あとがき：

お客様目線で、仕様を理解することが大事です。なぜ、この機能が必要なのか、ユーザーは、この機能を使って何をしようとしているのか、プログラムや詳細仕様だけを見ていると、システムの目的を理解しないまま、仕事してしまうことがあります。深みのある仕事するには、業務理解が欠かせません。精進していきましょう。（開発部 佐藤）

09 システム開発をプロセスマイニングする

開発部　佐藤

今年の夏は、矢沢永吉 50 周年ツアーの新国立競技場に行ってきました。73 歳で、6 万人動員するって、すごいなぁって思いながら、また永ちゃんに元気をもらいました。

俺も 73 歳までプログラム作るぜ！（笑）

さて、学びと生産性ということですが、生産性を上げるために、何を学びとして改善していくのか、これまでにも、いろんな手法を試してきているので、今さら、まだ何か改善できることがあるのだろうか？

少し、システム開発の業種以外にも視野を広げて探してみたところ、プロセスマイニングというものがあることを知りました。

プロセスマイニングは、現行の情報システムからイベントログを抽出することで、実際のプロセス（開発者が思い込んでいるプロセスではなく）を発掘し、モニタリングし、改善する分析手法だそうです。オープンソースのプロセスマイニングツールもあります。

であるならば、システム開発の仕事そのものをプロセスマイニングできないものかと考えて試してみることにしました。

ただし、私たちの仕事は、専用の業務システムがあるわけではないので、イベントログが取れません。取れないなら、作りに行こうということで、手始めに月間の作業時間表を、少し細かく明細を書くように改良して、それを蓄積して、分析してみようと、社員の人たちに協力してもらっています。今、スタートして 2 カ月くらい経ったところです。今のところ、まだ見えてきたものはありませんが、まだまだ始めたばかりなので、1 年後くらいに期待しています。

作業明細は、書くのがメンドクサイだろうことも想像に易く、いつか、書かなくなったり、コピペで済ませてしまったりと、データの質が懸念されるので、”明細書き”を簡単にする仕組みを考えないとダメだろうと思っています。仕組み化することで、”思い込み”の作業明細ではなく、客観的で正確な情報が記録できると、分析精度も高くなるでしょう。

一部の社員からは、rewind というツールがあることも教えてもらいました。Mac でしか動かないですが、仕事中の画面を録画し、機械学習に食わせて文章化して分析するツールのようです。これが使えるレベルに仕上がっているなら選択肢の 1 つです。Windows 版が出ることを期待します。

私たちの仕事をプロセスマイニングできたら、WEB 制作会社の仕事やプロダクトデザインの会社などクリエイティブな職種にも展開できそうです。

作業明細の記録が活用できるようになったら、次にやりたいのが、進捗管理との連動です。スケジュールが計画通りに進まなくなると、進捗が見えなくなるので、スケジュールを引き直すわけですが、これっ、結構時間がかかるので、あまり頻繁にはできません。でも、理想的には、実績を取り込んで、毎日、スケジュールを引き直したいです。

MS Project には、スケジュールの自動割り当て機能があるようだけど、普段、Excel でスケジュール管理しているので、MS Project を使いこなすのが面倒で、気が進みません。

そんなとき、私は、変態なので、ついついサンデープログラミングで、スケジュールの自動割り当てツールを作り始めてしまうのでした。タスクと見積工数を登録すると、自動的に担当者をタスク割り当てするツールなのですが、担当者をラウンドロビンで割り当てる機能は、そんなに時間かけずに作れました。でも、これじゃ使い物になりません。このタスクは○○な人しかできないとか、1 日の稼働時間が人によって違うとか、土日祝以外の休みも管理しないといけないし、人によって消化工数の%を変えたいとか、使えるレベルにするのは、まだまだ機能が必要です。

そして、考えました、開発テーマとしても、かなり面白そうなので、これを、オープンソースにして開発しつつ、同時に、社内研修の題材にしようと。

社内研修で使うなら、このプログラムを DDD のアーキテクチャーで作成して、DDD を学ぼうと準備を始めています。

スケジューリングの課題は、行きつくところは、ナーススケジューリング問題と言われているので、そこまでやり切れると、オープンソースとして価値が出るかもしれません。

システムの開発をプロセスマイニングして、進捗管理に連携

し、スケジューリング処理をオープンソース化して、社内研修にも活用する。どこまで実現できるかはわかりませんが、その過程で学べる事は、たくさんありそうです。

10 お菓子紹介

管理部　間宮

今期も日頃の感謝の気持ちを込めまして、アルタスファイブから選りすぐりのお菓子をご紹介させていただきます。

リモート勤務が続く中、なかなか人数が揃う日って少ないものですね。みんなで食べ比べる機会がないので、グランプリを決められない日々が続いています…。

つきましては今回、

「アルタスファイブ　おやつグランプリ」ではなく

「アルタスファイブ　おやつセレクション」！！！

としまして、代表佐藤が厳選した 2 品をご紹介いたします。

※「アルタスファイブ　おやつグランプリ」とは、普段の生活の中にある「アレ美味しかったよ」を持ち寄って、会社で購入して、月一の帰社日に、パートナーさん含めて、みんなで食べてみて、レビューを付けるという、ただそれだけのイベントです。

オススメするのはこちらです～

●廣尾瓢月堂　【B.T. B. SABLE】

B.T.B.とは、BLACK TRUFFLE BUTTER。そうなのです、こちらは「黒トリュフバターの薫りを楽しむサブレ」なんです！

サブレというとあまーいイメージがありますが、これは一味違います。生地には小麦粉、国産バター、砂糖が使用されほんのり甘いですが、ここにトリュフオイル、ブイヨンを加えてカリカリに焼き上げています。さらに！表面には黒トリュフをブレンドしたオリジナルシーズニングパウダーが。もう黒トリュフまみれです。ガーリックの風味もあり、お菓子というよりおつまみ。お酒にとても合います。

●VIVANT｜福井市洋菓子店 びばぁーん　【VIVANTBAR～ショコラバー～】

続いて紹介するのは、ご当地バウムクーヘンを決める祭典「ファイナルクーヘン総選挙 2021 秋の陣」で全国 3 位に選ばれたこちら！

何とも珍しいアイスキャンディー型のバウムクーヘンです。土台はプレーン・ストロベリー・チョコの 3 種類。そこへホワイトチョコや抹茶チョコなど、様々な風味のチョコレートがコーティングされていて、とても洒落てます。生地には福井県産のコシヒカリを使用しており、もちもちしっとり。見た目だけでなく
ちゃんと美味しい逸品です。

以上、今回はセレクションとさせていただきましたが、

また「グランプリ」を決める、そんな日がいつか訪れますように…。

来たるべきその時のために、お菓子を探し続ける旅はまだまだ続きます…！

あとがき

私が会社に出勤するのは、月に 1 回か 2 回くらいなんですが、出勤した日の帰りには、伊勢丹に寄って、デパ地下巡りしてます。お菓子バイヤーです。(笑)（開発部 佐藤）

次号につづく