=== Chapter-o はじめに ===

本書について

提供形態

今見ているウェブサイトの形で、無料で公開しています。

紙の書籍や電子書籍、PDFファイル、その他原稿データの公開等はありません。

想定読者

以下を対象とします。

> PC（パソコン）を使って活動していて、特に「タスク管理」という言葉を知っており、かつやり方や考え方やツールなどを自分なりでまがいなりでもいいので取り入れている人

デスクワークに限らず学業や日常生活やその他趣味による活動も想定していますし、年齢や職種や業種も問いませんが、上記に当てはまらない人には「未知の分野」となりハードルが高いかもしれません。

普段はパソコンを使わずスマホや手帳を使う人も問題ありませんが、一部コンテンツは（PC を前提としているため）当てはまらないことがあります。

本書のテイストと章構成

淡々と解説します。なるべく平易な表現・例・たとえを、また必要に応じて箇条書きによる列挙も使って、わかりやすく仕上げています。内容としては権威的に保証されたコンテンツを噛み砕くというよりも、まだ体系化されていない荒波を筆者なりにまとめるものとなります。またビジネス書や自己啓発書に見られるような「ストーリー」は書きません。

構成としては、「部」として章をグルーピングしています。各部は次のようになっています。

- 第 1 部では、タスク管理の全体像を眺めていきます
- 第 2 部では、本書が主に扱う「個人の」タスク管理の基礎として、様々な戦略やスタンス

を整理します
- 第 3 部では、タスクのようでタスクではないものを個別に詳しく見ていきます
- 第 4 部では、応用編としてタスク管理に関する周辺概念（例: 習慣、割り込み、振り返り）を詳しく見ていきます
- 第 5 部では、高度編としてテキストベースのタスク管理のあり方を提案します

第 1 部と第 2 部では俯瞰を意識しています。様々なやり方・考え方・あり方を整理しており、ご自身に合うものを探したり「そんなのもあるんだ」と出会ったりすることを期待します。

第 3 部と第 4 部では、より実践的な話題を扱っています。タスク管理そのものというよりは仕事術やライフハックの文脈で聞いた言葉が多いと感じるでしょうが、タスク管理とも無関係ではなく重要なものです。

第 5 部では、テキストベースによるタスク管理のあり方を提案します。

いずれにせよ章ごとに独立した内容となっているので、興味ある部分から読み進めることができます。不明な用語が出てきたら、リンクを辿って定義を確認するなり、本書を漁って探すなりしてください。ただし第 5 部に限っては、各章を順に押さえていかないと理解しづらい構成となっています。

用語について

太字 で記します。

本書では多数の用語が登場しますが、筆者が独自に定義したものと、一般的に、または界隈で知られているものとが混ざっています。必要なら各自調べることで判別してください。そもそもタスク管理というジャンルは学術的に確立されたものではなく、色んな人や組織が独自に用語を定めているのが現状です。用語という名前は絶対的なものではなく、特定の概念を指すための便宜上のものとお考えください。

特殊な用語について

- X(旧Twitter) は X と記載します
- Cosense(旧Scrapbox) は Scrapbox と記載します

脚注について

本書では脚注の書き方が少し特殊です(※1)。

脚注元には ※1 ※2 といった印を付けます。脚注先には箇条書きで対応を並べますが、この箇条書きは章末ではなく**その印がついた節や項の** 末尾に書きます。これにより、読者は脚

注元と脚注先を行ったり来たりする距離が減ると思います。サンプルとして一つ書いてみているので、こういう感じかと体感してください。

- ※
  - 1 このような書き方を採用したのは、従来の脚注だと行ったり来たりする距離が大きかったからです。気合の入った紙の書籍であれば、レイアウトを工夫して上手く同ページ内に収められますが、本書のような Web ページベースの書籍にはそのレベルの表現力がありません。デジタルの場合は一般的には遷移してくれますが、遷移だと遷移前の情報が見えないため記憶が必要であり読みづらいです。折衷案として、行ったり来たりする距離を短くすればいいのではと考え、章末ではなく、印がついた「節や項の末尾に」書くことにしてみました。筆者としては慣れれば読みやすいし、すぐ慣れるのではないかと感じます。読み辛かったらごめんなさい。

参考文献について

参考文献ページにまとめており、本文からリンクしています。逆リンク（参考文献ページから本文の言及箇所に飛ぶ）はありませんのでご容赦ください。

また書籍については Amazon へのリンクも張っていますが、アフィリエイト目的はありません。リンク先 URL を見ればわかるとおり、アフィリエイトパラメーターを廃した純粋な URL を使っています。

注意事項

- 本書の内容に基づく運用結果について、筆者は一切の責任を負いかねますので、ご了承ください。
- 本書に記載されている名称には各社の商標または登録商標が含まれていることがあります。本書中では ™、®、© マーク等は表示していません。

筆者について

吉良野すた(sta, stakiran)と申します。

→ 吉良野すたホームページ

連絡先につきましても、上記ホームページを参照してください。

更新履歴

- 2024/09/11 v1.0
  - 初版

=== Chapter-1 □第1部「タスク管理概説」===

タスク管理とは何なのか――全体像を眺めていきます。

=== Chapter-2 タスク管理を眺める ===

タスク管理のやり方や考え方は人それぞれです、ゆえに自分に合ったものを自分で探していく必要があります。そのためには、タスク管理としてどんな要素、性質、事例があるかを眺めることが有益です。眺めた上で、自分に合ってそうなものを試していけばいい――という話をします。

この捉え方は本書の根幹となっています。**本書の目的は、色んなやり方や考え方があることを提示して、読者のみなさまが自らのタスク管理を模索するお手伝いをすることです。**

模索の重要性

本質は「人それぞれ」

タスク管理の本質を一言で言うと、「やり方や考え方は人それぞれ」です。ここから導けることも至って単純で、「自分に合ったものを探っていきましょう」となります。

そうは言ってもゼロから模索するのは現実的ではありません。そもそもタスク管理について勉強することからして難しいです。教科書もなければ教育もありません。一応、有償のセミナーはありますが、高額ですし、合う・合わないの差も顕著です。書籍やインターネットからも多くのヒントを得ることはできますが、玉石混交としており一筋縄には行きません。色んな人が色んなやり方や考え方を好き勝手に発信している――それが現状なのです。

さて、多くの人が思い浮かべているのは「ツール」だと思います。タスク管理とは何らかのタスク管理ツールを使うことに等しい、と。たしかにツールは必要不可欠ですし、下手な思想やプロセス

よりも百倍役に立ちます。しかし、タスク管理は上述したように「人それぞれ」が本質ですから、受け身でツールを使っているだけでは不十分なことが多いです。実際、色んなツールを渡り歩く方も少なくないのではないでしょうか。

自分で模索する

結局、自分に合ったタスク管理は自分で模索する他はありません。

タスク管理は趣味のようなものです。趣味と言っても消費、対戦、運動、創造、奉仕と様々ですし、何がどれだけ合っているかも人それぞれです。ついでに言えば、同じ人であっても年月が経てば変わりますし、経たなくても調子や状況によっても変わります。タスク管理も同じです。

「いいや、違う。タスク管理は仕事だからセオリーがあるはず」と考える方もいるかもしれません。たしかに、そのとおりではあるのですが、大前提として私達は人間です。抱えているものも、体力も、モチベーションの源泉も人それぞれですし、そもそも人に共通する特徴として、人は怠けますし、迷いますし、そもそも忘れます。仮に「このやり方と考え方に従っていればすべてうまくいく」という魔法のようなものがあるとして、では本当にそれに従い続けることができるでしょうか？すでに仕事、特にチームでタスク管理をされている方はよくご存知だと思いますが、仕事という真面目な文脈であってもうまくいかないのが当たり前ではないでしょうか。

タスク管理は本質的に個人的なものです。これくらいならできそうだ、続きそうだ、というバランスを探る他はありません。そのバランスは人それぞれですし、たいていは自分の細かい癖や状況も（意識的に、少なくとも感覚的に）踏まえねばなりませんから、自分自身で探る他はないのです。

魔法のようなツールはないのか？ → ないです

自分で模索し続けるだなんて、とても面倒くさいですよね。ツールでなんとかならないのか、と思うかもしれません。

現時点では「ならない」と筆者は考えます。上述したとおり、タスク管理は本質的に個人的なものであり、ツール程度でこれに寄り添えるとは思えません。2024 年現在では生成 AI が盛り上がっていますが、これも同様、サポートにはなっても全面的に頼れるものではありません。

一方で、昔からよく知られた考え方もあります。いわゆる「委譲」と呼ばれるもので、具体的には秘書や執事が有名でしょう。財力や権力を持つ人は、タスク管理などという面倒くさいことを自分でやる必要はなく、任せられます。ツールというプログラムではなく、人ですから、融通も利いて便利です。あえて解説するまでもなく、よく知られた概念だと思いますが、同時に一般人には縁のない存在です。そもそも秘書や執事が使えるからといって、人生のすべてのタスク管理を任せられるわけではありません。すべてを丸投げしている経営者や資産家はいないでしょう。結局、面倒くさい雑務を任せたり、予定というダブルブッキングの許されない事象を管理してもらっ

たり、と限定的に頼る程度になります（それでも喉から手が出るほど欲しいでしょうが）。秘書や執事を代替できるツールが登場すればありがたいですが、まだまだ時を要すると思います。

総じて、使うだけですべてがうまくいくような魔法は現時点ではありえないと言えます。

模索から逃げない

いきなり厳しい現実を突きつけてしまいましたが、大事なことです。自分に合ったタスク管理を模索していくことからは逃げられませんし、逃げている限りは辿り着けません。

とはいえ、意外と悲観するほどのことでもなかったりします。知ってみれば「なんだそんなことか」と拍子抜けすることもあります。逆に「思っているより大変だ……」「いや無理でしょ……」と絶望する可能性もありますが、そういうものです。たとえば、この先で詳しく扱いますが、タスク管理ができるかどうかは特性にかかっているところがあり、できない側の特性を持っている人にはタスク管理は難しかったりします（解としては人や仕組みに頼りましょうとなります）。

辛辣になりますが、タスク管理の現実――適性や限界というものを提示することもまた本書のテーマの一つです。それでも本質的に個人的である以上、やりようはあります。部分的に取り入れて、少しだけ楽をするのでも全然違います。繰り返しになりますが、自分に合ったタスク管理を模索することから逃げないことが重要です。社会も、人間関係も、仕事も何かと面倒くさいですが、タスク管理も同じです。面倒の荒波を自分なりに乗り越えていきましょう。

模索するために眺める

では模索しましょう、と言われても、前述したようにゼロから考えるのは現実的ではありません。幸いにも「タスク管理」と呼ばれるものにはある程度の型があります。事例や方法論やツールもあります。学問レベルで体系化されているジャンルではないものの、一個人で盛り上がっているものから、界隈で知られているもの、あるいは言葉にせずとも大多数の人が薄々知っているものまで、先人の知恵が色々ととっ散らかっています。

本書ではこういったものを言語化して、可能な限りわかりやすく提示していきたいと思います。読者のみなさまは本書を読むことで模索をショートカットできます。具体的には「ああ、そういうやり方もあるよね」と思い出したり、「それは私には合いそうにない」と適性を判断したり、あるいは純粋に「良さそう」「試したい」とか、「うーん、くだらん」「やる気出ないわ」といったことを考えるでしょう。

タスク管理は本質的に個人的なものです。どういうやり方や考え方を選ぶかはあなた次第――どうか肩の力を抜いて、店頭やECサイトで品物を物色する感覚で漁っていただけたら幸いです。

ようこそ

それでは、模索の旅を始めましょう。早速内容に入っていきます。

タスク管理は 3P で分類できる

プロジェクト、パートナー、パーソナル（個人）の 3 種類があります。

特にタスク管理というと、仕事でチームで取り組むものを思い浮かべる人もいれば、各個人が自分のタスクを管理するものを思い浮かべる人もいます。両者の違いを認識しておくことは極めて重要です。

本書では主に後者の、パーソナルな方を扱います。したがって、前者のプロジェクトの方を望まれている方にはあまり参考にならない可能性があります。それでも端的にはまとめていますし、プロジェクト向けであってもパーソナルの側の知見は役に立ちますので、ぜひ立ち寄ってみてください。

さて、3P 分類の中身に入っていきましょう。

3P 分類

タスク管理は 3P で分類できます。

名前	対象	主な性質
Project	プロジェクト（チーム）	割り当て、調整、全体俯瞰と個別詳細
Partner	パートナー（夫婦、親子）	タスクの共有とメッセージのやり取り
Personal	パーソナル（個人）	人それぞれ

プロジェクトタスク管理

仕事におけるチームなど、複数人で行うタスク管理を **プロジェクトタスク管理** と呼ぶことにします。

- ツール例
  - Trello
  - Asana、ClickUp、monday.com
  - Redmine、JIRA、GitHub Issues
  - Microsoft 365、Notion

- フレームワーク例
  - WBS(Work Breakdown Structure)、ガントチャート
  - PERT(Project Evaluation and Review Technique)、CPM(Critical Path Method)
  - カンバン
  - バーンダウンチャート

昔は主にプロジェクトマネジメントベースで考案されたフレームワークが使われており、手作業でも運用可能でした。あるいは Excel や Google スプレッドシートなど表計算ソフトウェアで記入する例も多かったと思います（未だに現役でもあります）。

現在は SaaS 型のタスク管理ツールが使われます。タスクという個別の単位をつくり、その個別ページで担当者のアサインを情報のやりとりを行いつつ、すべてのタスクをリストやボードで視覚的に並べて俯瞰できます。各担当者が独立して個別に更新できますし、管理者は全体を俯瞰して状況を把握できます。

近年では高度な機能――たとえば他ツールとのデータ連携、一部ワークフローの自動化、設定項目や分類項目の柔軟なカスタマイズ、俯瞰用ビューのカスタマイズなどを備えたものも目立ちます。キーワードでまとめると連携、自動化、カスタマイズです。

いずれにせよ、右記のような本質があります。大きなタスクを小さなタスクに分解すること、タスクに締切を設定すること、タスクに担当者（遂行の責任者）を割り当てること、複数のタスクを俯瞰するためのビューを用意すること、チームメンバーの全員が同じ場所を見ることやメンテナンスすること（一元化）。要は皆の方向性と認識をあわせるために、タスクという情報を外に出してそれを原本にしようというものです。ものすごく単純に言えば、ただの見える化にすぎません。

逆を言えば、**見える化する以上の能力は無い** とも言えます。結局口頭やチャットで催促したり、更新が形骸化してツールではなく聞きまわったり、そもそも聞いた方が早いので高頻度に会議体を設定していたり、はあるあるではないでしょうか。また目の前のタスクをひたすらこなすことに専念しがちで、改善や勉強や新規検討といった中長期的な取り組みもおざなりになりがちです。

この性質上、主に現場で使われるツールになっています。中間管理職以上のマネジメント以上の層では（現場をマネジメントする立場として関与する以外では）ほとんど使われていません。そもそもマネジメント以上の人は、扱う情報の量も、やりとりする人の数も多いですし、状況もよく変わりますから、ツールではなく自分の脳内で何とかしがちです。それがミーティングの多発という形で現れています。

総じて、プロジェクトタスク管理とは、やると決まったもの（あるいは決めたこと）を確実にこなすために、やることを外部化すること、およびそれをメンテナンスすることと言えるでしょう。

パートナータスク管理

夫婦や親子など、恒常的に一緒に暮らしている親密な少人数の家族間で使われるタスク管理です。

元々は何のツールも使わず、口頭でやり取りされるものでした。親密ゆえに遠慮も要らず、口で言った方が早いからです。そもそも「男は仕事、女は家事」など男尊女卑的な価値観が長らく続いており、家庭のタスクは妻側が一方的に抱えるシチュエーションも多かったと思われます。

次にコミュニケーションツールが台頭したことで、やりとりが行われるようになりました。最も有名なのは LINE でしょう。やってほしいことをパートナーに依頼したり、催促したり、といったことは日常的に行っているのではないでしょうか。しかし、LINE のようなシンプルなツールはメッセージをやりとりするだけですから、タスク管理としてはいささか不十分です。たとえば古いやりとりは流れてしまいますし、流れたら忘れます。また、いちいち通知も飛ぶので、あまり高頻度にはやりとりできません or してもスルーされます。

近年では Slack や Discord といった「高機能なチャット」を夫婦で使う、といった取り組みが表れています。また TimeTree や Cross など夫婦間・家族間で使う用のタスク管理アプリも登場しています。これらの特徴として、一般人でも使えること（たとえばスマホアプリとして使える）、タスクや話題といった分類観点を使用できること・保存して後から参照できること、メッセージによるやりとりができることなどが挙げられます。

要するに、パートナーという遠慮のない間柄ゆえにやり取りは必須ですし、仕事ではないので本格的なタスク管理を使うのも面倒（あるいはビジネスマンでもなければそもそもスキルがない）だからかんたんに使いたいですし、でも記録がないとこじれるからタスクでも話題でもいいから特定単位で区切って記録が残るようにしたい――といったニーズがあるのです。このジャンルを筆者は **パートナータスク管理** と呼んでいます。

今のところ筆者が観測できているのは夫婦間や家族間だけですが、シェアメイトでも可能かもしれません。

個人タスク管理

個人が自分のタスクを自分で管理する形のタスク管理を **個人タスク管理** と呼びます。本書で主に扱うのも、この個人タスク管理です。

前述したように、本質は「人それぞれ」です。たとえば本当に重要なタスクだけをこなしたい A さんがいれば、掃除や爪切りやゴミ捨てといった日常の家事も完璧にコントロールしたい B さんもいますし、予定表につっこまれた予定をひたすら消化していくマシーンと化した C さんもいます。A さんのタスク管理は、B さんや C さんには通じないでしょうし、他の人もまた然りです。

さらに、どんなタスク管理を使うかは同じ人であっても変わります。たとえば平日はマネージャーとして C さんのような予定消化マシーンと化しているが、休日は有益に使いたくて重要なタスクを優先順位をつけてこなしていく（つまり A さん的）D さんもいれば、逆に平日は予定にとらわれ

ず自分の裁量でのびのびと働きたいが、休日はアクティブに動きたいので予定をぶっこんでCさん的に過ごすというEさんもいます。

「人それぞれ」というと、まるで投げやりですが、一方で傾向はあります。ここまで書いてきた「予定に頼る」やり方は、おそらく多くの人がご存知でしょうし、何なら使っていると思います。また管理職など忙しい方は、（明示的に抗わなければ）割とこのスタイルになってしまいます。本書では、この傾向なるものも、上手く整理しつつ取り上げていきたいと思います。

もう一つ、我々は人間であり、怠けるし迷うし忘れる生き物だとも述べました。そうでなくとも集中力やモチベーションといった概念はよく知られています。いくらタスク管理が完璧でも、タスクに取り組む人間自身が上手く動けなければ意味はありません。そういう意味で、タスク管理は、実はタスク管理の周辺の要素こそが重要だったりします。この点についても、本書では詳しく扱っていきます。

狭義と広義

タスク管理には狭義のものと広義のものがあります。

狭義のタスク管理

まず **狭義のタスク管理** については、純粋にタスクを管理するという意味でのタスク管理です。特にプロジェクトタスク管理はこの側面が強いです。タスクの管理しか見ていませんので、タスク自体の是非や、タスクを扱う人、さらには環境といった視点は持ちづらいです。そもそも想定されてもいません。良くも悪くも、人はタスクを消化する機械です。

広義のタスク管理

しかし、先ほども述べたように私達は人間であり、機械のようには過ごせません。労働者としては機械的であるべきかもしれませんが、ウェルビーイングの水準が高まった昨今、歓迎はされません。また、VUCAの時代ともいわれるように、個々人が主体的な意思決定を下す必要性も増えています。そうなると、単に狭義的なタスク管理を行うだけでは事足りず、その周辺要素も考えることになります。

たとえば目標は何かとか、ビジョンとか何かとかいった「方向性」に関する話は、うんざりするほど聞かされている方もいるでしょうが、重要なことです。結局、目標やビジョンがどうあるか次第で、生まれてくるタスクも全く変わってくるからです。他にも誰とつるむかとかどういう組織にいるかといった「環境」の話や、どんなツールを使うのかなど「ツール」の話もありますし、個人的なことを言えば集中力やモチベーションといった「精神」の話、体調など「調子」もありますよね。

本当の意味でのタスク管理とは、このように狭義のタスク管理に限定せず、その周辺要素も考慮に入れるものではないでしょうか。これを **広義のタスク管理** と呼ぶことにします。

タスク管理にとらわれないということ

狭義のタスク管理にとらわれていると、そのタスクやタスクを取り巻く環境がおかしい場合に詰んでしまいます。

特に忙しいシチュエーションはただでさえ余裕がないですし、悪質な場合だと情報遮断や洗脳、あるいはそれらと同等の呪いがかけられていることもあります。こういうと仰々しくて、縁がなさそうに感じてしまうかもしれませんが、他人事ではありません。精神的に心身を崩して退社・休職した人や、おかしな言動をするようになった人、依存していて抜け出せていない人などは決して珍しくないと思います。このようなときは、タスク管理は無力です。一時しのぎにはなっても、いずれ力尽きます。自分が好きで一世一代の勝負をしているのなら構いませんが、そうでないのならただの自滅行為です。

たとえばブラック企業に勤めて搾取されている場合、タスク管理を習得してもっと仕事をこなせるようになるぞー、とはならないでしょう。そんな場所からは逃げる方が優先度が高いと思います。連日、徹夜クラスの深刻な睡眠不足が続いている場合も、タスク管理で何とかしようとは思わないでしょう、まずは寝て体調を戻すべきですし、そもそも睡眠不足が続く仕事の仕方そのものが間違っているわけで改善（できないなら逃避）するべきです。

と、これらは極端な例ですが、軽い例であれば日常の至るところに存在します。特に社会人の方であれば、くだらない茶番や習慣、手続きや様式や文化には思い当たりがあるはずです。タスクそのものを疑い、対処するべきシチュエーションで「いやもっと頑張ろう」「頑張るためにタスク管理を工夫しよう」などと捉えるのは、愚かですらあります。もちろん、だからといって怠け癖、逃げ癖がつくのも考えものですが、ご自身の心身と時間と思いはもっと尊重したいところです。組織的な場面だと、複雑で重たい力学が働いているので自分ひとりでは何とかならないケースも多いですが、それでも自衛することはできます。

タスクで忙しいときは、まずは広義な目線で疑ってみることをおすすめします。行動が必要なら、一気にやるなり準備するなりしてやっていきます。別に失敗してもいいのです。広義的な問題を見て見ぬふりをして消耗し続けるよりはマシでしょう。広義な目線で行動しない限り、状況は続きます（運が良ければ好転することもありますが所詮は神頼みです）。

ここまで長々と小難しく書きました。当たり前のことを言っていると思われるかもしれませんが、これが案外難しいのです。知らないと発想を持てませんし、知っているだけでは（特に忙しくて余裕がないときには）やはり頭が切り替わりません、また他人から冷静に指摘されるとイラついて反抗したくもなりますよね。一筋縄にはいきません。だからこそ、きちんと向き合い、上手く扱えるようになりたいところです。本書でもこの広義的な視点はしばしば登場します。

タスク

そもそもタスクとは何でしょうか。

タスクとは

タスク(Task) とは何でしょうか。仕事、やること、TODO、アクション（行動）、アクティビティ（活動）、プロジェクト――様々な呼び方があります。

正解はありません。というより文脈によって変わります。個人タスク管理であればその個人が決めればいいですし、チームで仕事に取り組む場合はそのチーム特有の定義があるかもしれません。またタスク管理のツールやメソッドにおいても定義があるかもしれません。そうかと思えば、定義が存在せず解釈が各人に委ねられていることもあります。

