アルタスファイブ通信　2023 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！」より皆様へ、

心からの感謝を込めて、2023 年冬号をお届けします。今号では、「進化」「変化」「転換」というテーマに沿って、私たちの日々の活動を振り返りました。

技術の向上心と課題意識を常に持ち、自走する組織を築くことが、優れた成果につながっていくのだと思います。私たちの経験と専門知識が日々の業務を通じて深まり、皆様とのプロジェクトでより高い価値を生み出せるように、引き続き努めて参ります。

本号では、過去半年間のプロジェクト事例とそこから得られた教訓に着目しました。また、私たちの専門分野での技術の進化と、日常業務での小さな成功や発見についても紹介しています。

私たちアルタスファイブは、皆様との関係を大切にし、それぞれの業務に全力を尽くすことで市場への貢献を目指しています。この「アルタスファイブ通信」が、皆様との絆を強化し、共に成長するための一助となれば幸いです。

目次

01 職場の未来を創る：人事評価制度導入と社内研修で実現する成長と進化

管理部：大成

2023 年、Altus-Five は新たな 2 つの取り組みを行いました。一つ目は「人事評価制度の導入」、もう一つは Altus 史上初となる全社員参加の「社内研修」の実施です。

【人事評価制度：進化する評価の枠組み】

新しい人事評価制度は、自己成長や他者支援を行う社員に対し、報酬に反映した形で評価することを目的として作りました。すでに社内制度として自己研鑽への費用負担などはしていましたが、人事評価とも連動することで、より社員が成長しやすい環境を整えました。

人事評価制度には、等級に応じたスキルマップを用意しました。スキルマップは、社員が成長するべき方向を示す羅針盤です。時代によって要求されるスキルや技術は変化しますので、スキルマップは日々日々進化させる必要があります。まだまだ荒削りなところがありますが、社員の意見を反映しつつ、持続的にアップデートしています。

また、以前、社員に Altus-Five がどうみえるのかをヒアリングしたところ、そこでも学びの環境があることに対してポジティブなフィードバックがありました。他にも、嬉しい声があったので、以下にその一部を紹介します。

・「開発環境の自由度が高い」
・「プロジェクトに縛られない多様なチャンス」
・「広々とした作業スペースとアクセスの良さ」
・「定時退社が普通」
・「有給が取りやすい」
・「自己研鑽へのサポート体制が手厚い」
・「書籍購入支援がいい」
・「技術的な議論が上下関係なくできる」

【社内研修：コラボレーションと学びの融合】

「ドメイン駆動開発」をテーマにした社内研修では、今年 4 月の新入社員から勤続数十年のベテラン社員まで全社員が集まりました。午前中は ABD(Active Book Dialogue)という読書法で座学をし、午後はハッカソン形式で勤怠管理システムの開発に取り組みました。

ハッカソンでは、4 つのグループにわかれて行い、最も素晴らしいコードを書いたチームには、表彰状と景品をプレゼントしました。最優秀プログラミング賞を受賞したのは、澤邉チームでした。澤邉さん、西岡さん、小林さん、柳さん、太田さん、おめでとうございます！

研修後のアンケートからは、チームワークの楽しさや、対面でのコミュニケーションの新鮮さが伝わってきます。一部紹介します。

・「普段プロジェクトが一緒ではない方々とも協力して作業ができたのが楽しかった」
・「チーム内で話し合うことで、自分の中の理解度が増したのがよかった」
・「討論しながら進められたりと普段ではできない経験ができたことが楽しかった」
・「コロナ以降入社なので仕事で面と向かって協力したのが初めてで楽しかった」
・「普段話さないメンバーや久々に会う同期と話しながらチーム制作できるのは面白かった」
・「リモートワークにはない楽しさを実感できました」
・「リモートワークで人と直接話すことが少ないのでとても新鮮に感じました」
・「久々に対面で、一つの課題について話し合えたことが楽しかったです」