以下にいくつか例を示します。

- 1 タスクとは「やること」全般である
- 2 タスクとは「やることが確定した事柄」である
- 3 タスクとは「締切」と「担当者」が設定されたものである
- 4 タスクとは「一回の行動で終了させることはできないが、短期的な着手で終了に導ける程度の粒度を持つ事柄」である
- 5 タスクとは「具体的な手順や達成条件が定義された事柄」である

1 は最も意味が広くて、ほとんど何でも含まれてしまいます。2 は「やることが確定した」とあるので、会社の仕事や上司からの命令、あるいは役所手続きなどクリティカルな私事などが当てはまるでしょう。3 は締切と担当者に言及しており、プロジェクトタスク管理で見かけそうな定義ですね。4 は小難しく書いてますが、要するに「複数回着手すれば終了できるサイズの仕事」とでも言えましょう。これはタスクというよりプロジェクトと呼ぶのが似合いそうです。最後に 5 ですが、具体的であることを強調しており、これも仕事の文脈って感じがしますね。

どれが正しくて、どれが正しくないといったことはありません。どれも正しいです。どの定義も、それぞれ有効なシチュエーションがあります。自分にとって、あるいはチームや家族にとって必要な定義を使えばいいのです。

悲劇を回避する

もっとも通常はツールを使う形でタスク管理を行いますから、いちいち定義なんて意識しません。しかし、それでも事実上存在はしています。タスクの定義を意識することで、タスクではないもの（定義から外れた異分子）を無理やりタスクだと扱う悲劇を減らせます。

たとえば 5 の「具体的手順と達成条件が定義されたものがタスク」という定義に基づいている

場合に、手順や条件が曖昧なタスクがあったとしましょう。これに対しては「それはタスクじゃないです」「手順を具体的にしてください」「できないのならそもそも登録しないでください」といった形でガードできます。秩序を保てるわけですね。秩序を保たないと、具体的でない曖昧なタスクが混ざり込んでしまい、あとで見返したり更新したりするときに混乱します。タスク一個だけなら大した手間ではありませんが、積み重なると厄介です――最終的にはタスク管理そのものが形骸化してしまいます。

もう一つ、もっと個人的な管理を例にしましょう。締切が無いと腰が上がらない A さんがいるとします。この場合、A さんにとってのタスクとは「締切を持つ事柄」のような定義であるのが良いでしょう。そうしないと、いつまで経っても腰が上がらないからです。腰が上がらないだけならまだしも、締切の無いタスクもどきがたくさんあっては、本来のタスクさえも見失う可能性があります。タスク管理そのものに嫌気が指して挫折するかもしれません。何らかのツールを使ってタスク管理をするわけですが、A さんの場合、きちんと各タスクに締切を設定するべきでしょう。設定漏れを通知する仕組みがあればなお良いですし、A さん自身のキャパシティ（どれだけのタスクを抱えられるか、また捌けるか）を理解した上で総量を調整できたらさらにグッドです。

このように、タスク管理を行う人（達）自身が想定する「タスク」とは何かを定義し、定義したタスクだけを扱うようにするのが望ましいです。定義が堅苦しければ、大雑把な認識でも構いません。「私（達）にとって、タスクとは大体こんな感じのものを指す」くらいの認識でも、あるのとないのとでは違います。

タスクとそうでないものを区別する

ここまでタスクとは自分なりに捉えるものだ、のような話を書きましたが、あるセオリーを押さえておくとだいぶ楽ができます。タスクと「タスクのように見えるが、その実、そうではないもの」とを区別することです。

このようなものを **オルタスク(Altask)** と呼びことにします。Alternative Task、略して Altask です。直訳すると「代替タスク」となります。タスクの代わりに行うもの――タスクとは異なるやり方で対処する必要があるもの、との意味を込めています。

オルタスクとしては以下が存在します。

- 入れ物
  - フォルダ、カテゴリー、タグといった分類の概念
- 予定
- ビジョン、理念、標語や心がけといったもの
- メモ
- 習慣や日課

詳細は後の章で述べます。ここでは軽く取り上げるに留めます。

一番わかりやすいのは予定でしょう。予定はカレンダーというツールで管理するのが王道です。開始時刻を守らねばならないので、うっかり忘れないようにする必要もあります。出社して「場が賑やかになったり誰かが教えてくれたりするから気付ける」ようにしたり、リマインダーを設定して音を鳴らしたりといった対策があります。あるいは人と会う予定であれば、相手にフォローしてもらうこともできますよね（そうでなくともやりとりがいくらか発生しますから思い出しやすい）。

習慣や日課については、意外と思われるかもしれませんがタスクではありません。オルタスクです。というのも、習慣や日課からタスクが生まれる、という対応だからです。たとえば「早寝早起き」という習慣はタスクではありません（し、タスクとして管理してもおそらく役に立たないでしょう）が、この習慣から「朝 6:00 に起きる」「夜 22:00 に寝る」「夜 22:00 に寝るための諸々の準備を……」といったようにタスクが生じます。もちろん定着した習慣であれば管理などしませんが、新しく習慣をつくる場合は別です。習慣からどんなタスクが必要かをブレイクダウンした上で、それらを管理する必要があるわけです。もっと言えば、このようなやり方や仕組みを確立できれば、（学校や会社やパートナーといった外部要素に頼らずとも）必要に応じて習慣をつくったり削ったりできるようになります。このような営みはタスク管理だけでは行えません。習慣というオルタスクに対する方法も必要なのです。

このとおり、オルタスクは奥が深いものですが、まずはオルタスクがタスクではないこと、そしてタスクとは違った扱い方が必要になるという点を押さえてください。オルタスクをタスクとして管理しても上手く行きません。

タスク管理

タスクについて取り上げました。ではタスク管理とは何でしょうか。

タスク管理とは

タスク管理(Task Management) とはタスクを管理することです。

管理とは何でしょうか。先ほどから「人や状況による」のような言い方ばかりしていますが、すみません、管理についてもそうです。

管理のニュアンス

まずは「管理」のニュアンスを見ていきましょう。

MCA の 3 つに大別できると思います。元ネタは右記です: 1 2

名前	解説	一言で
Management	あらゆる手段を使って「うまくやる」	総合格闘技
Control	計画や基準のとおりに動いているかを「保証する」	ギャップを埋める
Administration	方向性や仕組みを定めるなど「舵取りをする」	トップダウンな支配

続いて、これを用いて●●管理のニュアンスを整理します。網羅は難しいので主なものを挙げます。

名前	MCA	解説
進捗管理	Control	期限までに完了できるか、進行ペースは問題ないか
完了管理	Control	何ができたのか、何を終えたのか
見積管理	Control	完了にかかる時間はどの程度か、正確な予測をするにはどうしたらいいか
会議管理	Management	うまくいくためにどんな会議体を設定するか、スポットや緊急で誰と何を打ち合わせるか
稼働管理	Administration	人材のリソースをどう定義するか、何を持ってリソースの消費とするか ※1
在庫管理	Administration	あとでやるタスクや中長期的にやるタスクをどう管理するか ※2

こうして見ると、Control（ギャップ埋め）とAdministration（トップダウンな支配）が多いことがわかります。

Management は少ないですが、会議はわかりやすいでしょう。定例会議やレビューなどの会議体設計だけならAdministration ですが、現実はそう甘くはなく、割り込みや緊急はあります。そうでなくとも雑談などカジュアルなコミュニケーションもありますし、1on1（※3）の考え方もすっかり定着しています。リーダーやマネージャーの腕の見せ所でしょう。まさに総合格闘技です。

- ※
  - 1 典型的なのは工数管理です。工数 =（所要時間）x（単金）
  - 2 バックログ、いつかやるリストなど、この手のタスクをプールし定期的に棚卸しする考え方はよく使われます。
  - 3 1on1 とは上司と部下が1対1で話し合う打ち合わせのこと。シリコンバレー発祥で、国内ではヤフー（現 LINE ヤフー）の取り組みから広まったとされています。内情は雑談重視だったりキャリアに関する真面目な対話の場だったりと様々で、筆者としては 1on1 を見ればその企業の体質がわかると思っています。

タスク管理は Management

一方で、タスク管理は英語では Task Management といいます。Control でも Administration でもなく、Management です。これはここまで説明してきた「タスク管理は本質的に個人的なもの」「状況次第で最適解も変わる」とも一致する捉え方ではないでしょうか。要するにうまくやれれば何だっていいのです。Control や Administration は手段としてはよく知られていますし、効果も出ますが、それだけではないのです。

本書ではタスク管理と呼ばれるものの中身を色々と紹介していきますが、どれをどう使うかはあなた次第です。タスクを Management するために、枠にとらわれず、使えるものは使って、使えないものは捨てて、と自分なりに模索していきましょう。

タスク管理の公理

「タスク管理」についても、これまでの歴史の中でおおよそ共通されてきた前提があります。これを **タスク管理の公理** と呼ぶことにします。公理というのは、理論の出発点としてとりあえず正しいとみなす事柄です。まさに前提です。

タスク管理の公理は、以下の 2 つです。

- 1: ツールの公理
  - タスク管理には必ず何らかのツールを使う
- 2: 解釈の公理
  - ある事柄がタスクであるかどうかは人が決める

ツールの公理

ツールの公理は、要するに **頭の中だけで完結するものはタスク管理ではない** ということです。

そもそも頭の中で完結するのならタスク管理なんて面倒くさいことはしなくていいですし、実際頭の性能に自信がある方はやろうとも思わないでしょう。しかし、人間ひとりが扱える情報量などたかが知れていますし、人間は怠ける迷う忘れる生き物でもあります。この限界を越えるために、タスクという形で外に（ツールに）出して、その出したものをメンテナンスするのです。

プロジェクトタスク管理など複数人で行う場合も、やはり外に出すことが必要です。そうでないと、いわゆる「言った・言ってない論争」が起きますし、状況を確認する際にいちいち尋ねたり会議したりが必要になって面倒です。ツールに出しておくと、そこを見たりそこを更新したりすれば済みます。

解釈の公理

解釈の公理は、ここまで散々強調してきたことでもあります。ある事柄がタスクなのかどうかはその人次第ですし、同じ人であっても変わります。

極端な話をすると、本書を書くことは 2024/04/30 現在の筆者にとってはタスクですが、筆者以外の全人類にとってはタスクではないでしょう。また一年後はおそらく筆者にとってもタスクではなくなっているはずです。もう少し広げて「タスク管理を体系化する」「タスク管理を啓蒙する」と捉えた場合、これをタスクと捉えている人は複数人はいそうです。筆者もそのひとりです。

別の言い方をすると、タスクかどうかを決めるという**意思決定こそが重要**だとも言えます。私達の大半は労働者であり、上から降ってきたタスクを消化する機械と化している側面があると思いますが、それらのタスクも結局は誰かが決めたことにすぎません。ここまで書いてきたように、個人タスク管理では主役は自分自身なので、何がタスクかを決めるのも自分自身です。またプロジェクトタスク管理についても、ただ言われるがままにタスクと向き合うのではなく、自分なりに疑ってみることも大事です（広義のタスク管理の項で述べました）。

現実的には「降ってきたタスクに従うしかない」のかもしれませんが、公理としてはそうではない、決めの問題なのだという点はぜひ押さえてほしいと思います。

タスク管理はスキル

公理から言えることは何でしょうか。

まずはツールが重要だということです。

ツールというと何らかのアプリやサービスをイメージしますが、冷蔵庫にぶら下げる落書きボードやペンで書く手帳などアナログなものも含みます。またメソッドやプロセスやフレームワークといった概念的な道具も含みます。すでに「タスク管理のやり方や考え方を意識せず、とりあえずツールを漁って使ってみる」タイプの人も多いと思いますが、これは言い方を変えれば、ツールにはそれだけの力があるということです。タスク管理を知らなくても、使うだけでタスク管理ができてしまうというポテンシャルがあるのです。

次に意思決定が重要だということです。

何をタスクとみなすかは決めの問題です――と言うは易しですが、これは案外むずかしいことでもあります。決めることには責任が伴うので、負いたくない人は決められません。また決断が苦手という特性もあります。そもそもそのあたりをクリアしていたとしても、では何に基づいて決めるのかという判断指標も必要です。判断指標をどうやってつくるのか・育むのかも考えないといけませんね。また意思決定は疲れるものでもあり、注意資源という言葉もありますが、ヒットポイントのようなものです。睡眠しないと回復しません。ということは、闇雲に欲張るのではなく取捨選択が入りますよね。いわゆる習慣だとか生活の改善だとかいった部分も絡んできます。

マネージャーや経営者などは意思決定が仕事のようなものですが、逆を言えば意思決定は高度な営みとも言えるでしょう。タスク管理、特に何をタスクとみなすのかという広義な目線に立つ

ということは、この高度な営みを始めることを意味します。一朝一夕には身につきません。判断指標のつくりこみやメンテナンス、意思決定という経験の蓄積、注意資源の効率的な使い方 etc、継続的に取り組まないと身につきません。要は鍛える必要があります。

ツールと意思決定。言ってしまえば、これらを行うためにはスキルが必要です。そういう意味で、**タスク管理はスキルなのです。**

まとめ

- タスク管理とは
  - 本質は人それぞれ
  - 自分なりに模索する必要がある
  - 魔法のツールはなく、模索から逃げないこと・向き合い続けることが大事
- 本書の方向性
  - 模索するためには、色々なヒントを眺められたらいい
  - 本書ではまさに「タスク管理」なるものの中身を色々と紹介する
  - 読者は店頭で品物を漁るつもりで、ヒントを漁ってみると良い
- タスク管理は 3P に分類できる
  - Project プロジェクトタスク管理、チーム
  - Partner パートナータスク管理、夫婦や家族
  - Personal パーソナル タスク管理、個人
- タスク管理には狭義と広義があり、広義の目線も重要
  - 単にタスクを管理するだけなのが狭義
  - 現実的には狭義を超えた目線が必要
  - ブラック企業など、極端な例を浮かべるとわかりやすい
- タスクとは
  - 状況次第なので、状況が想定する定義を探すこと
  - タスクとそうでないものを区別すること
  - 特にオルタスクは覚えておくと良い。タスクとは異なるアプローチが必要
- タスク管理とは
  - 管理という言葉自体は Control（ギャップ埋め）やAdministration（支配）を指す事が多い
  - しかしタスク管理は Task Management であり Management なので、うまくいけばそれで良い
  - ツールの公理と解釈の公理も意識したい
    - タスク管理とはスキルである

=== Chapter-3 個人タスク管理を眺める

＝＝＝

前章ではタスク管理そのものを眺めました。タスク管理にはプロジェクト、パートナー、パーソナル（個人）の 3 種類があるのでしたね。本章ではそのうち「個人タスク管理」を眺めたいと思います。

個人タスク管理には様々なやり方や考え方がありますし、究極的には自分に合ったやり方を自分で模索していくものです。一方で、おおよそのパターンも見えてきています。本章では主なパターンを紹介していきます。ご自身のやり方や考え方と照らし合わせてみることで、良さそうなタスク管理とは何かのあたりをつけることができるでしょう。

タスク管理する？しない？

そもそもタスク管理をするか、しないかという話があります。

最初にまとめておくと、以下のような内容を扱います。

- 特性や状況によるところが大きい
  - 特性
    - 頭の中で処理できる人には要らないし、処理できない人には重宝する
    - 運動・トレーニングの側面があり、地道な継続と向き合えるかが鍵である。向き合える人には向いている
  - 状況
    - 慌ただしい状況下だと使いづらい（臨機応変に場当たりの対応をしたり瞬発的にやりとりしたりで凌ぐ世界）
    - 几帳面で自分のリズムやペースに従って生きたい人には重宝する。生活の質も上げられる可能性が高い
- 向いてないからといってできないわけではない、自分なりに取り入れていけばよい

タスク管理が要らない人

前章にて公理を取り上げましたが、タスク管理にはツールと解釈が必要なのでした。**ツールを使わず頭の中だけで処理できるならタスク管理は要りません**。また生活に余裕がある人や仙人のような生き方ができる人など管理しなくても済む人にも要りませんし、人や組織に言われたことをやるだけの人にも必要ありません――つまり **自分にとって何がタスクなのかという解釈をしなくてもいい人には必要ない** のです。そもそもしようとも思わないのではないでしょうか。ツールを使うにせよ、自分で解釈をするにせよ、面倒くさい営みです。馴染みのない人がいきなり始めるのは難しいですし、そもそも必要性がないのだから始める意味もモチベーションもあ

りません。

逆に非常に忙しくて頭だけでは処理しきれない人や、指示命令だけではなく自発的に取り組んでいく人、また指示命令的であっても何をどの順番でこなすか考えなくてはならない人にはタスク管理は必須です。タスク管理をしなければ振り回されたり迷ったりするばかりで、状況は好転しません。しかし一方で、状況などたかが知れているし、コントロールできるものでもないから管理など無用だ、今を生きればいいだけだろう、という考え方もできます。実際、そうやってがむしゃらに生きている人は少なくありませんし、このような精神論は誰でも一度は見聞きしたことがあるのではないでしょうか。

タスク管理が要らない人の特性

薄々自覚または痛感している人もいるかもしれませんが、**タスク管理しなくても生きていける人にはある種の特性があります**。バイタリティに溢れ、メンタルが図太くなければいけません。わかりやすいのは、部屋や（PCの）デスクトップを散らかしている人たちです。汚部屋とかずぼらという表現もありますが、このような人たちは頭の処理能力が高く、メンタルも図太い傾向があるため「まあ何とかなるだろう」と楽観視しやすい（かつ実際に何とかしていける力がある）のです。何とかしていけるのですから、面倒くさいタスク管理に手を出すモチベーションはありません。

自己啓発的なことを言えば、カーネギーの名著 道は開ける にて「悩むのは暇だからだ」との金言があります。慌ただしい状況に身を投じて、あくせく働いているうちは悩まなくていい（し悩む暇もない）ということです。慌ただしい状況では管理うんぬんよりも臨機応変な取捨選択と会話が最重要であり、そもそもタスク管理の出番はありません。あるいはせいぜい直近何をするかの可視化と後々何したかの知るための記録として機能する程度でしょう。タスク管理などというまどろっこしいことをせずとも、目の前の忙しさに専念しているだけで済むのです。

逆にタスク管理とは切っても切り離せない人もいます。頭の処理能力が低い人は、頭だけでは進められないので情報を外に出して、それをメンテナンスしながら進めていく必要性に駆られます。また神経質な人は、いつ何をどのくらいの頻度で行うかというルーチンをストイックに構築することもあります。いずれにせよ、慌ただしい状況下を泳いでいける力はありません。自分なりのペースを死守する営みが生じます。よって、誰に言われるまでもなくタスク管理、あるいはそれに近しいことをしています。このような人達にとってはタスク管理は必須でしょうし、タスク管理について学ぶことでさらに生活の質を上げられます。

するか、しないかの目安

色々書きましたが、タスク管理をするか、しないかの目安は一言で言えると思います。

現状すでに私生活でタスク管理（ツールを用いることとタスクであるかどうかの解釈を能動的に行っていることの両方）を、あるいはそれに近しいことをしているかどうか。

「私生活で」と書きましたが、これは仕事などで強要・推奨される場面ではなく、使うかどうかは自由だけど自分の意思で使っているかと問いています。している人は、タスク管理は必要でしょう。おそらく苦戦されているとも思います。本書にはたくさんのヒントがあると思いますので、自分に合ったものを漁ってみてください。

していない人は、タスク管理は不要でしょう。そもそもしようとも思っていないのではないでしょうか。あるいは、周囲や常識の影響で「した方がいいのかなぁ」と考えてはいるが、本心では別に要らないと思っているかもしれません。タスク管理は大変な営みであり、それなりのモチベーションを要求します。本心として「要らないよね」「くだらない」「やってられるかよ」といった気持ちがある場合は、おそらく続かないので、無理して手を出さなくても良いと思います。出したとしてもたぶん続かないでしょうし、挫折した人（あるいはし続けている人）も多いのではないでしょうか。

トレメント

タスク管理は、たとえるなら運動のようなものです。運動は筋トレにせよ、スポーツにせよ、もっと軽いアクティビティにせよ、体を動かすという「本質的な面倒くささ」が伴います。これを **トレメント(Trement, Training Element)** と呼ぶことにします。

トレメントを好きになれるかどうか、あるいは好きじゃないけどまあ続けることはできますねと思えるかどうかは重要です。上記の目安も直感的なものであり、あなたがトレメントを相手にできるかどうかを問うています。

ここで一つ疑問が生じます。

トレメントと向き合えないとタスク管理はできないのか、と。

回答すると、「正直不利ではあるが、できないことはない」となります。幸いにもタスク管理のやり方や考え方は色々ありますし、トレメントが苦手な人でも取り組めるよう工夫されたものもあります。

一つ例を挙げると、ポモドーロ・テクニック は有名でしょう。これは 25 分の集中と 5 分の休憩を（キッチンタイマーなどを使って）ストイックに繰り返すことで集中のリズムをつくっていくものです。トレメントに強い人は「いやそんなことしなくても集中なんてできるでしょ」と捉えますが、それは強いからなのです。弱い人は、集中するのも一苦労なので、このような制約のかけ方に頼らないといけないのです。ポモドーロ・テクニックは、わかりやすい割には強く働いてくれます。

他にも Habitica や Chill Pulse などゲーミフィケーション要素を取り入れたツールもあり、ゲームで進捗を進めるような楽しさとともにタスク管理していけます。

そもそもあらゆるやり方を全部取り入れる必要はなく、ちょっと使うだけでもいいのです。自炊と同じです。別に料理人のようにつくれずとも、野菜と肉を切って鍋やフライパンに放り込むだけでもそれなりのものはつくれます。極めれば「雑な鍋料理だけで食事を完結する」こともできるでしょうし、そもそも一切自炊せず 100% 外食で済ます戦略も可能です。トレメントが苦手だか

らといって何も出来ないわけではないのです。

もっと言えば、どこまでならできるのか、何が合っているかには個人差があり、トレメントとどこまで向き合うかも含めて自分自身で探していく他はありません。しつこいですが、タスク管理は個人的なもので、自分自身で模索していくものなのです。

確実性か、納得性か、パフォーマンスか

タスク管理がカバーするものは多数ありますが、3 つに集約できると思います。

- 忘迷怠（ぼうめいたい）
  - 忘れないこと、迷わないこと、怠けないこと
  - 「確実にこなしたい」
- 調動脈（ちょうどうみゃく）
  - 調子、動機、文脈
  - 「納得したことがしたい（納得できないことはしたくない）」
- 熱夢集（ねつむしゅう）
  - 熱中、夢中、集中
  - 「パフォーマンスを最大限に引き出したい」

以下、各々の詳細を見ていきましょう。

忘迷怠

忘迷怠（ぼうめいたい） とは忘れること、迷うこと、怠けることの総称です。本書では文脈により忘・迷・怠そのものを指していたり、忘迷怠の防止や軽減を指していたりしますが、上手く読み取ってください。

タスク管理と聞いて、まず思い浮かぶのは忘迷怠でしょう。タスクという形で残しておくことで忘れない（正確には忘れても見れば思い出せる）ですし、それをやればいいのだともわかるので迷いません。やったものはチェックを付けて、まだ終わってないものに着手して、それも終わったらチェックをつけて――とリズムに乗れば怠けることなく消化もしていけます。

「そんな上手くいくわけないよね」と思われるかもしれません。そのとおりです。私達は意思を持った怠け者であり、機械ではないので、タスクを書き残していたからといって 100% そのとおりに行動するとは限りません。何から「タスクは見たし、認識もしたし、このあと行動するつもりだった」のに実際しなかった、みたいなサボタージュもよくあります。それでもタスクが目に見えないよりははるかにマシです。何も残していないと完全に気分と気まぐれ（あとは自身の頭の処理能力）に依存してしまいますが、まがいなりにも残して見える化しておけば頼れるのです。