記念写真をみると、みんなの笑顔や活気のある表情が、社内研修が成功裏に終わったことを物語っています。

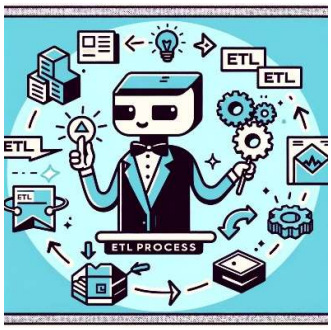
【今後の展望と期待】

「開発しかできないプロ集団」がモットーの Altus-Five。変態と自称する佐藤社長のもと、私たちは技術のイノベーションの道をさらに突き進んでまいります。これからも Altus-Five の成長と進化に、どうぞご期待ください！！！

編集からの一言：

スキルマップの作成は、私も加わって、かなり明細化しました。このスキルマップを埋めていくためには、いままで頭ではわかっていたけど、実践につなげられていなかったことを、具体的に実践していくことになるので、時間はかかりますが、確実に何かが変わってくると思います。何年後かが楽しみです。（開発部 佐藤）

02 現プロジェクトの ETL について



開発部 鈴木

涼しくなってきて夏が終わったと思ったら急激に冷え込んですぐ冬になってしまいました。バイク乗りの私としては一番いい季節の秋が短くて悲しいところです。Altus 通信の執筆は初めてになります、開発部の 2 年目の鈴木です。

今回のテーマは「変化、進化、転換」ということです。

そこで、現在のプロジェクトで携わらせていただいている、ETL の紹介とそれに付随した内容を書かせていただこうと思います。

ETL とは、Extract(抽出)Transform(変換)Load(書き出し)の略語であり、さまざまなデータベースやシステムからデータを抽出し、扱いやすいフォーマットに変換して、DWH(データウェアハウス)に書き出す一連のプロセスのことを指します。

現在のプロジェクトには既存で ETL があり、そちらは「各地にある機械」→「Stream に貯める」→「Stream から抽出」→「変換」→「DWH に書き出し」となっております。こちらの最終保存先である DWH を、Amazon AWS 上にしたものを分岐させてもう 1 つ作成するというプロジェクトに携わらせていただいております。

現在のプロジェクトは AWS 上に保存をするという目的から始まっており、AWS の保存先となるサービスを選ぶところから始まりました。保存先のサービスとして AWS Kinesis(ストリーム)と AWS S3 が候補に挙がりました。それぞれメリット・デメリットがありそれぞれのプロトタイプを作成し比較、検討しました。データの取得元がストリーミングでしたので、Kinesis にすれば Transform(変換)無しで Extract(抽出)と Load(書き出し)のみにでき、システムとしてはシンプルで良かったのですが、それぞれのストリームで 1 データ当たりの上限サイズが違い、AWS Kinesis の方が小さいことから断念し、AWS S3 へ決定しました。

そして S3 に決定したことで、「Stream に貯める」部分までは既存のものを活用し、「Stream から抽出」→「変換」→「AWS S3 に書き出し」システムを作成することになりました。

こちらのシステムを実装するに当たって、Stream とは何かを全く知らかった私は、まず Stream について学ぶ必要がありました。Stream とは何か、どのような場面で使うのかといった前提知識から Stream の使い方や使う際の注意点といった技術知識についてしらべ、Stream の利便性やどうして現在では盛んに使用されているかについて、連続データなのが便利であるが、故に取り出す際に順番性を保証することが大事なことなどを学べました。Stream は近年多くの場面で使用されているとのことなので、今後にも生かせる知識を学べたと思っています。

Stream から取得したものを AWS S3 に書き出すことになったため変換を実装する必要になりました。こちらの変換は、Stream に格納されている Json のデータを元に、実際のファイルに変換する処理になります。そして、変換したファイルを AWS S3 へ書き出しを行うように実装しました。これらの一連の ETL を今回は Python を使用して実装しました。私は学生の頃に機械学習の実装で Python を使用したことがありました。Python は今回のようなデータ解析や機械学習といったデータの扱いをするのに優れており、ETL を実装するのには向いた言語だと思います。少し今回の本題とはズレますが、最近は Python で実装されているものを多く見かけるようになり、これも変化だなと感じております。