もちろん単に残せばいいというものではありません。TODOリストという言葉は有名ですが、同時に「役に立たない」ことでも有名です。TODOリストとはTODO（やること）を書き並べたリストであり、タスクリストと同義と捉えてもいいものですが、単にタスクを並べたものにすぎません。極端な話、タスクが100個並んでいるとして、それを全部忘れず迷わず怠けず消化できるかというと、できないでしょう。

タスク管理ツールの存在

だからこそタスク管理ツールが存在します。優先順位とか、カテゴリーとか、状態（未着手・進行中・完了）とか締切とか開始日とかいった属性を導入して、タスクごとに値を設定しておけば、色々と便利です。たとえば「終わってないタスクだけを表示する」とか「優先度が高いタスクだけを表示する」とか「締切まで間もないタスクを表示する」とかいったフィルタリングができますし、特定の条件で整列させて見やすく俯瞰したりもできます。ただタスクが並んでいるよりははるかにマシです。

ただし、ツールを機能させるためには、日頃からタスクとその属性を細かく更新する必要があります。仕事だとプロジェクトタスク管理の形で更新します（させられます）が、個人だとそうもいきません。先ほどトレメントを取り上げましたが、日々トレーニングを積むように自ら行動できる人でないと中々できません。少しでも行動に起こさせるために、ツール側も日々進化しています。たとえば綺麗なデザインだったり、カスタマイズ性が豊富だったり、前述しましたがゲーミフィケーションを取り入れたり、と色々あります。また、稀にタスク管理ツールを自作する人がいますが、これは「自分が作ったものを自分で使う」というDIY的モチベーションを生むのに重宝します。

動線の重要性

つまり忘迷怠とは **ツール上に存在するタスクを見れば、ある程度は（忘れる・迷う・怠けることを）防げるよね** というだけの話です。100%では到底なく、ツールを使う私達自身の意思や適性やスキルは変わらず求められますが、それでも何も使わないよりははるかにマシなのです。

さて、この忘迷怠の質を上げるためにはどうすればいいのでしょうか。細かいテクニックは後の章に譲るとして、一つだけ最も重要なことを取り上げます。**動線** です。

導線ではなく動線です。「生活の動線」といった言い方をしますが、この動線です。意味としては「毎日頻繁に通る場所」となります。通るというと物理的な場所に限定されがちですが、そうとも限りません。たとえば自宅のトイレ、自宅のトイレのドア、自宅の冷蔵庫、スマホのホーム画面、PCのデスクトップ画面、SNSのタイムラインなどはすべて動線です。ただし人によっては動線は違います。筆者は私有のスマホは持っていないのでホーム画面は動線になりませんし、長期的な旅行や出張をしている人は自宅は動線にはならないでしょう。

さて、タスク管理ツールの話に戻りますが、私達がツールを用いて忘迷怠を担保するには、ツールを見る(Read) → タスクを選ぶ(Select) → 行動する(Act) → 行動後ツールに戻ってきて状

況を更新する(Feedback)、のステップが必要です。一巡したらまた Read から始めます。これを **RSAF サイクル** と呼ぶことにします。

では、RSAF サイクルをまわすにはどうすればいいでしょうか。もうおわかりですね、**ツールを動線に組み込む必要があります**。たとえば一日何十回と目を通す存在にしないといけません。動線に組み込みさえすれば、一日のうち何回も見るはずなので（実際に書かれているとおりの行動ができるかはさておき）少なくとも目には入ります。目に入れば忘迷怠にも繋げられます。逆に、そもそもツールを見ることができなければ話にならないのです。

動線以外の方法

忘迷怠には動線が必須なのでしょうか。

そんなことはありません。他にも色々な方法があります。カレンダーや環境の力に頼ることは言うまでもなく知られています。ツールを動線に組み込む、というと小難しいですが、カレンダーであれば、すでに動線になっている方も多いのではないでしょうか。一日に何度も見ますよね（※1）。あるいはパートナーや家族がいらっしゃる人は、フォローしてもらえるので忘迷怠を防げるでしょう。特に子供は、よほどしっかりしてなければ、基本的に親からのフォローで動くはずです。

現実的には様々な方法を上手く組み合わせるのがベストです。筆者の例を挙げると、基本的に PC 上の自作ツールが動線ですが、カレンダーも使いますし、家事に絡むタスクは自宅の冷蔵庫や玄関や浴室前も動線となっています。最後の浴室を取り上げますと、今日リモートワークが終わったら風呂掃除するかなーと思ったら、浴室前に掃除セットを置いておくのです。そうしたら、風呂に入る前に目に入るので「あ、掃除するんだった」と思い出せます。洗面所やトイレに行くときも目に入るので、一日何回も目に入ることになります。何回も目に入るなら、一回くらいは「掃除するか」となります。このやり方は、ツール上に「風呂掃除する」というタスクを書いておくよりも効果があります。あくまで筆者の例です。筆者はトレメントに耐えられる側の人間なのでこれで十分ですが、どういうバランスが適しているかは本当に人によります。

- ※
    - 1 カレンダーは本当に優秀なツールだと思います。過去の予定を更新しなくてもいい（放置しておけばいい）のが地味に便利ですよね。RSAF サイクル的に捉えると、最後の Feedback が不要なのです。この「ちょっとした手間の無さ」は大事で、カレンダーが親しまれている理由の一つだと思います。

調動脈

調動脈（ちょうどうみゃく） とは調子、動機、文脈の総称です。調子とは体調やコンディションのことです。好調とか不調とかありますがそれです。動機とはモチベーションを指します。やる気とも言いますね。**文脈（Context, コンテキスト）** とはそのタスクの前提、状況、背景、あるいはそのタスクが備える性質などを指します。重要かつ奥深い概念なので太字にしていますが、

詳細は後の章で扱います。

調動脈の観点はおざなりになりがちですが重要です。タスク管理と聞いても浮かびにくいもので、狭義か広義かで言えば広義に入ります。それゆえ軽視されさがちなのですが、軽視がもったないほど大事なものです。特にわかりやすく言うなら、「体調ややる気も大事です」と言われると納得できるのではないでしょうか。あまりに当たり前すぎることではありますが。

ここで「いやタスクは仕事なんだから体調ややる気は関係ない、とにかくこなすべきだろう」と思うかもしれません。そういう場面があることも否定はしませんが、常にそうだと息切れしてしまいますし、これは筆者個人の考えですが、自分の調子ややる気は尊重するべきだと思っています。ワークライフバランスやウェルビーイングなど、自分自身を大事にする価値観が色々と盛り上がっているのも時代がそうなっているからです。水準が上がっているのです。自分の意思でとにかく頑張る（頑張らないといけない環境に自ら飛び込むことも含む）のなら構いませんが、蔑ろにされるのはいただけません。そうでなくとも、ないよりはあった方が良いに決まっています――と、少し筆者の主張が強くなってしまいましたが、これが絶対的に正しいかどうかはさておき、本書では調動脈も軽視しません。

納得感の最大化

調動脈を考慮するとは、納得感を最大化すること に他なりません。納得できれば良し、という前提のもと、ではどうすれば納得できるかというと、調動脈を踏まえればいい、となります。

調子については、体調やコンディションが良いと納得感が高いと思います。逆にこれらが低いと納得感も低くなります。特に精神的に参っているときはネガティブ思考になりがちですし、身体的に疲れているときは何もかもが面倒に感じますよね。納得感を高めるためには、体調やコンディションを高めればいいです――というのはかんたんですが、はい高めますといって高まるものではありません。どちらかと言えば日頃から、あるいは中長期的に体調を維持したり、どんなときにコンディションが良くなるか（悪くなるか）を把握してそれに則ったり、といった行動が必要になります。

動機については、やる気のことですのでわかりやすいと思います。やる気があるなら納得もできますし、ないならできません。納得感を高めるためには、やる気を高めればいいのですが、ではやる気とはどうやって高めるものでしょうか。そもそもやる気とは何でしょうか。心理学や脳科学や哲学を深追いする気はありません。肝心なのはメカニズムを知ることよりも、どうやったらやる気が出るのか、増えるのかを知り、行動することです。端的に整理すると、次のようになります。

- 明確にする
  - 何すればいいかわからないタスクを分解する
  - 何がわからないのかもわからないので、いいからとりあえず始めてみる
  - 目標に紐づけて進捗感や意義を見えるようにする etc
- 浸透させる
  - 習慣、日課、こだわりなど労力なく勝手にこなせるようにする

    - 体力やスキルを鍛えて、抵抗感や消耗具合を減らす
    - 仕組み化や自動化や単純化などでシームレス（垣根をなくす）にして、腰を上がりやすくする etc
- とにかく始める（ための環境を整える）
    - 始めやすくするための環境をつくる
    - 環境を見つけて通う
    - 誘惑を減らす etc

最後に文脈については、そのタスクの文脈がわかれば納得感が増すこともあるということです。たとえば何のために行っているのかわからないタスクAと、これこれこういうために必要なんだよと丁寧に説明されたタスクBとでは、後者の方が納得感があります。タスクBについて、その文脈（この場合は取り組む意義）がわかっているからです。なので、文脈の観点で納得感を高めたいのであれば、文脈を知りに行けばいいことになります。調べたり、見聞きしたりはもちろん、自分の内面と対話したり、自己分析して現状や適性を踏まえたりといったことが必要になることもあります。やり方に個人差が出るのも特徴で、筆者は自分ひとりで考え込むのが好きですが、他者と話すのが好きな人もいれば、一見すると話し好きに見えるけど実は観察をしている（つまり観察こそが好き）人もいますし、事例や資料や論文といった情報を調べるのが好きな人もいます。

好きこそものの上手なれ

好きこそものの上手なれ、ということわざがあります。上手になるためには継続が大事ですが、好きじゃないと中々継続できません。

このことは、調動脈を考慮して納得感を高めることについても言えます。ここまで調動脈について軽く紹介しましたが、能動的かつ継続的な行動が必要になりがちです。ここで効いてくるのが興味や好奇心、好き嫌いや性癖、信念といった個人の嗜好なのです。嫌いなことやどうでもいいことは長続きしません。

「いやお金があれば関係ないよ」と思われる方もいるでしょうが、筆者は懐疑的です。お金のために、と言っている人に限って、実は仕事と嗜好がマッチしていたりします（※1）。「家族のためだから関係ない」も同様です。それで続いている人は、続くだけの嗜好を持っているのだと思います。

- ※
    - 1 お金のような外からの報酬は「外発的動機づけ」と呼ばれます。興味関心や信念をベースとした内発的動機づけよりもやる気を保ちにくいことが知られているそうです。一方で、現実では、お金によってやる気を維持している人もよく見かけます。どういうことでしょうか。筆者としては、単に「お金以外の内発的動機づけが実はある」だけかなと思っています。仕事は楽しいという興味関心だったり、お金がこれだけ貯まったらこういうことができるから絶対に必要だという信念だったり、あるいは作業をこなせばこなすほど結果が来るというゲーム性に快感を見出せる性癖（そしてそれを可能にする実力）

を実は持っていたりします。いわば内発的動機づけがあり、その上、お金も伴っているという状態です。やる気が出るのも当たり前だと思います。逆を言えば、内発的動機を見出せない人は、おそらく長続きしないでしょう。

熱夢集

熱夢集 とは熱中、夢中、集中の総称です。

忘迷怠は確実性（確実にこなすこと）、調動脈は納得性（納得感を増やすこと）にフォーカスしますが、熱夢集は継続性（続けること）にフォーカスします。

熱中とは熱意や興味が冷めないようにすることです。夢中とは熱中や集中を探したり育てたりすることです。集中とは取り組みを開始し軌道に乗せることです。一言で言えば、（夢中と集中の言及順を入れ替えますが）熱中はエンジン、集中は稼働、夢中はエンジンや稼働のメンテナンスと洗練です。夢中についてはこじつけ感がありますが、エンジンや稼働を扱うことに夢中になっていると捉えてください（※1）。

要はエンジンとその動かし方の両輪で、自らを動かそうというわけです。いくら忘迷怠にてタスクを捉え、調動脈にて納得感を醸成できたとしても、実際にタスクをこなせなければ意味がありません。そして、こなせるかどうかは「こなせるまで続けるか否か」にかかっています。さらに言えば、続けるのにしてもだらだらやって100分で終わらせるよりも10分でさっさと終わらせた方が効率が良いです。集中力という概念は、あえて説明するまでもなく知られていますよね。

- ※
  - 1 熱夢集という言葉、特に夢中の部分はもうちょっとわかりやすくしたかったのですが、これが限界でした……。こじつけ感があると書いてますが、正直言うと完全にこじつけです。

パフォーマンスの最大化

熱夢集によって高められるものは **パフォーマンス** です。ここでパフォーマンスとは性能という意味であって、業績や成績といった生産性は指していません。**パフォーマンスが高いとは、自分が持っている性能を高く引き出せる** ということです。必ずしも高い生産性に結びつくとは限りません（※1）。

パフォーマンスを高めるには、熱夢集を高めればいい。では熱夢集の各々はどうやれば高まるでしょうか。

- ※
  - 1 一般的にパフォーマンスと生産性は比例します。特にものづくりやナレッジワークは、その人に合ったやり方と集中の仕方が必要になるため「まとまった時間」と「高い裁量」が求められます。ここがわからずやり方を制限したり、細かく管理したり、高頻度に会

> 議を入れたり声を掛けたりするのはパフォーマンスを削ぐ妨害行為です。一般的に、と書いたのは、たとえば「一見すると生産性は高そうだがアラが多い」とか「考慮すべき前提が漏れてるのでせっかくつくってもらったところ悪いけどやり直し」といったことがあるからです。そういうわけで、このような仕事では当人に全面的に任せる → 出てきた成果物をレビューするという二段階のやり方が取られます。もっとわかりやすいのは作家や漫画家でしょう。

熱中

熱中については、まさに前述した調動脈が役立ちます。調子を整え、動機を引き出し（やる気を引き出すための行動をし）、文脈も理解して納得できれば、熱意や興味は維持できるでしょう。特にスポーツがわかりやすいですが、本番で出し切るために何日何週間何ヶ月も前から準備に手間暇をかけることもあります。また、ずっと気張っていると消耗してしまい最悪病んでしまいますので、オンオフの切り替えや気分転換も大事とされています。

集中

次に（またも順序を変えて、先に集中から述べますが）集中については、自分の癖を把握して飼いならすことが求められます。一つ例を挙げると、筆者は誰にも何にも全く邪魔されない静音環境だと集中しやすいです。一方で、テレビの音はあってほしいとか音楽を聴きながらやりたいという人もいるでしょうし、ペットの猫などアトランダムな刺激や癒やしを好む人もいます。また同じ人であっても状況によってやり方は変わります。新しいアイデアを考えるときは何も聴かないが、単調作業をするときは何か聴かないと耐えられないなどです。

別の言い方をすると、**集中は制約のかけ方をいかに工夫するかにかかっています**。最近わかりやすい動画を見かけたので紹介します。ケンガンアシュラ作画のだろめおん先生の密着動画 なのですが、作画作業に集中するためにポモドーロ・テクニック、発声、おにぎりを食べる、と色々駆使しています。これは単調寄りの作業の例ですね。単調作業は退屈であり、意識の大部分を持て余してしまうわけですが、この意識の暴れ馬をいかに抑えるかが肝心となるわけです。筆者の持論ですが、プロと呼ばれる人はパラレルかつスピーディーだと思っています。常人では非常に時間かかること、あるいは決してこなせないことを短時間でこなせますし、複数の作業も並行（パラレル）して進行できます。それほどの実力が伴っているのです。だからこそ仕事における単調作業の割合が必然的に増え、意識の暴れ馬の制御が問題となります――と、これは単調寄りの話ですが、難解な仕事の場合も発想は同じですが、工夫の難易度が跳ね上がります。

夢中

最後に夢中については、熱中や集中のやり方を積極的に模索すれば良いです（そもそも夢中の定義はエンジンや稼働のメンテナンスと洗練でした）。

仕事術やタスク管理について調べたり、上述の動画もそうですが、他人（特にプロ）のやり方を

参考にしたり、あるいは同僚や友人とそういう話をしてみても良いでしょう。自分で試行錯誤してもいいですが、慣れないうちは難しいと思います。先にどんなやり方や事例があるかをインプットした方が捗ります。

どちらかと言えば、模索するための時間をちゃんと取れるかにかかっています。たとえば「1日30分の時間を確保できるか」といわれると、どうでしょうか。特に忙しい方には難しいでしょうが、このような投資ができなければ話になりません。

浅く管理するか、深く管理するか

タスクをどこまで管理するか、は人によって大きく違ってきます。

ざっくり言えば、浅く管理するか深く管理するかに二分できます。

- 浅いタスク管理
  - 自分を取り巻いている多数のタスクを管理します
  - 目的は「理想的な消化に近づくこと」であり、忘迷怠したい人に適しています
  - 忘迷怠の項で述べたように動線は重要です、また複数のツールを使い分けることも一般的です
- 深いタスク管理
  - 少数の重要なタスクのみ管理します
  - 目的は「打開と前進」であり、忙しい人に適しています
  - 結局は決めの問題であり、意思決定する（特に捨てる）ための要領を日頃から磨いておく必要があります

浅いタスク管理

浅いタスク管理 とは、自分を取り巻く多数のタスクを管理するスタンスです。仕事の本業と雑務、日常生活の家事、日課や習慣、趣味、勉強やトレーニングといった投資――など生活はタスクに溢れていますが、これらをなるべく捉えて管理します。

目的は前述した忘迷怠です。多数のタスクを何も意識せず適当にこなしているだけでは何かと非効率なのです、忘れたり迷ったり怠けたりします。逆にそれらを全部捉えて、管理できれば、効率的にこなせます。あるいは自分のペースやリズムに則った形で割り当てていけば生活自体が快適にもなります（いわゆる生活の質が向上する）。さらに言えば、やり忘れのタスクがなくなることで、「タスクをやり忘れたことによる後々の被害の拡大」も防げます。

タスクの総量自体は変わりません。たとえば今日やるべきタスクが 20 個あって、6 時間かかるとしても、管理しようがしまいが 20 個や 6 時間は変わりません。それでも管理しなければ忘れて迷って怠けて 9 時間かかったり、9 時間かけても終わらないかもしれません。逆に管理ができ

て、かつ適切に実行できたら、6 時間で 20 個という理論値に近づけるのです。そういう意味で、**浅いタスク管理とはタスク消化の理論値に近づくためのもの** です。

やり方は色々ありますが、一般的に複数のやり方を使い分けます。たとえば予定にはカレンダーを使うでしょうし、忘れたくない用事にはリマインダーやアラームを設定するでしょう。起床時の目覚ましやカップラーメンの 3 分タイマーは有名です。ゴミ捨てなど定期性のある雑務については、タスク管理ツール側の設定を定期にしたり、あるいは単に時間割のような表をどこかに掲示してそれを見たりするでしょう。

もう一つ、大胆なツールとしてタスクシュートがあります。これはタスクリストに書かれたとおりに行動することを強力に促すツールで、いわば毎日何するかを全部リスト化する（できるように育てていく）ものです。人が一日に抱えるタスクは 20〜60 と筆者は思っていますが、そのすべてを投入して管理することさえ可能です。ちなみにタスクシュートも（忘迷怠の部分で前述しましたが）動線になりますし、動線として使うことが想定されていると思います。

深いタスク管理

深いタスク管理 とは、少数の重要なタスクを管理するスタンスです。たとえば「今日はここに書いた 7 個のタスクだけをやるぞ！」といったやり方は、その 7 個という少数のみにフォーカスしています。仕事上、あるいは生活上はもっと多数のタスクがあるはずですが、あえて 7 個だけを見るというわけです。

浅いタスク管理は大胆な取捨選択と言えます。本質にのみ目を向けるため、子細は捨てねばなりません。前章にて解釈の公理（意思決定の必要性）を述べましたが、まさに意思決定が求められます。いわゆる自己啓発や仕事術の文脈でも、こちらが想定されていることが多いです。アイビー・リー・メソッド は毎日 6 個ですし、時間管理マトリクス は重要と緊急の観点で大胆に優先順位を下します。

この性質上、忙しい人に適しています。忙しい人は 1 日 24 時間では到底足りないレベルのタスクを抱えていることが多く、ただでさえ状況に流されがちですが、そんな中だからこそ、今はこれをやるのだと決めることに価値があるわけですね。もっとも、決めただけで実際にできるとは限らず、むしろできないことの方が多いです。深く管理すると決めたところで、現実の忙しさは何も変わらないわけですからね。諦めること、捨てること、協力してもらうことからやり方や考え方を変えることまで多方面かつ継続的な努力が求められます。特に捨てる勇気（または委譲する要領や力）が要ります。

このように厳しい取捨選択をするのは、**深いタスク管理の目的が打開と前進だから** です。タスクが多すぎてどうしようもないときに、それでもまずはこれをやると決めて、実際にやるのです。口で言うのはかんたんですが、忙しいと余裕がありませんし、「うるせえ全部大事なんだよ」「お前に何が分かる」とあたりたくもなります。

だからといって当てずっぽうで選べばいいというものでもありません。何を選べばいいかは、前述し

た調動脈にかかっています。早い話、日頃から調べたり考えたりしておかないと、いざという時に役に立ちません。「わかったわかった、今から 20 個から 3 個に絞るから 2 日待ってね」は、大抵の場合許されないでしょう。「うん、じゃあこの 3 個にしようか」とすぐに判断できるだけの調動脈を日頃から備えておく必要があるのです。

ツールはアナログか、デジタルか

タスク管理ツールとして何を使うか、についても大きく分けて二つあります。

アナログとデジタルです。

- アナログ
  - デジタル機器を使わないツール
  - メリットは、親しみやすく始めやすいことと、体験が豊かなこと（手書きの効用や電子書籍に対する紙の書籍と同じ）
  - デメリットは、素早い操作が行えないことや空間に限りがあること、総じてある程度頭の中で処理する性能が求められること
- デジタル
  - デジタル機器を使ったツール
  - メリットは、操作の素早さとデータ同期で、大量のタスクを扱えるしどこからでもアクセスできる
  - デメリットは、スキルが必要なことやツールごとに勉強が必要なこと、そして目移りしやすい（手段の目的化）こと

アナログツール

アナログツールとはデジタル機器を使わないツール（道具）を指します。

例を挙げると紙に書く、手帳に書く、付箋に書いて貼り付ける、冷蔵庫に貼り付けた落書きボードに書く、ホワイトボードに書く、壁一面をホワイトボードにしてそこに書く（ホワイトボードウォール）、チョコの包装に書くなどがあります。他にもたくさんありますし、紙に書く一つを取り上げても、どんな紙で、どんな形式で、どんなペンで書くかといったバリエーションも色々とあります。そもそもタスクを言葉で「書く」だけでなく、図表にしたり、マインドマップのように線で繋いだり、あるいは単に気持ちを上げるために絵を入れたりなど「描く」こともあるでしょう。

アナログツールのメリットは親しみやすさと豊かな体験です。まずデジタルの知識やスキルが要らないので始めやすく、自分なりのカスタマイズもしやすいです。実は手先の器用さ、視覚的記憶力、空間認識能力といった適性があり、向いてない人もいるのですが、少なくとも（タスク管理自体を除けば）特別な知識やスキルは要りません。もう一つ、無視できないのが読み書きする時、特に書くときの情報量です。よくキーボードやタッチパネルよりも手書きの方が覚えやすい

といいますが、手書きという行為の方が刺激（情報）が多く、神経も使い、あまりスピードも出せないのでしっかりと向き合うことにもなり、と記憶しやすいからです。物書きの世界では写経（丸写しする）という鍛錬方法もあります。電子書籍があっても紙の本はまだ根強い人気がありますが、手書きも同様です。いわば体験として豊かとでも言えるでしょう。これが意外と重要で、退屈になりがちなタスク管理に彩りを与えます。別の言い方をすると **気持ちが上がります**。タスク管理はまず続けることが大事ですから、気持ちを上げて折れないようにできることは大きなメリットです。

デメリットもあります。間違ったときに素早く修正できなかったり、そもそも書くスピードがあまり出なかったりするため、多くのタスクを扱えません（さらに言えば紙面にも限界があります）。**ある程度頭の中で処理してから、それを外に出す** という営みが必要になります。もっと言うと、頭の中で処理するのが苦手な人にはアナログは向いていません。それでもないよりはマシです。もう一つ、紛失や覗き見に弱いことも挙げられます。デジタルだとクラウドに保存できますし、接続元の PC やスマホをセキュリティーで保護すれば他人には読まれませんが、アナログだと物理的になくしてしまいますし、見られてもしまいます。丁重に管理すればいいだけですが、神経を使います。個人タスク管理には誰にも見せないプライベートな情報も入れますから、このセキュリティーの低さは意外としこりになります。

総じて、アナログツールは **頭が強い人には便利なツール** と言えるでしょう。基本的に頭で処理できるが、ちょっと外に出しておきたい、といったように補助としてタスク管理を使いたい人に適しています。物理的な道具ゆえの限界はありますが、電子書籍に対する紙の書籍と同様、気持ちが上がる効果も見逃せません。

デジタルツール

デジタルツールとはデジタル機器を使ったツールを指します。

例を挙げるとスマホアプリ、PC 用のデスクトップツール、コマンド、テキストファイルやエディタ、ブラウザから使う Web サービス、スマホでも PC でもないデジタルデバイスなどがあります。直接的にタスク管理を謳ったツールが多いですが、工夫次第では、汎用的なツールをタスク管理用途に仕立て上げることもできます。たとえば task.txt をつくって自分なりのルールで運用すればそれも一種のツールですし、まさにそういうテキストファイルベースなツールもあります（todo.txt）し、プレーンテキストで管理するのが最強という人もいます。プレーンテキストは専門的なので後の章で詳しく扱います。

デジタルツールのメリットは素早さとデータ同期です。まずタイピングやフリック入力、もちろんコピペや複製も使えますし、修正や並び替えや挿入や検索もかんたんで、タグをつけたりそれでフィルタリングできる――と素早く操作できるのが強いです。向こう一年間のタスクを全部投入して管理する、なんてこともできてしまいます。ツールやスキルにもよりますが、何百何千というタスクも管理可能です。加えて、Web サービス型のツールですと、データの同期もできるので、どこからでもアクセスできます。普段 PC からアクセスして、外出先でスマホからアクセスして、なんてこともできます。特にやりたいことが多い人に有益で、要はやりたいことをタスクとして全部ぶっこん

でおき、自宅なり外出先なりから眺めて、今できることをやったり絞り込んだりするといった「ちょっとした作業」がしやすいのです。デジタルだからこその恩恵でしょう。

もちろんデメリットもあります。まずはスキルです。スマホにせよ PC にせよ一定のリテラシーは必要ですし、ツールごとに作法や機能も違うので、**基本的にツールごとに勉強が必要です**。要領がいい人や横着な人は、とりあえず使ってみることで何とかしようと考えますが、デジタルツールは意外と融通が利かないしですしわかりづらい。結局ドキュメントを読んで勉強するか、実験的な試行錯誤をすることになります。そもそも手に馴染むまで時間がかかるので、多少は辛抱強く使わないといけません。前述したトレメントですね。このあたりができないと、すぐ次のツールに乗り換えようとしてしまいます。何しても続かないダメ人間、というと言い過ぎですが、そんなスパイラルに陥りがちです。まさにこの現象もデメリットの一つで、**手段が目的化しやすい** のです。色んなツールを試したり遊んだりすることが目的になってしまう。厄介なことに、タスク管理ツール側も利用者の気を引くためにそれを助長してきます。魅力的な宣伝文句、クールで使いたくなるインターフェース、絶賛しているインタビュアーやレビュアーの声――抗うのも大変です。

総じて、デジタルツールは **操作の早さやアクセス性の高さを求める人には便利なツール** と言えるでしょう。アナログツールは頭で処理したものを出すのがメインでしたが、デジタルだと（ツール次第ですが）その必要もありません。かわりに素早く操作し、投入したタスクもツールに則って管理できるだけのスキルは要求されます。また、選択肢が多いので、目移りを許すと手段の目的化に陥ってしまいます。

様々な戦略とスタンス

戦略とはどんなやり方や考え方で望むかという方針のことです。スタンスとは「こういうときはこうする」という反応や行動のことです。タスク管理の戦略やスタンスは色々ありますが、主なものを捉えてみました。詳細は後の章に譲るとして、ここでは概略を眺めていきます。

- 戦略
  - 時間や話題などの「枠」で区切って、各枠を尊重する
  - 整頓せず一箇所にまとめておく
  - 日課や習慣など、繰り返しの力でまわす
  - ツールの活用は最小限とし、基本的に自分の頭に頼る
- スタンス
  - タスクを抱える量を減らす系
  - タスクの処理にかかる手間を減らす系
  - タスクの外の環境に働きかける系

戦略

「何を拠り所にするか」で分けることができます。

時間で区切る

最も有名なのはカレンダーを用いて予定ベースで生きることです。予定に従って行動すればいいだけなので比較的かんたんですし、特にビジネスの管理職層はこれに頼っている人も多いのではないでしょうか。

次によく知られているのが「時間帯を意識すること」でしょう。午前は何をする、昼食前後は何をする、午後1と午後2、夕方、通勤、夜、就寝前――時間帯の分け方は様々ですし、単に1時間とか1.5時間といった枠（授業や研修がまさにそうですよね）をつくってそこにタスクを差し込むやり方もよく知られています。

あとは、前述したタスクシュートもそうですが、今日とそれ以外、と日単位で分けて今日にフォーカスする潮流もあります。今日という区切りをつくって、今日やるべきことを決めて、全部終えたら今日はもうおしまいとするわけです。マニャーナの法則 も同じアプローチで、今日やることを決めた後に新たなタスクが降ってきてもそれは明日に回す、という先送りをしますが、これも今日と明日以降に二分しているからこそできる発想です。

話題で区切る

1-話題 1ページで扱えるシステムがあり、これを使うと 1-タスク 1-ページで情報をつくることができ、タスク管理を行えます。たとえば「何かしら運動を始める」「書籍 XXXX を読む」「引っ越しの検討をする」「冷蔵庫を買い替える」など、思いついたタスクをつくります。この場合は 4 ページ分のタスクが生成されます。各タスクページでは自由にコメントを書き込めるので、進捗や情報があれば追記できます。ツール次第ですが、終わったらクローズして見えなくしたり、締切や優先度を設定して強調したりもできます。

この戦略はかなりニッチではありますが、この手のシステムを知っているエンジニアの方には支持されやすいです。ツールとしては GitHub Issuesが知られています。汎用的なノートツールや原始的なファイルでもできなくはないですが、細かい操作が面倒くさいので、よほどの甲斐性か、あるいは自分で仕組みを自動化できる人でないと長続きしません。エンジニア向けと言えるでしょう。

一箇所にまとめる

とにもかくにも必要な情報をすべて一箇所にぶちこむ、という戦略です。

有名なのはガレージや書斎など仕事場、デスク（机）、PC のデスクトップ、スマホのホーム画面です。散らかっている人も多いのではないでしょうか。何も意識しなければ、基本的にこのスタイルになりがちですが、それで上手く行っているのなら無理して別のやり方に変える必要もありません。というより、この一箇所ぶちこみ作戦も立派な戦略の一つなのです。下手に管理するよ

りも圧倒的に楽ですし、結局見つかればそれでいいのですから。ただし、見つけるためには相当頭が良くないといけません。天才と呼ばれる人達は整頓が下手なイメージがありますが、頭が良くて引っ張り出せるから整頓の必要がないのです。

要は整頓せずに一箇所にまとめておくという原始的な戦略です。タスク管理に当てはめると、タスクを言語化して放り込むというよりは、タスクを表す何らかのもの（書類だったりファイルだったり物品だったり）が散らかっているイメージです。役所への申請が必要な封筒や通知があった場合、「～～の申請を行う」的なタスクをツールに追加するのではなく、その封筒をデスクに置いたり、通知の内容（メール本文やウェブサイトのURL）をファイル化してデスクトップに置いたりします。一方、note.txt のような「唯一の巨大なノート」を用意して、全部そこに書くというスタイルもあります。

繰り返しの力でまわす

習慣や日課、毎日少しずつ着手するといったことを重宝する戦略です。

習慣トラッカーのような記録ツールでモニタリングしたり、タスクシュートのような「定期的なタスクを扱いやすいツール」に頼ったりします。この戦略を採用すると、毎日「今日はこれとこれとこれとこれとこれをやる」といった形でタスクがずらりと並ぶイメージです。

少しわかりづらいので一つ例を挙げると、「XXXX という本を読みたい」といった場合は「10分読む」という日課を毎日差し込んだりします。毎日が嫌なら2日に1回とかでも構いませんし、もっと読めるなら1日30分でもいいでしょう。もちろん朝にやるのか、夜にやるのか、毎日タイミングを固定するのかそれとも適当に空いたときにやるのか、なども含めて自由です。細かい部分は人次第ですが、ともかく定期的に続けていくという発想で捉えます。

見るからに面倒くさそうですし、難しい戦略でもあります。どのタスクをいつ、どれくらいの頻度で行うかはきちんと管理しなければいけません（これを手作業で行うのは厳しいため専用ツールに頼ります）。忙しい人というよりは、やりたいことや備えたいことがたくさんある人に向いていますが、そう単純でもありません。ツールが提示してくれたタスク達と、それを実際にそのとおりにこなせるかどうかには大きな隔たりがあって、やる気や集中の面で苦労しがちです。

自分の脳内が正義

基本的に自分の頭だけで事足りるが、さすがにタスク全部は覚えていられないので必要最小限の手間で思い出したい、という人もいます。このような人は、タスク（の元となるやり取りや物）を保存しておいて、あとで見返します。たとえばブラウザやチャットツールや SNS のブックマーク機能を使ってチェックをつけておく、などです。

情報収集の文脈でいう「あとで読むタグ」や「あとで読むカテゴリー」は、まさにこの戦略です。メールを仕分けしたり重要タグをつけたりするのもこれですが、細かく分類するというよりは「何もつけない or 重要マークをつける」など二値にのみ頼るのが本質です。要はつけるか、つけないかです。つけたものは重要なものとみなして、あとで読み返して消化します。この程度の手間な

らまあ負ってもいいだろう、というわけです。

この戦略も本人の頭の良さが求められます。そもそもこんな単純なやり方だけでは上手くいかないからタスク管理に苦労するわけです。とはいえ、誰がどんな性能を持っているかは本人さえもわかりませんから、試してみるのはアリです。この戦略で済むならラッキー、済まなくて当たり前といったところでしょうか。

ちなみに、この戦略は「作業的なタスク」との相性が良いです。作業的なタスクとは、時間さえかければ終了に導ける類の「作業」とも呼べるタスクです。面倒くさいかもしれないが、かんたんなタスクというわけですね。逆に限界もあって、いくら作業的なタスクであっても物量が多すぎると歯が立ちません。また作業的でないタスク、たとえば正解も事例もない創造的な仕事や、中長期的な投資が求められる研究的な仕事などはお手上げです。

スタンス

いくつかの観点で分けることができます。

抱える量を減らす

そもそもタスクを抱える量を減らしていこう、と考えます。

一番わかりやすいのは「人に任せる」ことでしょう。フォローや管理は必要になるかもしれませんが、何でもかんでも自分で行うよりも前進できることがあります。そもそもある程度の規模になると個人だけではまわらなくなりますので、どこかの段階で多かれ少なかれこの発想は必須です。別に経営者やマネージャーの特権ではなく、身近なところで言えば、お金を使って家事代行サービスを頼む等もこれにあたります。言われてみると単純ですが、人はタスクを自分で抱えたがる（※1）ので、意外とこの発想は忘れがちです。

他によく知られているのは「抱えない」ことでしょう。メールやチャットが来たら即返信するとか、会議や相談時は先送りにせずその場で決断を下すといったことです。ボールというたとえが使われるのは有名だと思います。また、ボールとは違いますが「面倒くさい雑務はすぐに終わらせる」「2分以内で済むことはすぐ終わらせる」といったアドバイスも有名かもしれません。

もう一つ挙げると、戦略の項でもチラッと出ましたが「先送りする」ことです。特に今日はもうやりません、明日やります、と今日の領分が侵さないようにすることです。そうでなくとも人気店で品切れが発生したときに並び待ちはここまで、と締め切ったりしますし、昨今、Web サービスや通信回線なども使いすぎると制限が入ったりしますよね。先送りというとピンと来ないかもしれませんが、無理やりぶった切っているわけです。ぶった切ってしまえば、それ以上抱えることはありません。言葉にすると単純ではありますが、ぶった切るための判断基準と、何よりぶった切るのだという意思決定が難しいです。

- ※

- 1 人がタスクを自分で抱えたがる理由は秘匿性と排他性です。秘匿性については、要するにプライバシーです。タスクには個人的な情報や背景が絡んでいることがあり、無闇に人に渡せるとは限りません。部屋の掃除をお願いしたいにしても、まずは見られては困る物の対処が要りますよね。こういうのが面倒くさいので、「じゃあ自分でやってしまうか」となります。もう一つ、排他性については、要するに「部外者は黙っとれ」です。人は自分なりのこだわりやプライドを持っているものですが、これはタスクにも反映されています。タスクを無闇に人に任せると、そこが侵害されてしまうおそれがあります。赤の他人に部屋の掃除を任せたときに「汚いですね」とか「もう少し綺麗にした方がいいですよ、参考までに――」などと言われたら苛つくでしょう。侵害の可能性があると、「いや、まあ、自分がやればいいか」と逃げの心理になります。逆を言えば、この二つを軽減できれば、どんどん任せることができます。

省力化する

タスク管理に要する手間――特に操作や連携や選別を省力化していこう、と考えます。

操作や連携については、自動化機能（Automation）で省力化できます。近年のプロジェクトタスク管理ツールはこれをサポートするようになりました。たとえば Asana、ClickUp、monday.com（のWork Management）、Trello など。その本質は「もし XXX が起きたら、YYY を実行する」というもので、XXX をトリガーとして YYY を自動実行します。たとえば「デザインが必要」タグをつけたらデザイナーの A さんを担当者に割り当てるとか、締切が近づいたら担当者にメンションをつけたコメントを送信するなど、工夫次第で色々できます。

また、タスク情報をチャットツールなど他のツールに流す機能を備えたものもあり、Todoist など個人タスク管理ツールではこちらが多いかもしれません。Zapier や IFTTT といったワークフローツール（※1）との連携も想定されていたりします。そういう意味で、個人タスク管理のシチュエーションでは、省力化≒ワークフロー連携によりタスク管理を支援する、と言えそうです。一見すると省力化の効果にはピンと来ませんが、トリガー実行のシチュエーションは（個人レベルでも）意外と多いので、上手く落とし込めれば効果は期待できます。特に手作業が多い人は、1日30分以上の削減も可能かもしれません。

選別の観点としては、許可的と拒否的があります。

許可的とは「あらかじめ決めたもの以外は全部捨てる」というもので、前述した深いタスク管理が当てはまります。これをするんだ、と決めた少数のタスク以外は全部（いったんは）捨てるのです。すでに述べたとおり、たしかな判断基準と意志決定力が求められます。

拒否的とは「指定条件を満たしたものは問答無用で捨てる」というもので、前述した「時間で区切る」戦略がわかりやすいです。マニャーナの法則は今日降ってきたタスクは全部明日にまわす（今日やることを捨てる）のでしたし、他にも タイムボックスという時間枠を設けてその範囲でのみ対応する（つまり時間枠の外でタスクをこなすことを一切捨てる）やり方もあります。営業時

間の概念はまさにそうですよね。働く時間をビシッと定めてしまうことでメリハリをつけます。

いずれにせよ、思い切って決めてしまうことで余計な判断の手間をなくしています。判断の省力化と言えます。

- ※
  - 1 ワークフローとは「仕事の一連の流れ」を意味する言葉です。特に情報を誰に渡すかとか、どのツールに流すかといった流れを指します。プロジェクトタスク管理で言えば、たとえばチャットツール、タスク管理ツール、スケジューラー、管理者によるレビューと承認という要素があったなら、チャット上で仕事のやり取りが発生 → 正式なタスクはタスク管理ツールに登録する → レビューが必要な事項はタスクにその旨を設定後、当該管理者の予定を押さえる、のようなフロー（流れ）が想定されます。また個人で、たとえば記事や SNS の投稿を生業としている場合も、情報収集ツールで情報を集める → ネタになるものをブックマークで溜める → 書けそうなものを下書きツールで書いていく → 清書は清書用ツールで仕上げる → 投稿先にコピペして投稿、のようなフローがあるでしょう。通常、仕事ではこのようなワークフローは自ずと構築されていきます。ワークフローツールは、このようなフロー（の各々で必要な作業）を自動化しようというものです。いちいち手作業でやるのは面倒ですし、人間なのでヌケモレもありますが、ワークフローを定義してツールの力で自動で流せるようになると安定的に素早くこなせます。

盤外戦をする

盤外戦を重視しよう、と考えます。

盤外戦とは、タスク管理そのものよりも、より高次の視点での対処を行うことです。たとえばタスクの発生源そのものに働きかけたり、タスクを遂行する自分自身の心技体を変えたり改善したりといったことです。タスク管理という盤上の戦いではなく、その外の戦いにこそ目を向けようというわけです。詳しくは後の章で扱うとして、ここでは盤外戦を重視するスタンスをいくつか紹介するに留めます。

あそびや余裕を持つことや、それらを侵食させないことは盤外戦の一つでしょう。仕事と生活の配分はよく議論になりますが、ワークライフバランスという言葉もすっかり久しく、今はワークインライフと言ったりします。これはワーク一辺倒ではなくライフを行う余裕も持とうとも言えるでしょう。ちょうど一つ前の項で述べた「タイムボックス」も、ある意味これです。仕事する時間枠を強制的に決めることで、それ以上生活という余裕が侵食されるのを防いでいます。

また調動脈の脈、文脈を重視することも盤外戦のセオリーです。タスクにバカ正直に向き合うのではなく、その文脈――背景や前提や真意などを探って、ショートカットしたり根本から解消したりするためのヒントを探します。ただし、こちらは泥臭く向き合うことにも等しいため、手間暇がかかりがちです。特に文脈は誰かの脳内にのみ存在するケースも多く、その人から引き出すためには信頼関係の構築も必要で、なおさら時間がかかります。人間関係的な気力も消耗します。このように、余裕を持つこととは相反する傾向があります。一方で、最初に苦労して文脈をしっかりと把握しておけば、後々首が締まらずに済みます。文脈を考慮した必要最小限の行

動で済みますし、何なら先回りしてタスクの発生を抑えることもできるからです。直感に背くかもしれませんが、長期的に見ると余裕の担保にも繋がるのです。このスタイルが如実に求められるのがいわゆる新事業の開発で、顧客の隠れた文脈そしてそこから見えてくる真のニーズを探るための慌ただしく動きます。これができないと「一生答えにたどり着かない」もありえます。

もう一つだけ紹介すると、タスクを一気に処理するという考え方もあります。たとえば普段地方に住んでいる人が、久しぶりに1週間の東京旅行に行く場合、東京で行える用事をこの1週間でこなしたいと考えますよね。そのための準備は惜しまないはずです。このような考え方をタスク全般に適用するイメージで、「まとめて処理する」タイミングを常にうかがう形になります。一見すると面倒くさいことをしているように見えますが、たくさんの情報や制約から上手く準備する作業は楽しい（旅行計画が楽しいのと一緒です）ですし、この準備作業を通じて文脈の理解もできたりしますので、意外と理にかなっています。軍師というと大げさですが、そのような営みが好きな人には向いています。

まとめ

本章では個人タスク管理をいくつかの観点で眺めました。各観点ごとのまとめは各節冒頭を参照してください。

- タスク管理する人としない人
  - 特性や状況に依存する
  - トレメントに耐えられる人はする・できる
  - ゼロイチではなくグラデーションがある。自分なりのバランスを模索すれば良い
- 忘迷怠、調動脈、熱夢集
  - 忘れる迷う怠けるを減らして、確実にこなすか
  - 調子と動機と文脈を踏まえて、納得感を持って望むか
  - 熱中と集中と夢中を制御して、快適に取り組むか
- 浅く管理するか、深く管理する
  - 最小の手間で済ませたいなら浅く。色んな手段とツールを駆使して全部管理する。几帳面さが必要
  - 前に進みたいなら深く。多数を捨てて少数にフォーカスする。捨てる意思決定を素早くこなす地力が必要
- ツールはアナログか、デジタルか
  - アナログには頭の良さ（処理性能）が必要
  - デジタルにはスキルと勉強が必要
- 戦略（どんなやり方や考え方で管理するか）やスタンス（こういうときはこう対処する的な姿勢）も色々とある

=== Chapter-4 プロジェクトタスク管理 ===

本書は個人タスク管理を扱った本ですが、残る二つ――プロジェクトタスク管理とパートナータスク管理についてかんたんに扱っておきます。

本章ではプロジェクトタスク管理の本質を端的に整理します。PMBOK のような体系や各種プロジェクト管理ツールの解説、あるいは即効性のあるテクニックの紹介等はしません。あくまでも個人タスク管理を体系化しようとする筆者の目線で整理したものです。読者なりに気付きやヒントを得てもらうことを期待します。

プロジェクトタスク管理の 3A

プロジェクトタスク管理の本質は 3A で表せると思います。

英語	キーワード	解説
Assign	アサイン	人にタスクを割り当てること
Adjust	調整	タスクの属性値を変えること
Alternate	全体と詳細	タスク全体の俯瞰と特定タスクの注視を行き来すること

Assign/アサイン

人にタスクを割り当てることです。

プロジェクトタスク管理には複数人のメンバーが存在するため、**誰にどのタスクを割り当てるかという営みが必ず発生します**。割り当てとは約束であり、タスクTをAさんに割り当てるとは、Aさんに「あなたはタスクTを責任を持ってちゃんと遂行してね」と課していることになります。なぜそんなことをするかというと、そうしないと進まないからですね。人は忘迷怠する生き物ですし、複数人ですと遠慮や手抜きも発生します（たとえば心理学的には社会的手抜きと呼ばれる現象が起きます）。意図的に管理をしなければままならないのです。

一方で、近年ではマネジメントレスの潮流も見られます。「マネージャーは要らない」「中間管理職を廃止しました」のような取り組みを聞いたことがある人もいるでしょう。要はタスクをこなせさえすればいいので、誰に管理されずとも自律的に行動できるならそれでいいわけです。といっても言葉で言うほどかんたんではなく、アサインという本質はすでに会社のルールやプロセス、仕事そのもの（たとえば管理職はまさに管理でご飯を食べています）、ときには契約にさえ入り込んでいます。そもそも大部分の人は労働者として管理されることに慣れきっていますし、自律的に

進めるほど仕事にモチベーションもありません。まだまだマイノリティのやり方と言えるでしょうし、ここ数十年はマジョリティになることもないと思います。

マネジメントレスであってもタスク管理ツールは使います。後述の Alternate にも絡みますが、状況を知ってもらうためには結局ツールへの入力が必要だからです。そうしないと、いちいち本人に聞かないといけなくなります。仕事に集中してるのに聞かれたら集中が削がれます（割り込みといいます）し、定期的にヒアリングの場を設けるのはそれこそ管理的ですよね。これは見過ごされがちなことですが、**打ち合わせもタスクの一つ** であり、打ち合わせをしている時点でアサインが発生しています。打ち合わせの多いマネジメントレスは、なんちゃってマネジメントレスでしかありません。