今回は私が現在携わっているプロジェクトの ETL の紹介とそれに付随した内容について書かせていただきました。今後も ETL やそれに付随した Stream などの知識を深めていきより良い実装ができるよう努めて参ります。

編集からの一言：

ETL 自体が、変換の仕組みですからね。今回のテーマとしても、とんち的に、うまくハマってると思います。

それにしても、最近、ETL や DWH に関する案件が増えてきています。この号だけでも 3 つの記事で取り上げられています。小規模データ分析基盤や各種 ETL は、私たちの得意分野の一つとなったようです。（開発部 佐藤）

03 ChatGPT が開発で活用され始めて感じたことについて

開発部 柳

健康診断で肥満と診断されてから約 6 カ月が経ち、生活習慣に筋トレとウォーキングを組み込んだおかげで 8 キロも痩せることができました。開発部の柳です。

今年から新たに案件に参画させていただき、チームの構成やタスクの進め方など、いろいろなことが一新されました。特に印象的な変化として、ニュース番組でも取り上げられていた『ChatGPT』を、本格的に開発に利用しようということがありました。今回は ChatGPT を取り入れた開発で感じたことや変わったことについて話させていただきます。

ChatGPT は対話型 AI チャットサービスで、特徴として「会話の文脈」、「話の流れ」、「質問の意図」まで読み取って文章生成をし、プログラムのサンプルコードやライブラリの使用方法を教えてくれるなど様々なことに利用できる便利なツールです。私が参画している案件では GCP(Google Cloud Platform)の Dataflow というデータの変換と受け渡しを行うストリーミンサービスを使用しており、ApacheBeam というプログラミングモデルでコーディングする必要があったため、ChatGPT にメソッドの使用方法や Dataflow のドキュメントに記載されているサンプルの解説など学習面で大いに活躍しました。ChatGPT を活用した開発は、ふと忘れてしまったコードなどを気軽に質問できるため、効率面で自分で調べるよりも早く、今では手放せない存在になりつつあるような気がします。

開発で利用している ChatGPT では、2019 年までの情報で学習しているため、サンプルコードを作成した際に情報が古く、そのままでは動かないことが多々あります。サンプルを作成しなければいいのではとも考えましたが、ApacheBeam にはサンプルが余りにもなかったのでそれは断念しました。

どうにかして活用できないか考えた結果、私は ChatGPT を「何かを実装するきっかけを作る手段」として利用しました。やり方は単純で、ChatGPT にサンプルを作成してもらい、サンプルコードのライブラリなどを Web で調べなおすだけです。単純なやり方ですが、具体的に実装したい内容を日本語の文章として書けるため欲しかった回答が 1 分程度で知ることができ、調べなおすことで理解が深まるため、Web で闇雲に探すよりは効率的に実装を進めることができました。

ChatGPT の質問した内容が直ぐに返ってきてそれが開発で役に立つというのは、私の中では驚きでこれからさらに発展していくとなると、知らないと技術的に遅れるといった話は嘘ではないと実感させられます。案件で使用した ChatGPT は社内で共有されているもので、特に自分で作ったものではありませんでしたが、既に虜になっているのでそのうち自分用のものを作ってみたいと思います。

編集からの一言：

ChatGPT は、開発者をダメにする危険な性質があります。

例えば、ノープランで仕事を始めて、何から何まで ChatGPT に質問する底なし沼にハマってしまうケース。自分で考えて問題解決した方が早いということも多々あるのではないか。

それから、ChatGPT に教えてもらったことは、自分自身にはあまり学習できていない可能性。同じような質問を何度も、ChatGPT に聞いてしまう状態は、学習してない証拠です。

使い方は気を付けないといけないと、最近思います。（開発部 佐藤）

04 成長の実感を得られるようになるまで

開発部 小林

開発部 3 年目の小林です。

入社してからあっという間に 3 年が過ぎました。これまでに java・php・python などのプログラミング言語や、elasticsearch・AWS などの技術に触れるプロジェクトに参加し、開発や業務などの幅広い知識と経験を身に付けてきました。今年は、近頃話題になっている chatGPT を使用したプロジェクトに携わることができ、最新技術を使ったシステムを作るという貴重な経験をすることができました。様々な業務を通して、多方面で成長できたと感じています。

現在は自身の成長を自覚できている私ですが、新入社員になってからしばらくの間は、成長を感じられない時期がありました。仕事を通じて知識や経験は増えているはずなのに、その実感が伴わず、周囲の人と比べて自分は成長できていないのではないかと考えてしまうことが少なくありませんでした。プロジェクト内で振り返り等を行っても、出来なかったことばかりが目に付いて、自分を責めたり、気持ちが沈んで

しまうことが多々ありました。

そんなある日、上司からの指示で受けた研修で、「社会人になったら、自分で自分を褒められるようになりなさい」「小さな成長に気づけるようになりなさい」という事を学びました。このことがきっかけで、自分がこれまでにやってきた仕事を振り返り、内省する時間を作るようになりました。出来なかったことなどのネガティブな事を振り返るのではなく、自分が仕事の中でどういう学びがあったのか、今日の自分は何か新しいことを知ることができたのか、等といったポジティブな事を振り返るようにしてみました。また、他者と比較するのではなく、過去の自分と比較するようにもしてみました。そうすると、成長していないと思っていた自分にも、新しく学んだ事やできるようになった事があることに気づきました。

今年、私は 3 年目社員として、約 1 年にわたって新入社員のメンターをする機会をいただいています。彼らと話をしていると、かつての私と同様に、自分は成長できていないのではないかという相談をされることが何度かあります。私から見ると、彼らは日々の仕事を通してしっかりと成長できているように見えるのですが、彼らはそのことを自覚できていなかったり、自覚できていても「大したことではない」「他の人の方が成長している」と感じているようでした。

成長実感を感じるということは、仕事を楽しく続けていくために必要不可欠なことだと思います。成長というものには、目に見えるような大きな変化だけではなく、日々の地道な積み重ねも含まれます。しかし、そういった小さな成長や変化はなかなか気づきにくいものです。かつて成長実感を得られず辛いと感じた経験がある私だからこそ、メンターとして新入社員に対して、日々の変化や成長に気づくための手助けや、成長実感を得るための工夫やアドバイスができるのではないかと思っています。その人なりの小さな気付きが得られるように、そして成長実感が得られるように、出来る限りの支援をしていきたいと思っています。

まだまだメンターとして未熟なところはありますが、新入社員との日々のやりとりなどを通じて、誰にとっても働きやすい職場を作っていきたいと考えています。そして、私自身も切磋琢磨し合うメンバーの一人として、業務などを通して日々成長し続けていきたいと思います。

編集からの一言：

コメントは不要ですね。ジーンときました。（開発部 佐藤）

05 時代の変化を感知するための情報収集

開発部 神崎

今年一年、気温がジェットコースターのように暑くかったり、寒くなったりとエアコンの温度設定や服装選びに困る一年でした。ChatGPT があっという間に AI のトレンドになり、ビジネスの中に組み込まれはじめ、いよいよ Deep Learning の技術が一般化されたと思います。大学で少しだけ学んでいたので、一般的に広まって、嬉しく思います。

皆さんはこの一年、どのような一年でしたでしょうか？来年も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「変化、進化、転換」ということです。

変化というと IT 業界のトレンドの変化はエンジニアとして感知しておきたいものです。企業によってはチャットツールで IT のニュースサイトや知識共有コミュニティサイトなどの記事を BOT が投稿するチャンネルがあったりするほどなので、情報収集は重要度が高いでしょう。個人のスキルを高めるという点でも情報収集は重要だと思います。

今回は、個人的に情報収集に利用しているいくつかのサイトを紹介したいと思います。

まず 1 つ目は、GIGAZINE(https://gigazine.net/)です。ビジネスとして有用かというよりは、ネットサービスやソフトウェアの情報が簡単な解説記事とともに読めるので、手始めに読むのに最適だと思っています。読んで興味が湧いたものは後で調べたりします。IT 情報以外の科学分野の記事も多いのでよく見ています。