現在ではそこ（ツールへの入力）さえも放棄した潮流も出ています。ティール組織と呼ばれる組織のあり方です。ティール組織では無駄な制度やプロセスが省かれ、従業員の裁量が強く、心理的安全性も高くて、超高頻度なコミュニケーションを行えることが特徴です（と明言はされていませんが筆者はそう捉えています）。「何かあれば聞けばいい」とのメンタルモデルを持つ人は多いと思いますが、色んな制約があったりビジーだったりで中々そうもいかないでしょう。ティールならそれができるのです、というよりそうするための組織としてデザインされています。事例こそ色々ありますが、本質は驚くほど似通っていると感じます。というと難しそうですが、要は創業者数人で立ち上げたばかりのベンチャー企業のようなものです。事業のことだけに専念できて、変な制約や邪魔もなくて、気兼ねなくいつでも相談して、みたいなあの感じです。難しいのは、その感じをスケールさせる――組織規模が大きくなっても維持できるようにするところです。

Adjust/調整

ここでは **調整とはタスクの属性値を変えること** です。

属性とはタスクが持つ特徴のことで、たとえば以下があります。

- 名前
- 状態（未着手/着手中/完了/中止など）
- 担当者
- 締切
- 開始日
- 優先度（高/中/低、1-5の5段階など）
- 費やした時間（工数と呼ばれることが多い）
- コメント

システマチックな言い方になりますが、プロジェクトタスク管理とはタスクの属性値を変える営みともいえます。以下にタスクTの属性値を変えていく例を示します。例では一タスクですが、実際は多数あるいは無数に存在します。

経過	イベント	.	名前	担当者	状態	優先度	締切	コメント
1	タスクTを新規につくる	.	T					
2	Aさんを割り当てる	.	T	A				
3	着手した	.	T	A	着手中			
4	緊急ではないのでその旨	.	T	A	着手中			期限ないのでペンディング
5	状況変わって緊急に	.	T	A	着手中	高	2024/05/09	~~期限ないのでペンディング~~

このように状況に応じてタスクの属性値が変わっていく――と楽なのですが、タスクが勝手に変わることはありません。私達が能動的に変えるのです。ツールを使わない場合もありますが、その場合も頭の中で同じようなことをしています。しかし頭の中だと信用できませんし、引き出すのにも会話が要りますし、人によって何のタスクとどんな属性値を持っているかも違うので争いも起きがちです。調整の結果を記録することは必須であり、記録にはツールが要ります。つまりプロジェクトタスク管理ではツールが必須です。

ここで厄介なのが **調整が頻発する** ことです。プロジェクトにイレギュラーはつきものですし、単に慌ただしい場合でも調整の量は増えます。一度ツールにタスクを入れ終えたから、あとは属性を変えればいいだけだ――ともなりません。新しいタスクが増えたり、既存のタスクが消えたりといったことは平気で起こります。これらをすべてをツールに反映できなければ、誰かの頭の中だけに留まってしまうことになります。そうなると前述したとおり、引き出すための会話つまりは会議が必要になって、でも人によって抱えているものが違うから齟齬が起きて、と時間と労力がみるみる消費されていきます。プロジェクトは大変になりがちですが、その主因は単にツールに反映ができていないことによる秩序の崩壊です。

この「調整の頻発によるツール反映の漏れ」に対抗する手段は、今のところ二つです。一つは単純にツールへの反映を徹底されることですが、通常そこまでのツールやスキルは備わっていないため現実的ではありません。もう一つは反応的状況に溺れることです。**反応的状況（Reactive Situation）** とは、その時その時で起きたことに即座に反応する形で生き延びていくような状況を指します。慌ただしい世界は基本的にこれになります。反応的状況では調整（とツールへの反映）は追いつかず、ただただ目の前を見て瞬発的に反応し続けることしかできません。つまりタスク管理が通用しません。逆をいうと、タスク管理という面倒くさいことをしなくて済むとも言えます。それゆえ反応的状況はよく好まれます。プロジェクトが無駄に忙しくなりがち

なのもそのためです。

Alternate/俯瞰と注視

タスク全体の俯瞰と、特定タスクへの注視を行き来することです。

Alternate は「交互に行き来する」の意です（※1）。木と森を見る、ボトムアップとトップダウン、全部と細部など似た言葉はたくさんあります。改めて言われるまでもない、現代人ならおそらく誰もが触れる考え方だと思いますが、プロジェクトタスク管理にも登場します。

- ※
  - 1 交互に、とあるのでイーブン（50% 50%）で行き来しなきゃいけないのかと思うかもしれませんが、そんなことはありません。20 80でもいいですし、5 95でもいいです。最適なバランスは状況次第ですし、人次第でもあるでしょう。大事なのはとにかく全体と詳細の両方を見ることです。

俯瞰

仕事には多数、ひょっとすると無数のタスクが存在し、これらは有機的に絡み合っています。たいていは階層的に整理されがちですが、人間関係や地理的経路のようにネットワーク状になっていることもあります。そもそもタスクは最初からすべて見えているとは限らず、何らかの目的や目標があって、そこから分解されていくものです。つまり重要なのはタスクというパーツ各々というより、それらが織りなす全体です。目標には届くのか、あとどれくらいで届くのか。状況はどうなっているのか。ヤバそうなところはないか――そういった大局的な目線は、タスク単体よりも、タスク達を取り巻く関係全体を眺めないと見えてきません。

では **眺めるためにどうするかというと、何とかしてタスクを並べる** のです。リストにして並べたり（階層的に段々にして見せることも含む）、表に埋め込んだりすることもあれば、カンバンやボードといった形でカード状のものを配置したりします。個人の場合ですとデスクやデスクトップなどで適当に散らすこともできますが、プロジェクトタスク管理は複数人が見るものなので、並べ方は整然となるのが普通です。アナログな職場で我流に管理している場合や、医療現場におけるトリアージなど空間上の何か（この場合は治療対象という「人」）をタスクとみなす場合はそうでもないですが、あまり馴染みはないでしょう。ともかく、何とかしてタスクを並べて、俯瞰できるようにするわけです。

デジタルツールの場合、さらに俯瞰しやすくする仕組みがあります。特に「タスクが全部で 1000 個あります」となると眺めるだけでも一苦労ですので、絞り込む必要があります。ここで使うのがフィルタリングとソートです。**フィルタリング** は指定条件に当てはまるものだけを表示するもので、「優先度が高いタスクのみ表示」とか「完了したタスクは非表示」とかいったことができます。「指定したキーワードにヒットしたものを表示」など「検索」は特に有名です。**ソート** は並び替え操作のことで、指定した順に並び替えます。昇順や降順は有名で、名前や日付でソートした経験は割と誰でもあるのではないでしょうか。

この二つを組み合わせると、相当柔軟に絞り込むことができます。一方で、絞り込むためにはその観点をちゃんと理解し、かつタスク側にも入力しておく必要があり、現実的には形骸化することも多いです。たとえば優先度が高いタスクのみフィルタリングしたい場合、すべてのタスクの優先度属性をちゃんと記入しておく必要があります。優先度が高いのに記入が漏れていたタスクは表示されません。こういうヌケモレが多いと、フィルタリング結果は信用できなくて形骸化します。もちろん、最初は上手くいっていても、忙しくなってきたり、単に面倒になってきたり、あるいはサボる人が出てきて他の人もそこに影響されて次第にサボり始めて、など色んな原因でもかんたんに形骸化していきます。結局、メンバーを会議で集めてその場でヒアリング、などという原始的なやり方になってしまうのは、あるあるではないでしょうか。

2024 年現在、生成 AI は盛り上がっていますので、そろそろ AI に尋ねるだけで上手い感じに並べてくれるツールが出てくると思います。ただ技術的に精度面で不安がありますので、キラーと呼べるほどの存在にはまだまだ至らないでしょう。たとえば不要な情報が並ぶことがある、ならまだマシですが、必要な情報が並ばないことがある、は怖くて容認できません。

注視

次に注視についてですが、これは特定のタスクを注視するということです。

タスク管理ツール的に言えば、あるタスクが表現されたページを開くことに相当します。このページにはタスクの属性値が書かれており、誰でもタスクの状況を知ることができたり、何なら書き足したりすることができます。近年ではコメントやメンションなどチャットの機能を備えたものもよく見られ、ある程度のコミュニケーションも担えてしまいます。

多機能に目が眩むこともありますが、**注視の本質はいつでも関係者なら誰でも自分の好きなタイミングで（そのタスクだけに）注目できること** です。なぜそんなことがしたいかというと、状況を見たいから、または状況に介入したいからです。全体を俯瞰してもあたりをつけられるだけで、じゃあ具体的にどのタスクがどこで詰まっているのかといったことは、そのタスクを見ないとわかりません。そうでなくとも、仕事では色んな人が色んな都合で動いていますから、各自のペースで見れた方が良いに決まっています。場合によっては介入が必要なこともあるでしょう。このようなことを実現するためには、タスクごとにページを独立させ、いつでも誰でも見れるようにすればよいわけです。

これはデジタルツール特有の機能であり、アナログツールだとこうはいきません。あるいは最低でも 1 タスク 1 紙片のような形で場所を取ることになるので、その場所に本人が出向く必要があります（異世界ギルドのクエスト掲示板をイメージしてもらればいいと思います）。またデジタルツールであっても、Web サービスレベルのものでなければこうはなりません。Excel による表や票で管理されたものは、俯瞰こそできますが、注視はできません。

ここまでを踏まえると、**各メンバーが自分のペースで自律的に働けるかどうかは、注視の機能がどれだけ充実しているかにかかっている** と言えます。Slack や Teams などビジネスチャットはすっかり有名となりましたが、なぜこれらのツールでリモートワークが捗るかというと、これらが

注視機能的に優秀だからです。メッセージをやり取りできて、リアクションできて、ファイルを添付できて、必要ならメンションで通知も飛ばせて、しかもかんたんに使えて――とかなり優秀なのです。チャットはチャットであり、タスク管理ツールとして見るのは無理に聞こえますが、そうでもありません。ビジネスチャットには部屋の単位（ワークスペースやチーム → チャンネルやチャネル → スレッド）があり、これらがまさにページに相当します。1ページ1タスクというより1ページ1組織あるいは1ページ1目的になりがちですが、組織や目的はタスクの集合ということもできます。つまり「大きなタスクごとにページをつくることができ」「メッセージのやりとりという馴染みのあるやり方を重視したコンセプトの」「注視機能的に優秀なツール」ということもできるのです。もっとも、実際にそれでタスク管理ができているかというと、そんなことはなくて結局リモート会議地獄になるのですが。

この注視機能の洗練に注目したのが、次の章で述べるパートナータスク管理です。タスク管理とメッセージのやりとりを上手く融合しており、それでありながらビジネスではなく私生活としてかんたんに使えるというバランスも確保していて、上手いなと思います。この発想をもっと膨らませれば、プロジェクトタスク管理ももう一皮くらい剥けるのではないかなと筆者は考えています。その片鱗はある程度見えていて、後の章で紹介します（高度なタスク管理）。

Alternate モデル

プロジェクトタスク管理において俯瞰と注視がどのように実現されているか、についてはいくつかの段階があります。これを **Alternate モデル** と呼びます。

概略を示す前に用語を定義してます。俯瞰的な目線を **マクロ（Macro）** と呼び、注視的な目線を **ミクロ（Micro）** と呼ばせてください。

概略

以下のように 4 つの段階があります。

段階	名前	実現範囲	例
1	N-Micro	注視のみ	タスクが各自の脳内にのみ存在する
2	1-Macro	俯瞰のみ	メンバー全員が見る何とか票や何とかボード
3	1-Macro N-Micro	俯瞰と注視	固定的なビューを持つタスク管理ツール
4	N-Macro N-Micro	俯瞰と注視	ビューをカスタマイズできるタスク管理ツール

N-Micro

注視的な目線のみが存在する段階です。

最も単純な例は「各自の頭の中にのみタスクが存在する」状態です。ビジネスではなく、私生活や趣味の文脈だと基本的にはこれでしょう。いちいちタスク管理するのが面倒くさいし、それでも許される、かつ何とかなるからです。またビジネスであっても、少人数の場合はこれで通用することがあります。上手くやればティール組織のように「その都度コミュニケーションするから大丈夫」で済む世界もつくれますが、それができれば苦労はしないでしょうし、実際ティールのメンバーは精鋭揃いです（あるいは組織運用とメンバー教育にかなりの力とお金を入れているようです）。

この段階は原始的――テクノロジーや方法論が進んでおらず人手的で、場当たり的で、閉鎖的で、変更にも強くないといった体質の表れでもあり、労働環境の原始性との相関は高いと思います。特に就活や転職時だと「タスク管理はどのように行っていますか？」がキラークエスチョンになるでしょう。答えられなかったり、そもそもタスク管理を知らなかったりする場合は、おそらく原始的です。といっても、タスク管理は知らない方がマジョリティですから、気にしない人も多いと思います。肌感覚で言えば、タスク管理を使っている人の割合は「パソコンが使えて、チャットツールも使えて、必要に応じてソフトウェアをインストールしたり設定をカスタマイズしたりまでできる人」よりも少ないです。

他の例としては、タスクの進捗が物理的（あるいは画面越しに）見えるという形の注視もあります。作業所や現場など物理的に色々つくる場面だと、誰が何をどこまでつくっているかはある程度見えると思います。デジタルでも同じことで、たとえば共有フォルダに皆でファイルをアップロードしたり仕上げたりする場合、そこを見れば状況がわかります。といっても、これらは間接的な状況確認にすぎず、結局各自の脳内で各自がタスクを管理していることからは逃れられません。A さんは「もうできた」と言っているが、B さんは「いやそれはまだでしょ」と言っている、などが起きます。A さんの脳内ではタスクは完了しているが、B さんの脳内ではそうではないのですね。

N-Micro は、いわばメンバーの数（脳の数）だけ原本が存在するようなものです。俯瞰できないのはもちろん、タスク各々の状況を知ることさえ苦労します。

1-Macro

俯瞰的な目線のみが存在する段階です。

最も単純な例は「管理者の頭の中でのみ俯瞰が組み立てられている」状態です。これはリモートワークを行えるような大企業でもよく見られるものほどありふれたもので、管理者はメンバーと高頻度にコミュニケーションをして俯瞰を組み立てます。メンバーは、管理者と話さない限りは引き出せません。あるいはメンバー自身でも情報収集をすれば組み立てることはできますが、管理者ほど集められないのでたかが知れています。これは明らかに管理者が楽をするためだけの原始的な形態であり、管理者がボトルネックになります。管理者が優秀だとカバーできますが、拘束や束縛は厳しくなります。上下関係も激しくなりがちです。

このボトルネックを解消するために、何らかの俯瞰装置を用意するやり方もよく使われます。WBS やガントチャートといった言葉を聞いたことがあるでしょう。そうでなくとも、何々管理票とかボードとかいった形でタスクを見える化する取り組みは（タスク管理を知らなくても）自然に発生します。これだけでもプロジェクトタスク管理はまかなえますが、現代では不十分なことが多いです。というのも、このような俯瞰装置には「よりすぐりのタスク」だけが載るからです。よりすぐり以外のタスク（こぼれたタスク）は無視されます。が、現実は、特にデスクワークを伴う場合はそんな単純ではなく、仕事にも人にも色々あるわけで、こぼれたタスクも多いわけです。多いのに管理されない――そこでひずみが生じて、どこかで誰かが（ひどいと全員が）無茶をすることになります。管理者としても、何度も管理対象を変えるのは面倒くさいですし、自分が直接動くわけではないので痛くないですし、そもそも仕事の形態としてよりすぐりのタスクをもう変更できない（契約されていたりその内容で予算が降りていたり）状態になっていたり、など色々あるので融通が利きません。

総じて、トップダウンの強さと固定的な視野からは逃れられません。

1-Macro N-Micro

俯瞰と注視の双方をサポートした段階で、固定的なビューを持つタスク管理ツールを使っています。

タスクはツールに登録され、日々状況が更新されていきます。それらのデータから俯瞰的な見栄え（ビュー）をつくることもでき、このビューを見ることで管理者（必要ならメンバーも）は状況を把握できます。タスクごとのページもあるので、細かいフォローも可能です。また記録も残るので、記憶によるいいかげんな判断もなくせます。近年では DX――デジタルトランスフォーメーションが叫ばれていますが、DX も結局はデータ化とその俯瞰です。データドリブンという言葉も使われたりもします。プロジェクトタスク管理の、この第三段階は、いわばタスク管理に関する DX と言えます。

この段階ではツールの導入により「タスクが外に出て」「誰でも扱えるようになる」ので、管理の独占と不透明性を軽減できます。**タスクという情報を共通言語にしてコミュニケーションが取れる** と言い換えてもいいでしょう。メンバー同士で議論することもできます。とはいえ意思決定者がいた方が何かと便利ですから、管理者がその役割を担います。メンバー達でできるなら管理者自体が要りなくなります。

プロジェクトタスク管理におけるキャズム

第三段階とも呼べる 1-Macro N-Micro に至るには、非常に高いハードルがあります。これを **プロジェクトタスク管理におけるキャズム** と呼ばせてください。キャズムとは深い溝、の意味です。有名なのはイノベーター理論におけるもので、新製品を（少数のコアなファンを越えて）マジョリティに届けるのには非常に高いハードルがあるとしています。

1-Macro N-Micro もまさにそうで、タスク管理ツールの導入やメンバーのスキルはもちろん、形骸化させないよう日頃から入力しこれを使ってコミュニケーションを行うところまでやらねばなりません。これがとても難しいのです。「全員に同じデジタルツールを使わせる」「かつ常用させる」というと、いかに難しいかがイメージできるのではないでしょうか。

実際、この第三段階のツールは、主にクリエイティブな仕事をする人や組織で使われています。クリエイターは仕事柄、ツールの学習機会が多い生き物ですし、自分のペースで働くことの重要性もよく知っている（ペースを守るための努力に余念がない）からです。逆をいうと、クリエイターではない一般人が軽々と越えられるものではないのですね。

一般人の集団でキャズムを超えるには、長い時間をかけて啓蒙し続けなければなりません。ツールの費用や啓蒙役の人件費もしっかり確保する必要があります。つまりは経営者（あるいは大企業なら財布を握る役割や権限の者）の協力が必要不可欠です。再び DX の話になりますが、まさに DX もそうで、実質経営者次第と言えます。このことは IPA の DX 白書 2021 や マッキンゼーの緊急提言 でも強調されています。

プロジェクトタスク管理の場合、経営者ほど大きなスケールで見る必要はありませんが、タスク管理が適用される組織単位（チームやプロジェクト）内という範囲で同様の力学が働きます。マネージャーなのか、部長なのか、その他のキーマンなのかは状況次第ですが、その組織単位の経営者に相当する誰かの協力と投資が必要なのです。

N-Macro N-Micro

俯瞰と注視の双方をサポートした段階で、ビューをカスタマイズできるタスク管理ツールまたは仕組みを使っています。

第三段階の 1-Macro N-Micro では、ビューは固定的でした。ツールがサポートするビューしか使えなかったのです。「こういう観点で見たいのに……」といった要望があっても叶いません。叶わないので別のツールを探すか、それができなければ形骸化――結局打ち合わせをして人に聞いたりといったことになってしまいます。ツールの調査、移行、定着も大変で、そもそも探して済むのなら苦労はしません。形骸化はいつでもどこでも口を開けて、私達を待っています。

このような悲劇を防ぐためには、ツール側にビューをカスタマイズする機能があればいいのです。たとえば monday.com の Work Management では、ウィジェットという形でビューをつくれます。タスクの属性もカスタマイズできるので自由自在です。優先度を 1〜5 の五段階にしたり、優先度とは別に緊急属性をつくって、この値が 1 だったら問答無用でそれから着手しろ、という運用をつくることもできます。ビューとしても、たとえば緊急属性を一覧化するウィジェットをつくればいいでしょう（普段は何も表示されないが、表示された場合は今すぐそれに取り掛かるのだとわかる）。これは一例ですが、このように各チーム各現場に合ったビューをつくることで融通を利かせられます。

API

カスタマイズ方法は一つだけではありません。もう一つ、API を用いたやり方もあります。

API とは「プログラムから呼び出されることを想定した、各機能各データへのアクセスの口」のことで、現代的なデジタルツールは内部的に API を組み合わせています。いわば API はパーツで、パーツを組み合わせて処理を実現するわけですね。このパーツは通常、外には公開されませんが、現代ではこれを公開する潮流があります。公開すれば、他の利用者に使ってもらえるからです。思わぬ使い方を実現してくれるかもしれませんし、そうでなくとも「最悪 API で使って何とかしてね」と逃げることもできます。

インターネットもそうですが、なるべく情報や機能を公開して色んな人に使ってもらうという思想があります。一見すると、無料で公開していてビジネス的に筋が悪そうに見えますが、これで広まれば結果的にファンが増えたり存在感が増したりしてプラスになります。何より目先の利益にとらわれず、幅広く公開して人類に貢献するという意識は、現代ではリテラシーになりつつあるかもしれません。資本主義と相反するように見えますが、消費者もそういう持続的な目線を持ちつつあります（いわゆるサステナビリティですね。この言葉は環境破壊の抑制という文脈が強いですが）。そういう目線を持ってない企業は相手にしない、が起こり得ますし、エンジニアの方ならすでにツール選定において API の有無を見ているのではないでしょうか。

と、話を広げすぎたので戻しますが、要はタスク管理ツールも API を備えていることがあります。API を使ったプログラムを自分でつくって、お望みのデータを取得し、お望みの見せ方で見せればいいのです。無論これには専門的な開発能力が必要ですし、つくるのにも時間がかかります。ただ、その気になればカスタマイズできるよ、というわけです。

ビューの種類

タスクを俯瞰するビューには様々なあり方があります。ツールごとに異なりますし、前述の第四段階ではビュー自体をカスタマイズできるのでした。しかしながら、大まかな種類は限られています。

名前	一言で	事例やツール	特徴
リスト	一覧	□	つくりやすいが、読む気になりづらくタスクの詳細も見えづらい
テーブル	表	□	行と列で二次元的ゆえに情報密度が高いが、認知負荷が高くて疲れやすい
ボード	板	□	カードとその配置が直感的にわかりやすいが、色や見た目の情報量が多くて疲れやすい
チャート	図式	□	煩雑な情報を視覚的に表現できるが、色んなチャートがあり学習コストがかかる
ネットワーク	網	□	ノード同士の繋がりを多次元的に表現できる（詳細は後の章）
センテンス	文章	□	文章で書くため文脈を含められるが、タスク自体の情報量をあまり盛り込めない

ビューを選んだりつくったりしたいときは、この種類を頭に入れておくと良いでしょう。たとえば、これらを超越した魔法のようなビューはありません。所詮はタスクを集めて表示しているにすぎません。どちらかと言えば、そのビューに慣れてしまうこと、特に**ビューを見る人全員が見慣れること**が重要です。あちこち浮気するよりは、まずひとつを決めて使い潰すつもりで取り組んだ方が学べることが大きいです。ここを履き違えると、理想のビューをひたすら追い求める「ビューの亡者」になってしまいます。個人で、趣味でそうするのは好き勝手ですが、プロジェクトタスク管理という文脈でそうするのは迷惑以外の何者でもありません。

さて、種類との向き合い方を述べたところで、各種類の詳細に入りましょう。

リスト

リスト（List）とは箇条書きのことです。

1行1タスクで並べて表示します。並び順がありますので、フィルタリングやソートにより見やすくできることがあります。

メリットはつくりやすいことです。TODOリストもそうですが、手作業でつくるのも比較的容易いです。何せ並べるだけですからね。もちろん、どんな情報をどんな順番で並べればいいかは肝心です。極論、1000個のタスクがあったとして、タスク名だけをランダムに並べたとしても、ほとんど役に立ちません。タスク名と締切を表示して、締切の降順で並べる、だとだいぶマシでしょう。

デメリットはいくつかあります。

まずは読む気になりづらいことです。というのも、リストは全面的に言語情報に頼っているからです。下手すると文章ですらありません。もはや羅列です。言葉の羅列。これは人間にとって、非常に抵抗感のあるものです。特に娯楽を思い浮かべるとわかりやすいと思います。ストーリー性のあるものや面白い文章の方が読みやすいですし、言語情報よりは音声や絵や映像の方が消費しやすいですよね。タスクのリストは、ただの羅列ですから、読書が得意な人や活字中毒者でさえ読むのに抵抗があります。そういうわけでリストビューは実用上はほとんど使われず、他のビューをつくったり、あるいは打ち合わせを開いて直接聞いたりされます。リストは思ってる以上に手強い存在なのです。

もう一つ、タスクの詳細情報が見えづらい点も挙げられます。1行1タスクで、言語でしか表示できませんから、表示できる情報量に限りがあります。実際、現在のプロジェクトタスク管理ツールでは、次で紹介するテーブル形式が主流です（行と列で二次元になるので情報量が増える）。

テーブル

テーブル（Table）とは表形式のことです。

1行1タスクで並べますが、列も存在しており1列1属性に対応します。フィルタリングやソートは行を増減する調整ですが、テーブルでは表示列の調整（カスタマイズ）も可能です。また、列名をクリックするとソートになる（昇順と降順を交互に切り替えるなど）機能もよく知られています。

メリットは親和性と見やすさです。まず表計算ソフトの Excel が圧倒的な市民権を得ていますし、そうでなくとも表という概念はビジネスにおいて極めて身近です。社会人であれば、あるいは学生やアルバイトの段階で、ほぼ必ず触れる概念ではないでしょうか。圧倒的に日常に馴染んでいるのです。もちろん馴染みうんぬん以前に、行と列で二次元的に表示しているから見やすいとも言えます。当たり前のことを小難しく言っているように聞こえますが、表は本当に偉大な発明だと思いますし、現代でもまだまだ有力です。**「テーブルビューを選んでおけば間違いなし」と言えるほど無難な選択肢でもあります**。より大胆に言えば、テーブルビューでプロジェクトタスク管理が成立しないのなら、それ以前の問題がはらんでいます。プロジェクトタスク管理、特にビューによる俯瞰を機能させたいのなら、まずはテーブルビューで機能させるところを目指すのが良いでしょう。

デメリットはいくつかあります。

まずは疲れることです。タスク情報が隙間なく敷き詰められるので、ちゃんと読むと認知資源を見る見る食います。ネットサーフィンや SNS の巡回、あるいはメールやチャットの昇華とは比になりません。まるでタイムラインのように、高頻度にテーブルビューを見る方がたまにいらっしゃいますが、非常に疲れるので推奨しません。とはいえ工夫すれば消耗は抑えられます。たとえば細かい粒度は端折って表示する（大タスクレベルのみ表示するなど）、知りたい条件でフィルタリングやソートをかけるだけにする（目的特化）、あとはこれは他のビューにも言えることですが **欲しい情報が記入されてないのに読み取ろうとしない** ことです。エスパーではないし、推測したと

ころで外れることも多いですから、ちゃんと記入してもらうよう働きかけましょう（それが難しいのは前述、キャズムとして述べたとおりです）。

ボード

ボード（Board）とは板形式のことです。板というと聞き慣れませんが、掲示板や伝言板など何かを貼り付けるスペースを思い浮かべてください。

タスクは付箋やカードといった形で表現され、これを二次元空間に並べることで俯瞰できるようにします。自由に配置させるというよりは、整然と並ばせるものが多いです。特にプロジェクトタスク管理はカンバン方式の影響を受けていることが多く、未着手・進行中・完了といった領域が区切られていて、そこにタスクカードを配置し、また動かします。

メリットは見やすさです。リストのように言語の羅列ではなく、テーブルのように情報を詰め込んでもおらず、適度な余白とデザインの反映されたカードが並んでいる状態ですので、視覚的に易しいのです。カード内にもある程度の空間がありますし、ビュー側でサイズ調整もできますので、表示する情報も工夫できます。デザイナーの腕の見せ所でもありますね。

そして肝心なのは、カードであるため「場所を移す」という操作が使えることです。カンバンもまさにそうですが、たとえば開始したタスクがあれば、そのカードを未着手ゾーンから進行中ゾーンに移せばいいのです。ツール上ではドラッグして移動させるといった直感的な操作で行えます。これは「タスクの詳細ページを開いて、状態属性を未着手から進行中に変更する」みたいな操作よりもはるかにわかりやすいし、手間も少ないです。

これだけ見やすく、手間も少ないと、慌ただしいプロジェクト下でも使えます。みんなを集めて、会話しながらボードを見て「多すぎるからこのタスクは捨てるか」「これはもう終わったよね、完了に移すね」などささっと操作して状況反映できます。リストやテーブルだとそうはいきません。

デメリットは二つあります。まずは別の意味での疲れやすさがあることです。リストやテーブルよりもグラフィカル（視覚的）なので、視覚的にうるさくなることがあります。また空間的な配置にもなるので、空間認識が苦手な人にも負荷が高い傾向にあります。大まかな目安として、**ボードが合う人とリストが合う人がいます**。視覚性や空間認識に強い人はボードが合いますが、そうじゃない人は合わないかもしれずリストの方が馴染めたりします。次に、表示できるタスクの数が少ないため、実質的にアジャイルなど「1〜2週間ごとにやることを決めてまわす」感じの仕事も仕方が要求される点です。仮にボード上に並べられるタスクが 20 個の場合、同時に扱えるのは 20 個までとなります。タスクが全部で 100 個あったとしても、扱いきれません。そのため「今月はこの 20 個にフォーカスしようか」など区切りを入れて、その分だけをボードで扱う、といった対処が必要なのです。

ボードビューはリストやテーブルよりも後発なので、使えるなら使いたいところです。しかし、合う合わないはあるので、合わないようなら無理しない方が良いでしょう。

チャート

チャート（Chart）とは図式形式のことです。

何とかチャート、何とかグラフ、何とか図（ダイアグラム）といった視覚化のやり方がありますが、そのあたり全般を指します。タスク管理としては、主にフレームワークとして整備されていることが多いです。

例:

- WBS
- PERT図
- ガントチャート
- バーンダウンチャート
- 累積フロー図

チャートの本質は、縦軸や横軸など「特定の軸」を選んで、それに基づいてデータ（この場合はタスクに関する情報）を配置し、兆候の良し悪しを視覚的にわかるようにすることです。特に危険な兆候を知るのに役立ちます。重なっていたら危ないとか、領域が広がったら危ないとか、線の傾きが大きくなると危ないなど、直感的に危なさを把握できます。何を知れるかはチャート次第ですので、そのチャート（のフレームワーク）をきちんと学ぶ必要があります。

色々なチャートを集めたビューはダッシュボードと呼ばれることもあります。タスク管理の文脈では聞きませんが、統計分析やセキュリティの文脈ではよく登場します。データドリブン経営などと言ったりもしますが、データを重視する潮流も近年は盛り上がっています。今後はダッシュボードレベルでチャートを重用したタスク管理ツールも出てくるかもしれませんね。

さて、メリデメに入りましょう。

メリットは煩雑な状況を視覚的かつかんたんに把握できることです。諸々のフレームワークは、それができるよう工夫が凝らされています。素人の思いつきでできることではありません。つまりチャートとは自らつくるものというより、何らかのフレームワークに基づいたもの（かつツール側がサポートしているもの）を使うものです。正しく使えば、視覚的にかんたんに俯瞰できるようになります。

デメリットは、まさにそのフレームワークの勉強が必要になることです。ツールによっては独自のチャートがサポートされていることもありますが、その場合もやはり読み方などを学ばねばなりません。もちろん学んで終わりではなく、実際にチャートを機能させ、それを読み解いて状況判断ができるようになるまでには訓練も必要です。

またチャートごとに適切な状況が想定されており、ご自身の状況と照らし合わせる必要があるかもしれません。たとえば WBS やガントチャートは、一度立てた計画のとおりに動くという安定的な場面に適していますし、バーンダウンチャートや累積フロー図などはアジャイル開発という

「計画を動的に変える」「1-2週間など短期間の計画と実践を繰り返す」場面に適しています（なので計画よりも実績の可視化に比重を置く）。計画駆動な場面でバーンダウンチャートを使ったり、アジャイルな場面で WBS を使ったりするのはアンマッチでしょう。

ネットワーク

ネットワーク（Network） とは網形式のことです。

網というと直訳的でわかりづらいですが、点を線で結んだような構造を指します。点のことをノード、線のことをエッジと呼ぶこともあります。理論としては専門的ですが、人間関係、ウェブサイトや書籍の言及関係など身近にもよく見られる構造です。これをタスクにも適用します。つまり、タスクとタスクの関係を描いたネットワークを見せるのだと考えられます――と濁しているのは、**ネットワーク形式のビューがまだ存在しない** からです。

そもそもネットワーク構造をつくるためには、タスクとタスクを日頃から接続する（何らかの関係性があるのでその旨を登録する）必要があるのですが、通常そんなことはしません。プロジェクトタスク管理としても想定された動きではありません。ツールによっては言及先と言及元を表示する程度の機能を持つもの（Redmine や GitHub Issues）もありますが、あくまでもタスク各々つまりは俯瞰ではなく注視の話であって、すべてのタスクの接続関係をネットワークのビューで俯瞰することはできません。もう一つ、ワークフロー機能を備えたツールがあり、ネットワーク形式で見せることが多いですが、これは「処理の流れ（ワークフロー）」のビューであってタスクのビューではありません。ワークフロー機能は、タスク管理というよりはビジュアルプログラミングの話になります。

最近ではノートベースのツールが出てきており、これをタスク管理用途で使えば一応できないことはないのですが、高度な話なので後の章で述べます（文芸的タスク管理）。

そういうわけで現状事例がないため、メリット・デメリットの記載は割愛します。

(余談)ビジュアルプログラミングとタスク管理、Automation

ビジュアルプログラミングとは、ビジュアルに（視覚的に）プログラミングを行うことです。何らかの処理を行うパーツを配置して、どのパーツからどのパーツにどんな情報を渡すか、といったことも考えて、まるでパズルのようにつなぎ合わせることで全体の処理をつくります。

パーツの品揃えと働きは理解する必要がありますが、難易度的にはゲームと大差がなく、もちろん個人差はありますが未経験の小中高生でも使えるようになれます。教育用の Scratch が有名です。

これは **プログラミングという難しい営みも、ビジュアルだと比較的かんたんにこなせるようになる** とも言え、近年ではノーコード（プログラムコードを一切書かない）・ローコード（ほとんど書かない）と呼ばれるジャンルも出てきました。たとえば、プログラマーではないバックオフィス業務の

人が、自分で、ビジュアルプログラミングを使って、業務改善ツールをつくる――なんてことも可能になりつつあります。日本ではサイボウズの kintone が知られていると思います。

さて、このような着眼は、タスク管理にも来ています。それが Automation と呼ばれる機能です。Automation は直訳すると「自動化」で、タスクに関する操作を自動化するための仕組みです。以前の章の「省力化する」にて少し述べましたが、「XXX が起きたら YYYY を行う」という形で事前に処理を定義しておくと、XXXX が起きただけで YYYY が自動で処理されるようになります。お決まりの「ワークフロー（処理の流れ）」があったら、それを Automation で自動化しておくと楽になるわけです。別の言い方をすれば、**Automation とは（タスクの操作に関するワークフローの）自動化を行う仕組みである** とも言えます。

少々小難しくなりましたが、要点は 3 点です。

- 1: ビジュアルプログラミングという、プログラミングを比較的かんたんに行う手段がある
- 2: ビジュアルプログラミングの流れはビジネスにも降りてきており、一般の人でも処理を組み立てることができつつある
- 3: タスク管理にも降りてきており、一部ツールはすでにサポートしている。タスクの操作を自動化できる

よって、ビジュアルプログラミングに慣れ親しんでおくと、今後 Automation にも頼りやすくなるでしょう。

センテンス

センテンス（Sentense） とは文章形式のことです。

センテンス型のビューも既存事例はありません。強いて言えば、生成 AI の力でつくれないこともないですが、結果の正しさを保証できないため実用性はあまり高くないでしょう。というわけで、当面は私たち自身が手作業でつくることになります。何をつくるかというと、**俯瞰できるような状況を「文章として」事前に書いておきます。**

例を示します。書籍をつくるプロジェクトだと考えてください。Before はリストビューで、After はセンテンスビューです。

Before（リストビュー）:

- □目次案
- 43% 第1章
- 20% 第2章
- 00% 第3章
- 16 参考文献

After（センテンスビュー）:

> まずは「□目次」をつくる。小説におけるプロットと同様、羅針盤があった方が書きやすいから。ここは譲れない。本文に入る前に必ず目次は確定させる。ただし仮でよい。どうせ変わる。
>
> その後は「1章(43/100)」、「2章(20/100)」、「3章」と進めていく。各章は独立しており、並行して進められるが、1章だけは基礎用語を導入するため、先にひととおり完成させるべきである。
>
> 「参考文献(16件)」は執筆の都度、追記していくべきである。全章書き終えたあとに改めて整備するため、章執筆中は雑で良いが、urlや書名など必要な情報は必ず揃えること。また後でリンクを直すのも手間なので、リンクの正しさは都度確認すること。

センテンスビューでは文章による補足が書き込まれています。ここまでビューはタスクを上手いこと並べて表示するものでしたが、単に表示するだけでなく、文章も書いています。もちろんタスクの情報が見えないと話にならないので、それはツールの力で自動で表示します。上記例で言えば「□」や「43/100」などがそうです。これらはいちいち手作業では書きません。その他詳細は後の章で扱います（文芸的タスク管理）。

メリットは文脈がわかることです。ビューには通常、タスクが並ぶだけでしたが、センテンスビューでは文章にて文脈も書いているので意思決定がしやすくなります。上記例で言えば、何をどのように書いていけばいいかが書いてあるので、チームで振り返った時でも気付きやすいですよね。「1章を先に完成させるべきじゃないのか」とか「2章がちょっと進んでるけど、1章にもっと注力するべきでは」など気付きやすいはずです。

このようにガイドライン的な簡潔な文章を書いて羅針盤にするアプローチでは現代のビジネスでもよく使われます。転職活動をしたことがある方はよく見かけると思いますが、働き方や哲学といったものを文章化して紹介している会社は珍しくありません。パーパス経営とか、企業理念とか、そういった言葉を耳にする機会も増えてきたのではないでしょうか。また、自己啓発の分野でもミッションステートメントなど、自分の人生観を言語化して掲げる取り組みはよく知られています。**文章的な羅針盤は古くからよく知られたテクニックです**。それをタスク管理の、ビューにも生かしたものがセンテンスビューです。

次にデメリットですが、文章を書くのが大変です。文章力の話ではなく、合意形成の面で、という意味です。プロジェクトタスク管理では、ビューはメンバー全員が見る可能性があるものですから、全員が納得できる文章が求められます。文脈は「これでいいですよね？」と決める必要があります。曖昧は許されません。曖昧な状態を反映したところでセンテンスビューの旨味はありませんし、さして意味のない文章は読み飛ばされて形骸化するだけです。

文章力の話ではないと書きましたが、この点も地味にキツイかもしれません。というのも、センテンスビューでは、あまりタスクの情報を出せません。テーブルビューはもちろん、リストビューよりも情報量は低くなります。ということは、あまり広範囲を俯瞰したビューはつくれないことになります。センテンスビューをいくつもつくらねばならなくなるかもしれません。たとえばタスクが100個あった場合、テーブルやリストだと数個以内のビューで足りるでしょうが、センテンスだと数個ではす

みません、仮に 10 個ごとにつくるとしても 10 センテンスになります。中々にハードですね。それから、ガイドラインと書きましたが、状況が変われば微修正は当然入るので、センテンスも修正します――と、要は **文章を書いたり修正したりという営みが生じます**。文章が苦手な人にはキツイですし、得意でない人でもキツイでしょう。

さらに言えば、現状ツールが無いこともデメリットでしょうか。特にタスクの情報を自動で表示するものがありません。上述した例のような文中の展開を行えるツールはまだないと思います。センテンスビューを実現したければ、前述の第四段階でも述べたように、ツールの API を駆使して、自分達でそういう仕組みをつくらねばなりません。

シャドータスクを上手く扱う

ビューの話はここで終わるとして、次にプロジェクトタスク管理が上手く行かない主因の一つを取り上げます。シャドータスクです。

シャドータスク

シャドータスク（Shadow Task） とは、タスク管理ツール上で管理されてはいないが、事実上タスクと呼べるレベルで存在しており、私達のリソース――時間や体力やお金や精神を食っているものを指します。残業を勤務管理に計上しないことをサービス残業と呼びますが、同じようなものです。管理はしていないが、実際には存在しているわけで、それなりに（どころかたいていは「かなり」）負担になっています。

シャドータスクの例:

- コミュニケーション全般
  - 仕事に関する相談や議論
  - 当日発生した打ち合わせ（緊急 or 思いつきのいずれかが多い）
  - その場で行う指導や教育
  - 根回し
- 社内雑務全般
  - 全社から降ってきた依頼対応
  - イベント運営のお手伝い
  - 備品やシステムなどの故障・障害に伴う対応
  - 同上、アップデートやバージョンアップ
- 穴埋め全般
  - 業務の引き継ぎ（特に突発的なものなど業務の一環として正式に準備・計画されていないもの）
- メンテナンス

    - 始業前や就業後の準備
    - デスク、デスクトップ、その他情報の整理整頓
    - 昼食や休憩や買い出し（移動時間含む）
- キャッチアップ
    - 業界動向や顧客の学習
    - ドキュメントや書籍の読解
    - 雑談の場やイベントへの顔出しや寄り道
- ツール
    - タスク管理ツールの学習や操作
    - メールやチャットの操作
    - PCの準備、設置、起動、終了、撤収
- プロセス
    - 手順
    - 手続き
    - フロー
- 通勤・出社

シャドータスクは至るところに、想像以上に存在しています。インパクトのある例として通勤が挙げられます。パンデミックによりリモートワークでも通用することがわかった組織では、通勤も業務時間に含めたり（※1）、交通費を全面的に立替精算にしたりしています。長らくシャドーだった通勤をすくいあげたわけです。

- ※
    - 1 会社として正式には認めていないが、現場の融通で含める、とのバランスもあります。この件に限らず、現場で融通を利かせることは組織力学または処世術としてよくあることです。このようなことが必要なのも、正式なルールに厳密に従っていえばシャドータスクに忙殺されてしまうからです。もちろん、行き過ぎるとコンプライアンスを犯してしまい損害に通じることもあります。

シャドータスクを捉えることの重要性

通勤の例だけでもよくわかりますが、シャドータスクは私たち従業員にとって都合が悪いものです。それなりに負担があるのに、タスク管理上は計上されないので、完全にこちらが割りを食ってしまいます。

ここで「直接関係がないタスクなのだから管理しなくてもいいのでは」と思われるかもしれません。一見するとたしかにそうですし、実際シャドータスクも全部ツールに入れても邪魔なだけでしょう。そうではなく、**シャドータスクによりリソースをそれなりに消費しているという事実がある** 点が重要ですよ、という話です。

たとえば都内のオフィスへの通勤に片道 1.5 時間かかる人がいたとします。この人は人混みが

非常に苦手で、片道分通った後は 30 分休憩をはさまないといけない体質だとします。また給料も十分ではなく、家賃を押さえるために郊外に住んでいます。さて、この人の 1 日の持ち時間は定時分の 8 時間（7.5 時間とかでもいいです）だと見ていいでしょうか。答えはノーです。実際には「2時間のタスク」を「1日2回」抱えているようなもので、より率直に言えば毎日 4 時間のサービス残業をしています。そんな状態で、毎日 8 時間も働くのはかなり不利でしょう――このような事情は、プロジェクトタスク管理ツール上では見えません。

別の例を挙げます。上位部門の審査と承認が何かと求められるチームがあるとします。審査には資料や打ち合わせの準備、打ち合わせへの参加、参加後は議事録の記入と共有が必要だとします。一回の審査あたり平均して 45 分費やしてるとします。もちろんこれは雑務にかかる時間にすぎず、他にも上長の審査が始まるまでの時間待ちもあります（たとえば 3 日後であるなら 3 日待たないといけません。この審査は一人あたり 1 日 1 〜 3 回行っています――この場合、事実上一人あたり 1 〜 2 時間を費やしていることになります。が、タスク管理上では管理されていないと、「タスクの進みが思ったより遅いぞ？」「なぜだ？」となってしまいます。単純計算でも 1 〜 2 時間抜けているのに、実際は待ちもありますし、仕事と審査とでは頭を切り替えたりもしますよね、そういったことも踏まえると、筆者としては毎日半日近く飛んでいるに等しいのではと感じます。

他にも例は無数にあると思いますが、要するにタスク管理ツールが示す情報だけでは足りないのです。特にシャドータスクは、思っている以上に私たち従業員のリソースを食いつぶしています。結果として**「なぜだかわからないが、タスクの進行が想定より遅い」現象や「まともにタスクをこなせる人材が少ない」現象が起きます**。当たり前ですよね、シャドータスクに引っ張られているわけですから。また、上手くやれている人も、シャドータスクにも負けずに頑張れている「強い人」にすぎません。

このような現象を **シャドーエフェクト（Shadow Effect）** と呼ばせてください。ようやく結論になりますが、なぜシャドータスクを捉えることが重要かというと、シャドーエフェクトを減らす手がかりになるからです。よくわからないまま振り回されるのではなく、具体的にこういうシャドータスクがあって、これだけの影響が出ているから、こういう対処をしよう、と建設的に改善していけばシャドーエフェクトは減らせます。

シャドーエフェクトは意識的に目を向け、減らして行かねばなりません。分かりづらいがゆえに無自覚である人や組織も少なくないからです。意識的にやらねば土俵にすら立てません。

シャドータスクの扱い方

プロジェクトタスク管理において、シャドーエフェクトを減らすには――シャドータスクを見つけて対処するにはどうすればいいでしょうか。

まず前提としてシャドータスクには二種類あります。そのプロジェクトに関するタスク（ **ダイレクトシャドー** ）と、そうではないタスク（ **インダイレクトシャドー** ）です。上記の例で言えば、通勤はインダイレクトシャドーです。通勤というタスク自体は通常、プロジェクトとは何の関係もないはず

です。ただし出張など例外もあります。一方、上位部門の審査はダイレクトシャドーでしょう。このような審査はプロジェクトの一環として行うはずです。対処は **ダイレクトシャドーかどうかで変わる** ので、この区別は意識してください。

対処のアプローチは三点あります。

- 1: キャパシティベースのタスク管理を行い、シャドーエフェクトの影響を考慮する
- 2: バッファとスラックを確保し、各自がシャドータスクに対処する余裕をつくる
- 3: ダイレクトシャドーをタスク管理する

1: キャパシティベースのタスク管理

キャパシティ（Capacity） とはあるチームの、ある期間における処理能力を指します。たとえば 1 週間で大体 15 タスク消化できるのなら、キャパシティは 15 です。15 タスク以上は割り当てない方が良い、と判断できるでしょう。キャパシティの細かい計算方法や精度の高め方は色々ありますが割愛します。重要なのは「処理能力ベース」で考えて、それ以上は積まない、という考え方です。カンバン はまさにキャパシティベースと言えます。