2 つ目、最近見つけました TechFeed(https://techfeed.io/)と呼ばれるサービスです。こちらは、他サイトの記事を収集して分野ごとに載せているサービスのようです。RSS リーダーで情報収集するような形に近いと思います。情報量が多いのでトレンドをぱっと知るには有効なサービスです。情報の正確度については記事によるので最新技術やトレンドを集めるのに使うのが有効かと思います。海外記事も収集しているので海外でのトレンドもある程度知ることができると思います。

3 つ目は、Amazon Kindle Unlimited です。IT 関連の本は価格が高い本が多いので、月額 980 円(2023 年時点)で最新の本

も読めるのが、個人的に重宝しています。本は知識が体系化されているので、トレンドを後から追うのに良さそうです。あと電子書籍は、かさばらないメリットが一番大きいですね。どうしても紙で読みたいもののみ、紙で買うようになりました。電子書籍のサービスで他に良いがあるかもしれないのですが、このあたりはあまり調べていないので皆さんの中でもっと良いサービスがあるかもしれませんね。もう少し電子書籍のサービスを調査してみたいと思います。

４つ目は ChatGPT ですね。皆さんも利用しているのではないかと思います。特にこだわった使い方はしていないですが、回答内容が怪しい時は「本当に？」などで聞くと間違っていましたと訂正を行ってくれるやより詳しい解説をするので度々使います。個人的には変数名やメソッド名を考えるのが苦手なので、ChatGPT に候補をいくつか出してもらうのが重宝しています。最近は、Github Copilot で簡易コードレビューをしてもらえるらしいですね。

話は変わりますが、今後は、できるだけプログラミングをせずに開発するノーコード・ローコードが進んでいくといわれているようです。今後はプログラミング力だけでなく設計工程の能力も重要になってくるそうです。私も自社内で行われた DDD（ドメイン駆動設計）の社内研修に参加しました。なかなか難しいですが、有意義でした。今後も本や研修で設計工程について学んでいきたいと思います。

さて、皆さんはどのように情報収集しているでしょうか？社内で共有してみるのも良いかもしれません。

最後に、トレンドに振り回されないように、基礎を疎かにせず、情報収集できるよう気を付けていきたいですね。

編集からの一言：

最近、ノーコード・ローコードの製品いろいろ出てきたなぁ・・・と思っていたら、生成 AI が出てきました。ローコードツールも、生成 AI が置き換えちゃいそうだなぁ・・・。

実は、何年か前に、私も、体に鞭打って空き時間を使ってローコードツールの開発に勤しんだことがありました。そのツールは、SAStruts 向けのコードを生成するツールだったのですが、ある程度、自動生成できるところまで開発できたとき、なんと Seasar2 が EOL で終了するニュースが出たんです。空き時間で開発したので 1 年くらいかかったのに、可哀そうな俺。その後、Spring 向けに改良することも考えましたが、完全に燃え尽きました。（開発部 佐藤）

06 ストリーミング処理の変化と Apache Beam

開発部 檜山

現在参画させていただいている案件で、Apache Beam を使ってストリーミング処理を実装する機会がありました。

今回のテーマは「変化・進化・転換」なので、ストリーミング処理の変化と Apache Beam の登場によって何が変わったのかについて、簡単にお話をさせていただきたいと思います。

※Apache Beam：分散処理のためのオープンソースで、バッチやストリーミング処理の構築に利用されています。

そもそもストリーミング処理とは？

本題に入る前にそもそもストリーミング処理とは何か？を明確にしておきたいと思います。
ストリーミング処理とは、無限に流れてくる(可能性がある)データをリアルタイムに処理していくことです。
例えば、金融取引の為替のように半永久的に流れてくるデータをリアルタイムに処理するような感じです。

[1]の記事を参考にすると下記 3 つの要素を含んでいるのがストリーミング処理と言えます。

無限のデータを処理する
処理が永続的に続く
低遅延/リアルタイム性が求められる

対して、有限のデータを扱う処理はバッチ処理と言われています。

ストリーミング処理の変化

- ラムダアーキテクチャ：バッチとストリーミング処理の合わせ技
- カッパアーキテクチャ：ストリーミング処理 Only
- Apache Beam の登場：バッチとストリーミング両方構築可能

ラムダアーキテクチャ

ラムダアーキテクチャは、バッチ処理とストリーミング処理を組み合わせることで正確性とリアルタイム性の両方をある程度確保しようとするアーキテクチャです。

ストリーミング処理で最新データの近似値をリアルタイムに提供し、後でバッチ処理で正確な結果を提供するというものです。

しかし、下記のような課題がありました。

- リアルタイムに出力されるデータはあくまでも近似値であり正確な値でない可能性があること
- バッチ処理とストリーミング処理を異なるプログラミングモデルで構築する必要があること

後者については、別々にコーディングしたプログラムで、ロジックレベルの整合性を取る必要があるので、保守が非常に大変になります。

カッパアーキテクチャ

カッパアーキテクチャは、すべての処理をストリーミング処理だけで実現したアーキテクチャです。

そのメカニズムの詳細は理解できていないのですが、[16]の記事の言葉を借りると、「そもそもストリーム処理だけで全て計算できるのではないか」という考えで編み出されたようです。(おそらく、ハードウェアやソフトウェアの進歩により、リアルタイムでデータを捌けるようになり、だったら、ストリーミング処理だけでいけるのでは？となったものと思われます。)

Apache Beam の登場

カッパアーキテクチャにより、リアルタイムに最新データを処理できるようになりましたが、あくまでもストリーミング処理に主眼を置いたものでした。

それに対して、Apache Beam は、バッチ処理とストリーミング処理の両方を同じプログラミングモデルで構築することができます。

- ストリーミング処理でもある程度バッチ処理で検証できる
- バッチ⇔ストリーミングへの変換が比較的容易
- UT もやりやすいとコストが全く異なるということもありました。

Apache Beam は、公式のプログラミングガイドやサンプルコードはありますが、それ以外だと、ググってもあまり記事がなく(特に日本語の記事は少ない)、最初のころは動くコードを書くだけでもかなり苦戦しました。

そのため、コードを書いて、実行して、エラーや不具合を修正して再度実行して・・・とトライ & エラーで学習していきましたが、ストリーミング処理で検証する場合、毎回データを流す or データを流し続けておく必要があるのに対し、バッチ処理であれば、用意した JSON ファイルや Text ファイルをそのまま読み込ませることができ、すぐに実行・検証することができます。

また、ある程度バッチ処理で検証して、コーディングや I/O の使い方を学習してしまえば、ストリーミング処理への変更は比較的容易で、入力データソースを変えたり(無限→有限)、オプションを変えたりなどは必要ですが、大きくコードを変えずに変更することができました。

またちょっと処理を変えて動かしてみたい場合もすぐに検証できるので楽でした。(もちろん、 I /O によってはバッチしかサポートしていないものもあるので、対応されているかは要確認ですが)

こういったことができるのもバッチとストリーミングで同じプログラミングモデルで実現されているおかげなので、非常に助かりました。(開発当初はその有難みを感じられていませんでしたが笑)

まとめ

Apache Beam は、バッチ処理とストリーミング処理を同じプログラミングモデルで書くことができ、両方の処理を構築できるという点の良さを再認識することができました。

ただし、ほぼ同じ処理でも、ストリーミング処理だと$200 近くかかるものが、バッチ処理だと$30 程度まで抑えられたりもしたので、リアルタイム性を求められないものであれば、バッチ処理の検討をするなど、どちらにするかは機能ごとに検討が必要になりそうです。

また、実行には Google Plat Form の Dataflow を使いましたが、処理は同じでも利用する Runner や、実行オプションによってリソース消費が大きく変わり、処理によっては正常に動かなくなったり、料金体系が異なり費用が大きく変わるといったこともありました。
一部ではありますが、一般提供されている機能でも表示上の不具合があったりなど Dataflow に関してはまだまだ改善してほしい点があると感じています。