この考え方を採用すると、シャドーエフェクトをスルーできます。というより **シャドーエフェクトが存在する前提でのキャパシティを採用します**。これを **シャドー・ベースド・キャパシティ（Shadow Based Capacity, SBC）** と呼ばせてください。本当にチームにとって無理のないキャパシティを考えると、自然とそうなります。たとえば定時間が 8 時間だとして、シャドータスクのせいで実質 5 時間くらいしか働けない人が 3 人いたとしても、無理のないキャパシティを想定すれば、チームのキャパシティは「その 3 人分については 5 時間働いた程度のもの」になります。「3 時間分働いてないじゃないか」と怒ったり、無理やり働かせたりするのではなく、このキャパシティが現実的なラインなのだからこれでいい、とするのです。

理屈上は難しくないですが、実際に SBC を採用するのは中々に困難です。まずキャパシティの計算頻度と見積もり精度をある程度安定させなくてはなりません。アジャイル開発というソフトウェア開発手法ではこの手の理論を扱っていますが、エンジニア向けですし練習も要ります。

もう一つ、難しさを助長しているのが「強い人」の存在です。「強い人」とはシャドータスクの負荷に慣れていて、かつ耐えることができる人達です。彼らは SBC の視点を持てませんし、持てても行動にまでは落とし込めません。別に SBC にしなくても何とかなるので、あえて SBC にしようとは考えないのです。部屋が汚くても別に生活できるのならわざわざ掃除しないのと同じです。どころか、SBC にすると、その分労働時間が減って生産性も減ります。別に耐えられるのに、わざわざ減らすかというと、減らさないでしょう。時間給が絡んでいるならなおさらです。そういうわけで、「強い人」が多ければ多いほど、SBC は適用しづらくなります――といっても難しい話ではなく、単にその人に合ったキャパシティを尊重しましょう、というだけの話ですが、ビジネスでは「強い人」が正義になりがちです。特に日本は平等思想が強いですし、すでに強い人ベースでビジネスが回っていますので、彼らのペースを強要されがちなのです。強くない人達は、そういう経験が一度は、あるいは何度もあるのではないでしょうか。

2: バッファとスラック

バッファ（Buffer） とはイレギュラーに備えるための時間的余裕を指します。仕事でも（一週間あれば終われそうだけど）見積もりは二週間で出す、とかしますよね。これは「一週間分のバッファを積んでおく」などと表現されます。2 倍という数字に絶対性はありません。数日積むとか、1.2 倍積むとかいったなどもありえます。いずれにせよ、ある程度積んでおくことは経験則として重要です。仕事は何が起こるかわからないものですし、やってみてわかることも多くて、それらに合わせて修正する時間も要るからです。

バッファの必要性はシャドータスクについても言えます。バッファを積んでおくと、思わぬシャドーエフェクトが生じてもバッファの分を使って対応できるようになります。通常、バッファはシャドータスクのために確保するものではありませんが、シャドータスクは（見えていないだけで）ありふれているので、日頃から **なんだかんだシャドータスクに邪魔されることは割とよくある** 想定で過ごすのが無難だったりします。どの程度積むかは状況次第ですが、一週間のスパンで考えるとすると「半日」くらいのバッファをまずは積んでみると良いでしょう。何かと融通が利いて助かるという体験が出来ると思います。といっても、バカ正直に「私は半日分はサボります」とは言えませんので、現実的には自分の意識として心がけるか、冒頭の例のようにしれっと積んでおく形になります。チームメンバーを騙すようで気が引けるかもしれませんが、お互い様です。バカ正直に「シャドータスクのバッファを積みましょう」というと、正論や法規で語らなきゃいけない人達（特に上司など）は立場上潰さないといけなくなりますし、前述の「強い人」達を説き伏せるのも難しいので、しれっと、で良いのです。もちろん正直に議論できるに越したことはありませんが、その場合は 1: の SBC で良いでしょう。

スラック（Slack） とは恒常的な余裕を指します。ここでいう余裕とは時間的にも精神的にも、です。たとえば毎日 1 時間、仕事にとらわれず雑談したりのんびり休んだりする時間を取れるでしょうか。昼休憩のことではなく、昼休憩を差っ引いた現在の定時間 7〜8 時間において、1 時間取れるかという話です。これは毎日 1 時間のスラックを確保していると言えます。スラックのおかげで気分転換や休憩ができますし、雑談から親密感を醸成したり思いがけないアイデアが出たりもします。私たちは資本主義的価値観に縛られており、真面目に忙しなく働かねばならないととらわれがちですが、案外そうでもないです（もちろんスラックが許されない状況や仕事もあります）。スラックを取り上げた書籍 もありますし、1on1 ミーティングという取り組みにて業務時間中に雑談をする例も今や珍しくありません。また、（これはスラックというより労働時間量の絶対性に対するカウンターですが）週休3日制の働き方を採用した企業もあります。

スラックがあれば、シャドータスクの対応に時間を使うこともできます。特に「別に余裕なんて要らない」という人でも、シャドータスクは存在しているので重宝します。仮にシャドータスクさえないとしたら、普通に仕事をすればいいのです。重要なのは **スラックが存在する（必要なら使える）** ところにあります。なければ使えませんが、あれば使えます。要らなければ使わなければいい。在ることに意味があります。現状スラックを正式に採用した事例はまだ無いと思いますが、今後は魅力の一つになるでしょう。たとえば「毎日 1 時間のスラックがあります」といわれたら、どうでしょうか。魅力的な職場に見えてきませんか。

3: ダイレクトシャドーのタスク管理

シャドータスクがダイレクトシャドー――つまり仕事に直接関係している（けど管理されてなくて見えてない）ものであるならば、それを見えるようにすればいい、というのも対処の一つです。具体的にどんなタスクがあるかをきちんと言語化して、タスク化して、担当者を決めて、必要なら期限も決めて、と普通にタスク管理をします。

重要なのは普段の流れに仕込むことです。ツール上で担当者を決めて「空いたときにやっておいてね」ではなく、普段の仕事の流れに組み込むということです。たとえば毎日、朝会にてビューを見ながら今日何をするかを決めている場合、（ちゃんと管理すると決めた）ダイレクトシャドーについても、他の正式なタスクと同じように扱うのです。プロジェクトの一環として、仕事の一つとして、ちゃんと対応します。もし言語化すらできていないのなら、まずは「メンバー全員でダイレクトシャドーを言語化する会」を開催するとか、メンバー全員に「ダイレクトシャドーを洗い出す（想定時間: 1時間）」タスクを割り当てるなどが必要かもしれません。しつこいようですが、空いた時間でお願いしますではなく、業務の一環として正式にダイレクトシャドーと向き合うということです。

と、一般論だけ書いてもわかりづらいですが、いわゆる「改善」は典型的な例でしょう。何らかのダイレクトシャドーを何とかするために、何かを改善する必要はしばしばあります。その改善活動をタスクにするのです。

ちなみにインダイレクトシャドー――仕事に直接関係しないシャドータスクについては扱っていないことに注意してください。というより、扱っても困りますよね。たとえばAさんがリモートワークをしていて、「近所の犬の鳴き声がうるさくて集中できない」「このうるささに耐えるためには耳栓や薬や瞑想といった手続きが必要で、毎日平均 30 分を要している」として、じゃあこの A さんの犬の鳴き声対処タスクをプロジェクトタスク管理するかというと、ノーですよね。これは A さんが自分で対処するべきことです。プロジェクトで行うことではありません。前述のバッファやスラックを使う（ここでメンバーと雑談して対処方法を相談するなどはもちろんアリです）のでもいいですし、キャパシティベースにて最初から 30 分少ない前提でのキャパシティを踏まえる、でも良いです。

プロジェクトタスク管理がついでに管理しているもの

管理も過ぎれば毒です。管理が過ぎれば過ぎるほど融通が利かなくなり、そのための対応に疲弊しながらも本来の仕事に費やせる余地も小さいというダブルパンチを喰らいます。やる気も削がれますし、満足度が下がります。何なら「管理を乗り切るために頑張る・誤魔化す」ことが目的化してしまい、**プロジェクト自体が形骸化してしまいます**。現実は綺麗事ではないですし、そういう場面を見たことがある（何なら現在進行で巻き込まれている）人も少なくないのでは

ないでしょうか。この手の現象は法則としても知られており、たとえばグッドハートの法則があります。

それでも昔は結果さえ伴えば良しとされてきましたが、現在はそうではないはずですし、そうあってはならないとさえ思います。細かく管理しすぎるマイクロマネジメントを好む人は少ないでしょうし、組織のパラダイムも暴力的恐怖政治的なトップダウン → 階級 → 成果主義と階層 → 平等でフラット → 自律と秩序（ティール組織）との変遷を辿ってきています。無論、状況が余談を許さないことはありえますが、前提にしてはならないと思います。かといって何も管理しないと秩序も何もあったものではありません。

話をタスク管理に戻しましょう。プロジェクトタスク管理についても案外色んなものを管理しており、管理過多になりがちです。この点を自覚した上で、管理のしすぎという毒を薄めるのが重要です。本節ではそのヒントをお届けします。

5つの管理対象

タスク管理という言葉の裏には、思っているよりも多くの管理対象が存在します。5 つにまとめることができます。

名前	日本語で	解説	例
Progress	進捗	ゴールに対する進行具合	37%/100
Quality	質	状態の良さや安定性	エラー発生率は2%、73%が満足度が高いと回答
Process	プロセス	順序やルールへの踏襲具合	手順、手続き、形式、法規、規則規約
Resource	リソース	総量に対する消費具合	時間、お金
Property	専有	所有感と従順性	出社や会議など場への参加、（江戸時代の）参勤交代

愚直に捉えると**タスク管理＝タスクの進捗管理** です。それ以上でもそれ以下でもありません。タスク管理とはタスクを終わらせるためのものであって、終わらせるために、どこまで進んでるかを可視化したいのです。3A の本質 としてアサイン、調整、俯瞰と注視を挙げましたが、結局やりたいことは **進捗を知りたい。ただそれだけです**。進捗がわかれば、特に遅れがわかれば対処を打てるからです。

しかし実際には進捗以外も管理されがちです。質も、プロセスも、リソースも、下手すると専有も管理されます。もちろん現実は進捗が見えただけで上手くいくほど単純ではなく、癖の強い

人間を上手く動かすためにも管理は役立ちますが、前述のとおり、やりすぎると毒になります。これらを自覚し、要らない部分をどれだけ削れるかがポイントとなります。

では、各管理対象の詳細を見ていきましょう。対象の性質がわかれば取捨選択――管理するかどうか、あるいはどこまで管理するかを判断するのに役立ちます。

進捗の管理

進捗の管理 とは、タスクの進行具合を管理することです。何らかのゴールがあって、それに向かってどこまで進んでいるかを見ます。シンプルだと「終わったかどうか」の二値で見ますし、まだやってない・着手中・終わったの三値もよく見られます。もう少し細かいと、パーセンテージなどで表現します。

どこまで管理するかは管理者（いないならチーム全員）の腕の見せ所です。早い話、問題無くタスクをこなせていけるなら「終わったかどうか」をたまに確認する程度でも良いのです。その程度で済むのに「毎日 1-100 のパーセンテージで記入しろ」「毎朝報告しろ」などと管理するにはやりすぎです。これがビジネスの文脈だと「それが仕事だろ」「プロフェッショナルだからこそだろ」などと正当化されますが、ただの単純化や現行踏襲にすぎません。

進捗の管理は加算方式が良い です。できるだけ管理しない、という前提があって、それで上手くいかない場合は「上手くいく程度に」少しずつ管理を追加していきます。もちろん結局は加算を重ねて厳しい管理になってしまうこともあります。

進捗の管理が厳しくなりがちな理由は二つで、管理者の都合と報告者の都合です。

まず管理者の都合とは、他にやることがない管理者が管理という仕事を無理に捻出する、あるいは自分の仕事を守るために管理にしがみついている等を指します。通常、組織が大きくなってくるとこれが増えます。組織力学として自然なメカニズムです。ここに抗うには、ティール組織など「健全な組織体系のままで大きくなる」設計が必要です。他にも、単に管理者の資質が未熟なこともあります。たとえば結果が出るまで待つことに耐えられなくて、安直に見たがるという心理があります。このような耐性をネガティブ・ケイパビリティと呼ぶこともあります。ネガティブ・ケイパビリティが低いわけです。もう一つ、管理をするとその管理指標でコミュニケーションを取れるという（管理者から見た）メリットがあります。責任の所在を曖昧にしたり、進行が遅れていることの免罪符に使ったりできるわけです。作文や数字いじりとの相性もいいですし、真面目につくられていたら心理上、報告先も甘くなります。さらに言えば、管理の名目で暇や寂しさや紛らわせたり、部下を詰めて楽しんだりといった隠れ職権乱用もあります。

次に報告者の都合とは、仕事上の報告先と向き合う報告者による都合を指します。たとえば契約で「毎週詳細な進捗レポートを提出しなさい」となっていたとしたら、それはもう管理過多であろうがやるしかありませんよね。実際は上述したとおり形骸化するわけですが、大事故が起こらない限り調査のメスは入らないので長らく続いたりします――と、これだけ見ると報告先が悪いように聞こえますが、そうではないのです。そもそもそんな契約になってしまわないよう、報告先

と接する報告者側が何とかしなければなりません。また、ここまで露骨ではなく、契約など強制力がなかったとしても、管理者と同様、報告者が残念だとその力学が降ってくることがよくあります。鶴の一声、はまさにその一例です。それでも管理者のレイヤーで食い止められたらいいのですが、管理者が事なかれ主義だったりするとイエスマンになってしまい、現場にまで降りてきます。

いずれにせよ人の問題です。管理者にせよ報告者にせよ、管理を担う権限の強い人達が厳しくしまっているにすぎません。進捗の管理を緩和できるかどうかは、このような人をいかに是正できるか、あるいはそもそも巻き込まないかにかかっています。

ここで「ルールやプロセスとして決められているから仕方ないというケースがある」との疑問を抱かれるかもしれませんが、本当にそうでしょうか。後述する「質の管理」については、たしかに決められているケースもありますが、進捗の管理についてはいかがでしょうか。契約で決めてしまったケースを除けば、結局管理者や報告者の都合に帰着できると筆者は考えます。要は厳しめの管理という常識しか知らないから、それ以外のあり方がわからないのです。タスク管理を学ぶ重要性の一つが、まさにここにあります。管理の固定観念から脱するためです。

もう一つ、言及しておきたいのは「大事」でしょうか。何か大きな事件や事故が起きたときに、バカ真面目に厳しい対応を全体に敷く現象はご存知の方も多いでしょう。ひとりのポカによって従業員全員が不便を食らう経験はあるあるだと思います。その積み重ねで不便な業務環境を強いられている界隈さえあります。これも固定観念があると「仕方ない」と思うかもしれませんが、そんなことはなくて、管理者や報告者の都合にすぎません。「従業員が社用車で事故を起こしたので今後社用車は一切廃止します。公共交通機関を使ってください」くらいに馬鹿げたことです。現代では消費者の目が厳しいため、気持ちはわからないでもないですが、だからといって安直な一律適用は思考停止でしかありません。

まとめ:

- 進捗は加算方式で管理するのが良い
  - できるだけ管理を少なくして、問題があるなら少しずつ増やしていく
- 進捗の管理は厳しくなりだちだが、その理由は人にある
  - 管理者と報告者
  - 人として、あるいは組織力学として自然な現象でもある
  - 緩めたいなら是正するか、そもそもそういう人を巻き込まない・発生させないようにする

質の管理

質の管理 とは、タスクそのものではなくエージェント（タスクを遂行する側の人や仕組み）やアウトプット（タスクによって生み出した何か）の「良さ」を管理することです。よく知られているのは品質管理です。

質の管理では「質」を定量的に定義、計測して、基準値を下回らないように監視することを目

指します。定量化が難しいものでも、人に尋ねて分類したり動向を見てその数や傾向を追うなどすれば定量化できますし、すぐに見つからない・わからない場合は頑張ってひねり出します。タスク管理からは外れますが、プロダクトマネジメントではまさにそのような指標として North Star Metric（北極星指標）を探しに行きます。ちゃんと管理するために、何とかして数字に落とし込むわけです。

質の管理は厳しく行うこと――特に安定させること がセオリーとなっています。事故やエラーが起こる率が「わかりません」は論外ですし、未来永劫ゼロですは無理にしても、できるだけ下げたいですよね。あるいはブランドなど付加価値を勝負する世界では高品質を担保せねばなりず、そのために高品質とは何かを定量に落とし込んで、客観的に判断できるようにすることを徹底させます。手間もお金もかかる営みですが、そうすることでようやく顧客に選ばれるサービスになります。

タスク管理との関わりですが、質の監視や向上といった活動がタスクとして存在する（**Q-Task** と呼ばせてください）、との対応になります。ここまで見てきたとおり、タスク管理は直接仕事にかかわるものばかりが重視されがちですから、この手のタスクはこぼれがちです。下手をするとシャドータスクになっていることもあります。前任者がいなくなった途端、ぼろぼろになる現象はよくありますが、それはその人がシャドータスクの Q-Task をたくさんこなしていたからだったりします。小説の話になりますが、近年ではまさにその立場から見たカタルシスを味わうジャンル（なろう系という一小説ジャンルのサブジャンル「追放系」）もあるほどです。

Q-Task のタスク管理の仕方ですが、まずタスクとしてしっかり認識させることです。シャドータスクであるならきちんと掬い上げて正式なタスクにするべきですし、他のタスクの最中に行う「ついで」や「一応手順ではあるが明確化されていないもの」であるなら、切り離して別途タスクにするか、明確化します。要は Q-Task を舐めないでください、ちゃんと正式にタスクとして尊重してくださいということです。

次に、可能なら以下も導入したいです。なお品質管理をご存知の方は、目新しいことは言ってませんのでスキップしてください。

- 言語化・定量化
  - 前述したとおり、そもそも質とは何かをきちんと定量化する
  - 定量化が難しい場合、少なくとも言語化はする
- 自動化
  - 人間による監視には限界があるので、機械やプログラムで自動で行えるようにする
  - 普段様子は自由に見れるようにしておくと、なおよい
  - 問題が起きそうなときだけ通知する（普段は気にしなくてもいい）、とさらによい
- 専任化
  - Q-Task を集中的に、あるいは重点的に担う専任者をつくる（アサインする）
- レビュー
  - 質についてちゃんとチェックする場（会議体）を設ける
- カジュアルな会話ができる関係
  - 質についての疑問や違和感をさらっと口にできる関係をつくる

    - あとで大事になるのを防ぎやすい

アプローチとしては二つあって、機械やプログラムによって徹底的に自動化するか、逆にレビューなどで人間の目を繰り返し通したりチーム全員の総意をもって通したりするか、の二択です。両方を採用することもできますが、リソース的に厳しく、傾向としてもどちらか一方に偏ると思います。もちろんこれらの活動もタスクとして尊重します。最後の会話については、タスク化というよりはそういう関係をつくるための余暇活動を業務時間に組み入れるイメージです（バッファとスラック）。Q-Task はかんたんに形骸化、あるいはシャドータスク化しやすいので注意してください。

最後に Q-Task の見直しも継続的に行ないます。特に手間暇が大きすぎたり、必要性が薄かったりすると形骸化しやすいです。そういう意味でも、上述したとおり専任化できるかどうかが鍵になります。品質管理部門、のような専用部門はありがちですが、対応が広く浅くなったり部門側がボトルネックになって形骸化したりなど中々難しいです。チームに質の専門家がいるのが理想ですね。ソフトウェア開発では「QAエンジニア」という、まさにそういう役割もあります。

もう一つ、質を確保する活動には向き・不向きが表れるので、向いてない人に強要しないことも何気に重要です。作家の世界でも質の保証は編集者が担ったりしますが、一般的には分業・分担が望ましいとされます。一方で、全く質を考慮しない人のフォローが大変なこともあり――など、このあたりは突き詰めると非常に奥が深いものとなっています。

まとめ:

- 質は厳しく管理して安定化させるのがセオリー
- その上でタスク管理するには:
    - 1: とにもかくにも、まずは Q-Task をタスクとしてちゃんと捉える
    - 2: 自動化か、人間レビューか、どちらかを採用する
        - 特に重要なのは専任者のアサインと、道中の活動もちゃんとタスク化すること
    - 3: 形骸化やシャドータスク化も起こり得るので継続的に見直す
- 向き不向きが激しいので、誰にどれだけやらせるかの見極めや調整もおそらく必要

プロセスの管理

プロセスの管理 とは、手順を管理することです。

タスク管理には「手順」や「手続き」として表れます。タスクの出来（前述でいう「質」もそうですね）を確保し、安定させるために、みんなこれこれの通りに作業してくださいね、とするのです。再現性という言い方をすることもあります。いつ誰が行っても同じ結果が出るようにしたいわけです。手続きは公式的な手順のことです。チームで決めたものはただの手順ですが、会社や業界や法律レベルで決まった手順は手続きと言えます。いずれにせよ本質は手順であり、指定された作業を指定された順番で行え、というものです。プロセスと手順は同義と考えて差し支えありません。

プロセスは典型的なシャドータスクになりがちです。たとえばタスクTの見積時間が3時間（3時間で終わらせてほしい）だとして、そのうち従うべきプロセスの部分で1.5時間食われるとしたら、Tの「プロセスに従う以外の活動」は1.5時間しか残っていないことになります。前者を**プロセス活動**、後者を**非プロセス活動**と呼ばせてください。つまりタスクはプロセス活動と非プロセス活動から成るわけですが、両者とも捕捉できていないと見積もりが甘くなります。どちらかがシャドータスクになりがちです。

もしTが難解なタスクで、「アイデアを形にするのに3時間くらいかかりそう」「形にした後、正式に採用してもらうまでの諸々のプロセスが1.5時間」だとしたら、Tの見積もりは4.5時間が正しいでしょう。これを3時間にしちゃうと、プロセス活動の1.5時間は外せませんから、アイデアを形にする非プロセスの部分を3 → 1.5時間に圧縮するしかなくなります。アイデアの質が落ちてしまうわけです。落としたくない場合、おそらくどこかで時間を捻出するでしょう（つまりシャドータスク化している）。これを防ぎたいなら、ちゃんと4.5時間と見積もるか、あるいはタスクTを「形にする」「プロセスに則って採用する」の二つに分けるべきです――と、これは非プロセス時間がシャドータスク化する例ですが、逆もあります。プロセスは慣れてくると麻痺してくるので、「自分達が使ってるプロセスには1.5時間かかる」という事実を忘れがちです。

ここまではシャドータスク化――本当は存在しているのに無いものとみなされてる（から課外でカバーする、場合によっては諦める）について見てきましたが、もちろん管理過多もあります。やり方なんて自由でいいのに無理やりプロセスを適用したり、プロセス自体は必要だけど細かすぎて無駄に時間がかかったり、あるいはプロセスの中に承認ステップがあるが承認者が忙しくてボトルネックになっていたりします。これらと全部向き合うのはバカらしいことが多いですから、やはりシャドータスク化します。プロセスという活動のそれなりを占める部分がタスク管理上は表れていないので、タスク管理全体が形骸化します。

では、どのように対処していけばいいのでしょうか。まずは状況を整理します。

- プロセス活動
  - 1: 必要な部分
  - 2: 過剰な部分
- 3: 非プロセス活動

1: 〜 3: の三つがあります。3: は非プロセス活動なので割愛します。

1: は必要なのでちゃんとタスク化します。上記の例で言えば、タスクTを二つに分けてます。あるいは、プロセスを包含するタスク側でプロセス活動分も考慮しても良いです。例で言えば、4.5時間で見積もってますよね。

2: は過剰なのでなくします。進捗の管理の項で加算方式を勧めましたが、似た発想をします。**要らないものはとりあえずやめて、それで様子を見てください。**問題が出たのなら、そのときにプロセスを追加すればいいのです。