参考記事

[1] 【要約】The world beyond batch: Streaming 101
[2] 【要約】The world beyond batch: Streaming 102
[3] ストリーミング処理とは何か？+2016 年の出来事
[4] Dataflow が解決するストリーミング処理の課題と基盤構築で考慮すること
[5] Big Data + Fast Data = ラムダアーキテクチャー！
[6] GCP ブログ記事 Dataflow の仕組み: 誕生秘話
[7] Google Cloud Dataflow ってなんだ？
[8] なぜ Apache Beam なのか : Dataflow のライバル参入を促す理由
[9] Dataflow が解決するストリーミング処理の課題と基盤構築で考慮すること
[10] Dataflow Prime のストリーミングのジョブにおける垂直自動スケーリングの導入
[11] 使用すべきデータ パイプライン アーキテクチャとは
[12] 新しい高速アーキテクチャにより、多言語 Dataflow パイプラインが利用可能に
[13] Python を使用して Dataflow パイプラインを作成する

[14] パフォーマンス、コスト、キャパシティ プランニングのための Dataflow ジョブのベンチマーク
[15] Dataflow Runner V2 を使用する
[16] ストリーム処理を 1 から Kappa Architectre まで学ぶ

編集からの一言：

私も以前、Apache Beam で開発したことがあります。並列分散処理ということを強く意識して、集約と再分散が起こらないようにパイプラインを作ることで、性能を絞り出さないといけないですよね。最初の頃、そのあたりが、ピンと来てなくて、性能の悪い処理を書いていた気がします。ビッグデータは、パーティショニング、インデックス設計、データフレームの冗長設計、それと集約再分散の最適化、このあたりが勘所でしょうか。（開発部 佐藤）

07 コードを書かない時代のシステム屋

開発部 佐藤

今号から、記事１つ１つに挿絵を付けてみました。実は、この挿絵は、記事の内容をプロンプトで与えて、内容にあったイメージを ChatGPT に作ってもらったものです。

いや、これ楽しくて、丸１日、遊んでしまいました。「藤子不二雄風に」とか、「梶原一騎風に」とか、それっぽくイラストが出力されて、とにかく面白いんです。そして「手塚治虫風に」とリクエストしたら「申し訳ありませんが、お客様のリクエストはコンテンツポリシーに違反して・・・」とエラーになったので、何を判断基準にしているか不明だが著作権対策もなされているようです。次に「ドラえもんの作画タッチでお願い」とリクエストしたら、それ ドラえもん だから！っていう絵が出力されちゃった。まだまだ完全ではないようです。この未完成感は、ホッとします。

私が最近気になっていることは、「GitHub、Copilot の将来像「Copilot Workspace」発表　人間がコードを書くことなく、Copilot が仕様作成からコード作成　デバッグまで実行」という記事です。来年にリリース予定となっているので、そんなに早くにコードを書かない時代が来てしまうのかと、ちょっと寒気がしました。ドラえもんの絵と同じように、未完成感が残っていて欲しいです。まだしばらくは。

アジャイル開発が普及して、ドキュメント量が減ってコード中心の開発にシフトしてきましたが、Copilot Workspace では、仕様書を AI が作成し、人間が加筆修正して、自然言語でシステムを作っていくことになります。人間にとっても、コードを読むより仕様書を読んだ方が、システムを理解できるので、理想的ではあるけれど、記事で紹介されているデモを見た感じでは、Copilot Workspace が作成する仕様は、Issue チケットの指示内容に限定した仕様作成にとどまっていて、システム全体を 1 冊の仕様書として維持管理されたものではないように見えます。これだと、完全自動化への道は、まだ、先が長そうです。今でも Open Interpreter とか、AutoGPT などがありますが、その便利さ具合と、そんなに変わらないかもしれないです。

でも、そのうち、本当にコードを書かなくなる時代が来るのでしょう。

コードを書かない時代のシステム屋の仕事はどうなっていくのでしょうか。AI との意思疎通が自然言語で取り交わされるので、文章力は必要になりそうです。自然言語には曖昧さがあり、勘違いを生みやすいので、文章を読み解く力、書く力の両方が求められるでしょう。そして、AI を使いこなすためのテクニックの習得も必要です。ハルシネーションというのは、AI の代表的な問題ですが、それを回避するために AI の癖を知り、正しい答えを出しやすくするテクニック、これが必要です。すでに、さまざまなプロンプトのテクニックが紹介されています。中にはオマジナイのようなものもあります。
コードを書かなくなる代わりに、AI を使いこなすプロンプト術を織り交ぜながら文章を書く、これが AI 時代の仕事、そんなところでしょうか。
文章は普段から書き慣れていないと、すごく億劫で筆が進まないので、エンジニアは、ブログなどを書いて慣れておくとよいかもしれません。伝えたいことが言葉にできないことがよくあるので、言語化する訓練にもよいと思います。

他方、コードを書かない時代のシステムの開発生産性は、確実に向上します。1 年かけて開発していたものが、半年で開発されるようになるかもしれません。そうなると、システムの開発案件は、市場から半減してしまうのだろうか？
たぶん、その心配は不要で、システムを導入する企業からすると、1 年で 2 つのプロダクトを開発できるようになるので、システムの開発競争は、さらに激化していくのではないかと想像します。

ただし、同時に内製化も進みそうなので、私たちシステム屋も、SI だけではなく、プロダクト開発を始めるときが来ているのかもしれません。AI を活用した超ニッチなプロダクトを開発してみたいです。

08 お菓子紹介

管理部　宇野

今期も、日頃の感謝の気持ちを込めまして、社内で食べ比べしたお菓子をご紹介させていただきます。

コロナ禍になってから、在宅勤務が主流となったので社内でのお菓子試食会の頻度が減りました。ですが、今期は社内研修が開催され社員が全員集合する場がありました。皆にお菓子を配れるぞ！とうきうきお菓子探しをすることができ、社員全員で食べ比べをして、高評価だったお菓子をご紹介することができます。

今期は、比較的ネットや百貨店などで買い求めやすいものなので、是非探してみてください。

今期おすすめのお菓子は 2 種類でございます。

福島銘菓　いもくり佐太郎

こちらは福島の限定スイーツです。その名の通り、いも（サツマイモ） × 栗 のコラボレーション。間違いのない組み合わせのお菓子です。こちらは地元のイオンでも売り切れることがあるぐらい愛されている商品らしいです。浜松でいううなぎパイみたいなものでしょうか？（宇野は静岡出身です）お菓子の大会で賞を受賞したこともあるこのお菓子は、表面を”軽く”カリっと焼き上げる職人による繊細な焼き加減が特徴です。全体的な味はサツマイモ感が強いのですが、中身のしっとり食感の中にアクセントで主張してくる栗が飽きのこない味と食感を演出しています。サイズも大きすぎず小さすぎず、小腹が空いたおやつにぴったりです。

ラメゾン白金　ショコラサンド

サクッと食感、なめらかなくちどけ、リッチで”濃厚な”ショコラサンドです。ベルギー産のクーベルチュールを贅沢に使用した分厚いチョコレートがサンドされています。王道愛されチョコスイーツです。サンドされているチョコレートは板チョコのような見た目と裏腹に、サクッと食感のクッキーに負けないぐらいの密度・程よいかたさ、しかしクッキーを噛んだ勢いを邪魔せず歯が入るような程よい柔らかさがあり、チョコレート好きにはたまらない長方形のトリュフのように感じました。もう一つ食べたらもっとうまいことが言えそうな気がします。今度はキャラメルクッキーのほうを味見すべく、伊勢丹にお買ものに行きたいところですね…

毎年なんだかんだありますが、今年の 12 月はおかしな気温が続きますね。朝は 10℃、昼は 20℃。ダウンで良いのか暑いのか。よくわからない気温差に体調が負けそうな日々です。

冬のお菓子選定は、気温に左右されるものを選びがちなので寒暖差少なく 12 月らしい気温に落ち着いていただきたいという気持ちになります。

是非、おすすめのお菓子があればいつでも情報募集しております。

紹介したお菓子も是非食べてみてください。

編集からの一言：

栗が食べたくなりますね。ここで紹介したもの以外にも、新宿伊勢丹で平日限定で販売されている「超和栗」というフワフワなどら焼きを食べましたが、それも、おいしかったです。この時期、栗菓子を制覇したくなります。（開発部 佐藤）

次号につづく