私たちはビジネスの文脈だとなぜかやたらとプロセスを定めたがりますが、そんなことをしなくても

行えるタスクは案外多いです。プロセスを定めるのは再現性が欲しいからであり、昔の工業的大量生産時代の名残ではありますが、本当に再現性は必要でしょうか。個人に任せて、やり方も任せて、結果をチェックする、ではダメなのでしょうか。プロセス至上という固定観念を一度取っ払って、冷静に「このプロセスは本当に必要なのか」と自問自答してみると良いでしょう。

なお、プロセスの管理でも邪悪な管理者は顔を出します。管理者としては「そのとおりに則っているかを見ればいい」というかんたんなお仕事で済ませられるので、しがみつく人は多いのです。自分の邪悪さに自覚がないこともあります。プロセスは正しいのだから当然ではないか、というわけです。プロセス原理主義と呼ぶ人もします。法律でたとえてもわかりやすいと思います。法律は絶対的に正しいのだから当然でしょ、と。もっとも法律の場合は大まかな部分しか規定していませんし、個人で勝手に追加することもできませんが、プロセスは違います。管理者（あるいはその上位者）によってかんたんに追加できます。暴走化しやすいのです。自覚がないどころか、善意に基づいていることさえあります。地獄への道は善意で舗装されている、とはヨーロッパのことわざですが、まさにそうです。

プロセスの管理者の対処は、より困難を極めます。進捗の管理者と同様の難しさに加えて、上述のとおり法律のような信念もあるからです。事実上、不可能と言ってもいいと思います。できることは多くありません。管理者から離れるか、表向きは従うふりをするか、あるいは（より上位と組んだりチームで結託したりして）追放するかです。現代のセオリーでは管理の少ない仕組み（組織）を最初からつくる、かつそれを維持し続ける、が一つの解となっています。

まとめ:

- プロセスの管理はシャドータスク化しやすい
  - タスク = プロセス活動 + 非プロセス活動であり、どちらかがシャドーになる
- プロセス活動には必要な部分と過剰な部分があり、後者をいかに削れるかが重要
- 邪悪な管理者はここでも顔を出す
  - 法律を絶対視するかのような信念があり、なお手強い
  - 真面目に対処するなら、従うふりや追放など苛烈なものが必要

リソースの管理

リソースの管理 とは時間やお金を管理することです。活動には資源を消費しますが、資源には総量があるため使い切らないよう注視する必要があります。

タスク管理では「消費量を記録・申告させる」という形で表れます。タスクのページに「かかった時間」を書いておけとか、毎日どの仕事にどれだけ時間使ったか記録しておいて月一で提出しろとか、お金を使いたいならその前に承認依頼を出せ、残業したいなら事前に申請しろなどです。

これらもまたシャドータスク化、特にプロジェクトとは直接関係がないインダイレクトシャドーとみなされやすいです。要は時間にせよお金にせよ各個人が使うものなので、本人自身が把握して

計上もしてね、となりがちなのです。この計上はビジネスではコンプライアンスにも絡んできますし、個人であっても生活の生死に関わってきます。信頼を測るバロメーターとしても見られるので、インダイレクトシャドーと言えど無闇に削れないのです。どころか「必要だから」と正当化されがちです。

では、対処としてはどうすればいいでしょうか。ここまでの繰り返しまたは応用になりますが、以下のとおりです。

- ちゃんとタスク化する
  - お金や道具や設備など、仕事として必要になるものの申請はわかりやすい
  - スポットで必要になるかもしれないものについては、事前にタスク化して「プール」しておき、使うときに持ってくる
- バッファやスラックを設ける
  - 各個人、必要な記録や申告があればここで吸収する
- 記録会や申告会を開催する
  - 打ち合わせの一種として「リソースの記録や申告を行う場」を設ける
  - この場では皆が手を止めて、記録や申告の作業に集中する
  - メンバーで集まると「あれ忘れてない？」「あれもあるよね」など補完しあえるので、後々のバタバタを未然に防ぎやすい
- 専任化する
  - 名前をつけるならリソースマネージャー、あるいはリソースウォッチャー
  - プロジェクト内のリソース管理に目を光らせたり、必要な手続きを行ったりする専任者
  - 業務は苦手だが、こういうフォローや雑務は好きという人が適任
- 自動化・省力化する
  - 特に記録は、人間がちまちま行うのは全くもって非生産的な営みなので、できれば自動化したい
  - そうでなくともできるだけ手間がかからないよう、必要最小限のプロセスで済ませたい
  - いわゆるIT、もっといえばDXの出番
    - 楽に行えるようなツールを使う、設計する、何ならつくる

社内システムがあるから大丈夫、経理部が居るから大丈夫といった話ではないことに注意してください。むしろ個人やチームといった小さな単位において、リソース管理がシャドータスクとして猛威をふるいがちだから対処しましょうね、という話です。

この他、リソースの管理自体も減らせるなら減らしたいです。

時間については、本当にバカ真面目に記録する必要があるのか、など再考の余地があります。プロセスを変える権限がなくても、運用面で工夫することはできます。あまりにひどいとコンプライアンスに反してしまいますが、そもそも数字で厳密に管理すること自体に無理があります。ただ、表向きはそんな事は言えないので、裏でうまくやるのです。逆に管理者が邪悪だと悪用されたりもします（たとえば残業を申告させずにサービス残業にする）し、現状は悪用面の方が蔓延りがちだと思います。

お金については、申告のプロセスを減らせる可能性があります。お金が大事なのはそうですが、だからといって無闇にプロセスを複雑にするのは典型的な管理脳です。近年では色々なやり方が開拓され始めており、たとえば以下があります。

- 誰でも無条件に使えるが、誰がいつ何のために何円使おうとしたかは全部見える（透明性）
  - 皆に見えているので悪さはできませんし、してもすぐわかります
- 各社員や各チームに自由に使えるお金やカードを支給する、あるいは立替精算できるようにする
  - 各人が有償のツール、特に SaaS を使うのに重宝します
- 承認者をランダムにする
  - たとえば金額に応じて n 人分、チームメンバーの誰かがランダムに承認者になる等
  - 承認者が管理者ひとりに集中してボトルネックになるのを防げます

Rate Limit（使える上限を設定しておく）と併用すると、なお効果的です。銀行でも上限引き出し額の設定ができますし、世間で話題の ChatGPT でも、プログラマー用の API は使った分だけ課金するモデルなのですが、誤って使いすぎて破滅しないよう課金額の上限を設定できるようになっています。プログラムはループ構造をつくって一秒に何万回リクエストする、なんてこともかんたんにできますので、トークン（使用時に使うパスワードなるもの）が漏れると悪用されて使われまくります。「40000 ドル請求が来てるんだけど……」なんて事故が起こり得るのです。しかし上限が設定されていると、上限以降はエラーになってそれ以上進まないので事故は起こりません。最近ではさらに改善されて事前購入制（50ドル買ったとしたら50ドル分しか使えない、それ以上は再び購入が必要）になりました。

と、少し話が逸れましたが、工夫は意外とできるのです。リソース管理というと、時間やお金を扱うという性質上、つい絶対視して過剰に管理しがちですが、本当にそこまで必要なのかを見直して、上手く融通を利かせてください。

まとめ:

- リソースの管理は、時間やお金の記録や申告という形でシャドータスク化する
- 対処としては、ここまで挙げてきた内容の応用
  - シャドータスクをちゃんとタスク化する
  - バッファやスラックを設けて個別に対処
  - 専任化、自動化と省力化 etc
  - また記録会や申告会など、リソース管理用の場を設けるのもアリ
- そもそも管理自体も減らしたい
  - 時間については、記録の手間を減らしたい
  - お金については、申告の手間を減らしたい
  - いずれにせよやり方は色々ある

専有の管理

専有の管理 とは人を人質のように管理することです。別の言い方をすると「物として所有する」とも言えます。物は全面的に所有者のものであり、物の事情は考えません。そもそも物に意思などありません。物は所有者がいつでも好き勝手できるように、所有者の手元に置いておくと便利です――と、このような発想を人に適用したのが専有です。私物化と言っても差し支えありません。

私たちは何かと専有されがちです。私生活でも束縛がきつい、門限、過保護といった言葉がありますし、仕事では言わずもがな会社が従業員を専有します。この専有ですが、さすがに現代では本当に物のように、たとえば奴隷のように扱われることは通常ないでしょう。代わりに規則と場をつくって、それに従わせようとします。**専有の特徴は拘束であり、時間と場所を拘束してきます**。典型的なサラリーマンやアルバイトには就業時間がありますし、店舗やオフィスへの出社も義務付けられます。何を当たり前のことを……、と思うかもしれませんが、そもそも拘束とは絶対的に必要なものでしょうか。答えはノーです。特に最近ではパンデミックの影響でリモートワークが加速し、出社しなくて仕事が通用することは広く知られてきました。現地での対応が必要な仕事もまだまだ多いですが、今後技術や方法論の発展につれて減ってくるでしょう。

専有とは拘束であり、拘束にはコストがかかりますが、タスク管理ではここが計上されません。つまり専有に絡む行動がタスク化されておらず、しかし必要なことなので各自対応している――というわけで、ここもシャドータスク化しています。通勤とサービス残業の話はすでに述べましたが、通勤とはまさに専有を満たすためのシャドータスクです。

タスク管理が上手くいかない、特に思ったように消化されないときは、実は専有が足を引っ張っていることも多いです。あえて強調しますが、**専有は思っている以上に足を引っ張っています**。通勤はわかりやすいですが、他にもあります。道具や設備が一つしかなくてそこに集まらないといけないとか、定例会議や重要イベントが設定されていて事実上集まらないと排斥されるなど、拘束が求められるシチュエーションはそこかしこに転がっています。一つ一つは軽微でも、積み重なるとかなりの負担になります。拘束という言葉がわかりづらければ「予定」と言い換えても構いません。仕事の名目でやたら拘束されるという現実は、いわばやたら予定を組まれることにも等しい。日常生活で言えば一日に何人もの人と会ったり、旅行のスケジュールを過密にするようなものです。移動時間だけでも相当費やしていますし、何より疲れますよね。それと同じことが平然とまかり通っています。さらに厄介なのは人が社会的動物であり、一緒の場を過ごすだけで満足してしまう生物であることでしょう。要は仕事した気になりやすいのです。専有のコストがかなりかかっているにもかかわらず。

では、専有にはどう対処すればいいでしょうか。

まずはシャドータスクの対処と同様、キャパシティベースによる無理のない見積もりと、バッファとスラックによる余裕の確保は使えます。

しかし、専有は思っているよりも蔓延っており放置していると辛いですし、勝手にはなくならないので、意識的に減らしていくことこそが重要です。そのためには「拘束せずに働くには？」「拘束せずにタスクをこなすには？」を考えなくてはなりません。拘束に抗うあり方を **脱拘束** と呼ばせ

てください。専有に対処するには脱拘束を目指したい。

では脱拘束には何が必要でしょうか。方法論とツールです。たとえばリモートワークも、それを行うためのやり方と道具がなければやりようがありません。知識を知らないと始まらないのと一緒で、無知の中で自ら編み出すのは現実的ではありません（そもそもそんなことにたっぷり費やせる時間がありません、ないしは与えられません）。脱拘束は、これだけで本を何冊も書けてしまうほどボリューミーなものですが、いくつか紹介します。

- 非同期コミュニケーション
  - お互いがお互いの時間を専有して、リアルタイムにやりとりするコミュニケーションは「同期的」
  - 一方、手紙やメールやチャットなど、各自のペースで読み書きするコミュニケーションは「非同期的」
  - 脱拘束には後者の非同期的なやり方が必要になります。**脱拘束は非同期的なやり方をどれだけ採用できるかがすべてです**
  - 現代ではまだまだ同期的なやり方が主流であり、「リアルで顔を合わせて喋れないと仕事にならないでしょ」と考える人も多いです、それほどに難しい
- QWIC
  - 非同期コミュニケーションの主なあり方の総称
  - Q&A、Wiki、Issues、Chatの略です
  - Chat（チャット）については Teams など、リモートワークの必須ツールとしてご存知の方も多いと思います
  - Wiki（ウィキ）は Wikipedia やゲーム攻略ウィキなどで知っている人も多そうですが、チャットよりも情報を整理・検索しやすいです
  - Issues（イシュー）は GitHub Issues などのことで、1 話題 1 ページで話題ごとに独立して議論を行うスタイルに適しています
  - Q&A は Q&A サイトのことで、Yahoo!知恵袋、教えてGoo、Stack OverFlow や Quora など Web サービスとしてはよく知られており、質疑応答スタイルのコミュニケーションに特化します
- Communication Injection/コミュニケーションの注入
  - 私達は人間なので非同期的なやり方だけだと足りません。欲求不満で耐えられませんし、信頼関係や親近感も育みにくい
  - なので、脱拘束とはいっても、従来の同期的なコミュニケーションも意識的に差し込む必要があります
  - つまり、以下のように考えます
    - □同期コミュニケーションのみ
    - □同期コミュニケーションがメイン、非同期はサブ
    - □非同期コミュニケーションがメイン、同期はサブ
  - 非同期的なやり方がメインという前提で、同期的なコミュニケーションを **必要に応じて注入する** と考えます
- 個人タスク管理
  - 脱拘束が進行すると、各人には自律性が求められます

        - 言われないと動けない人や、場の雰囲気に流されることを生業とする人には務まりません
    - 気合で何とかなるものではなく、やり方や考え方の工夫が必要です
    - 実はこれは仕事から私生活まで、個人の行動をいかに上手くやるか――個人タスク管理と同義です
        - 余談ですが、本書はまさにこの個人タスク管理を扱おうとしています

本書でも何度か言及したDX――デジタルトランスフォーメーションは、私達のあり方をデジタルの流儀に合わせるというものです。同様に、**脱拘束についても、私達のあり方を非同期コミュニケーションに合わせます**。そこまでしなければ脱拘束、もっと言えば専有の軽減はできません。しかし、これができたら、専有にかかっていたコストがかなり浮きます。実際にリモートワークができている企業では、各人はライフをしっかりと担保しつつワークも行えていると思います。それで業績を伸ばしている企業も少なくありません。

一方で、GAFAMなど名だたる企業でも出社に回帰する動きは珍しくないように、脱拘束は取り組み続けるのが難しい営みでもあります。そもそも方法論からして、あまり盛り上がっていません。上記のQWICとCommunication Injectionは筆者の造語です。筆者が造語できてしまうほどに未開拓な領域なのです。もちろん、世界中で全くないかと言われるとそうでもなく、高度に脱拘束できている組織やチームも珍しくはありません。ただ一般層だったり大きな組織への適用だったりという文脈では、正直まだまだだなという印象です。

筆者としては生成AIが台風の目になると考えています。非同期コミュニケーションは「読み書き」が大変ですが、生成AIは要約や翻訳が得意なのでカバーできます。自分のコンテンツを学習させたり、音声や映像もつくれたりするので、将来的には疑似人格もつくれるでしょう。Aさんと打ち合わせするのではなく、Aさんの AI 疑似人格と打ち合わせをする、のような未来もありえると思っています。よく「人は人と直接会って話したい生き物なので同期コミュニケーションはなくならない」という人がいますが、別にゼロにする必要はなくて、適度に注入すれば事足ります。それよりも私たちは自分の生活（ライフ）が大事であり、すでにライフを重視・尊重する潮流は来ています。何なら水準も上がっています。特にパンデミックによるリモートワークがこれを後押ししました。そもそもライフのあり方は人それぞれですから、専有とは合わず、非同期コミュニーションは必然なのです。奴隷が当たり前だった時代はなくなりました。現代は奴隷ほどひどくはありませんが、拘束という意味ではひどいままです。今は単に手段がないから同期的なやり方に頼っているだけですが、上記でも軽く触れたように手段はもうあります。この先、手段がさらに整備・啓蒙されれば、非同期にシフトしていくことは避けられないと思います。大胆に言えば、現代人が拘束から解放される動きがこれからおこっていくと思っています。

と、脱拘束の話が盛り上がってしまいましたが、お許しください。専有の軽減には脱拘束が必要ですが、脱拘束という考え方はまだまだ難しい（というより自覚や馴染みがなくてとっつきづらい）ため、意図的に紙面を割かせていただきました。他にも専有を軽減するあり方や、そもそも専有を容認する潮流もありえるかもしれませんが、本書の範囲を越えるため割愛します。

まとめ:

- 私たちは専有を管理されている――時間と場所を拘束されがちである
- 専有には思っている以上に足を引っ張られているが、タスク管理上は出てこないので気づきにくい
- 専有に対処するにはシャドータスクの対策が有効だが、それ以上に専有自体を減らす脱拘束が必要
- 脱拘束という営みは難しい
    - 難しいが、できないことはなくて、手段の目処はついているし事例もある
    - 特にリモートワークは脱拘束の好例
    - 筆者の予想（そして願望でもあるのですが）では、今後盛り上がることは避けられないと考える

まとめ

- プロジェクトタスク管理の本質は 3A
    - Assign アサイン、人にタスクを割り当てる
    - Adjust 調整、タスクの属性値を変える
    - Alternate 全体と詳細、タスク全体の状況を眺める（俯瞰）と特定のタスクの詳細を見る（注視）の行き来
    - これらを上手くやれればそれでいいし、やれるなら別に管理は要らない
- 俯瞰と注視をどれだけ行えるかはツールの能力であり、段階がある
    - Alternate モデル、計 4 段階
    - タスク管理ツールを使う段階はレベル 3 であり、キャズム（深い溝）が存在するほど難しい
    - つまりプロジェクトタスク管理は成立そのものが難しい営みである
        - 実際に結局成り行き任せのコミュニケーション過多で何とか乗り切るという原始的なスタイルになりがち
        - DX と同様でツールをちゃんと使う、そのやり方や考え方に自分たちが合わせるといった姿勢のシフトがそもそも必要
- 俯瞰を行うためのビューにも種類があり、一長一短である
    - リスト（一覧）、テーブル（表）、ボード、チャート、ネットワーク、センテンス（文章）など
    - 後ろに行くほど後発だが学習コストも高い、またネットワークとセンテンスは事例自体が無いか少ない
    - 現状無難なのはテーブルであり、まずはテーブルで上手くまわすことを目指したい
- タスク管理に計上されない隠れタスク「シャドータスク」が存在する
    - 対処としてはキャパシティベース、バッファとスラック、シャドータスクをちゃんとタスク化して管理する等がある
- プロジェクトタスク管理は、実は周辺の要素の管理もしてしまっている
    - 進捗、質、プロセス、リソース、専有
    - しかしこれらはタスク管理としては計上されない（シャドータスク化している）

- 管理も過ぎれば毒、意識的に減らしていく努力が必要

=== Chapter-5 パートナータスク管理 ===

本書ではパートナータスク管理について軽く取り上げます。前章と同様、筆者なりに整理することで、読者が何らか気付きやヒントを得ることを目指します。

パートナータスク管理

パートナータスク管理 とは、パートナーと共同で行うタスク管理を指します。パートナーの定義は深追いしませんが、ニュアンスとしては夫婦や親子などです。「恒常的に一緒に暮らしている」「親密な」「少人数の」間で使われるタスク管理、とでも言えるでしょう。筆者は「家族」、特に夫婦やカップルなど 2 人組をイメージしていますが、同棲やシェアメイト、あるいは子供を含めたりポリアモリーだったりと 3 人以上のあり方にも当てはまるかもしれません。本章では以後パートナーを「2 人組」のニュアンスで使いますが、適宜読み替えてください。

パートナータスク管理で扱うタスクは日常生活全般です。家事や育児は典型的なタスクですし、引っ越しや旅行といったイベントの際にはどちらが何をするか、いつまでにするかといった分担も行うでしょう。もしかすると同じ会社に勤めるとか自営業など同じ仕事をしているケースもあるかもしれませんが、それはプロジェクトタスク管理の範疇であり本章の対象外です。パートナータスク管理は、あくまでも日常生活の延長でタスクを管理します。

ファミリア・デバフ

そもそもパートナータスク管理は成立からして難しいところがあります。

ここまで見てきたとおり、タスク管理は面倒くさいものですし、やり方に考え方にツールにと要求するスキルとリテラシーも多いです。誰もができることはでありませんし、何ならできる人であってもやりたくないのが本音です。一方で、厄介なことにパートナー相手であれば怠惰は許されます。遠慮は要りませんし、そもそもそういう間柄だからこそパートナーになれているのです。

したがって、タスク管理に必要な「面倒くさい諸活動」はパートナー相手だとサボりがちになり、そもそも成立しづらいということが起きます。自分の仕事や専門性をパートナーに見せない人は多いと思いますが、それは親しい相手だと高度なトピックスや面倒くさい事柄が通じにくいと体感的にわかっているからです。これを筆者は **ファミリア・デバフ** と呼んでいます。親しい間柄にかかるデバフ（弱化）です。

ツールの変遷

パートナータスク管理で用いる手段――ツールはどのように変遷してきたのでしょうか。たしかな研究はないと思いますが、筆者の考察を述べます。

元々は口頭と電話でした。基本的に同座しているときに口頭で要件を伝えつつ、同座していない場合はスポットで電話をするという原始的なスタイルです。これは誰でも自然に取れるやり方ですが、ただのコミュニケーションであってタスク管理ではありません。当然ながら管理できることも非常に限られます。言った言わなかったが多発しますし、何でもかんでも伝えると鬱陶しくて、喧嘩するか、あるいは伝える側が我慢することになります。

次にポケベルやメールなど、テキストメッセージをやり取りする手段も台頭してきましたが、これも電話と本質的には変わりません。そもそも IT 音痴の人は文字入力ができず意味がありません。

流れを変えたのは LINE とスマホの存在だと筆者は見ています。メールよりもはるかに素早くメッセージを交換でき、スタンプにて高速かつ多様なリアクションも行えました。またスマホの普及により時代としても IT が使える人が増えてきて、手段として日常に入り込んできました。LINE を使うことで、タスクに関するやりとりも色々と行いやすくなったのです――とはいえ限界があります。たとえば、LINE を「自分のペースで好き勝手に送ってもいい」「各自が空いた時に見て返事すればいい」手段として使える人、つまり非同期的な手段として使える人は意外と少なくて、通知が来たらすぐ反応する人の方が多いでしょう。特に現代人は SNS の影響で通知に反応する体質になっていますし、そもそも非同期的な考え方は（前項「専有の管理」でも見たとおり）高度なものです。結局、これも口頭や電話の域を越えられません。

口頭と電話、メール、LINE。どれもコミュニケーションツールですよね。コミュニケーションツールだけでタスク管理を行うのは難しいのです。どうすればいいのでしょうか。

ライフハッカー、エンジニア、デザイナーなど高度なスキルを持つ夫婦が試行錯誤を始めます。本格的に目をつけられたのは Slack だと思っています。Slack は Microsoft Teams のような「ビジネスチャット」の一種ですが、Teams よりも先発で、元々はエンジニア向けの軽量でカスタマイズ豊富なチャットツールでした。LINE と異なるのは、チャンネルという形で用途別の部屋を自在につくれることやメッセージの検索が行いやすいこと（そもそも SLACK は Searchable Log of All Conversation and Knowledge の略です）、また連携機能により色んなサービスからの情報を流し込めて情報ステーションにできることです。リマインダーやブックマークといった機能もあります。タスク管理のポテンシャルがありそうです。特に情報の保存ややり取りに便利で、たとえば旅行したいなら旅行計画用のチャンネルをつくって、そこで調べ事を共有したりすればいいわけです。このようなあり方は **夫婦 Slack** や **カップル Slack** と呼ばれたりしました。また、あまりに便利なのでひとりで使う **ひとり Slcak** もありました。同様の使い方は Discord でもできます。Slack との違いはボイスチャットの品質ときめ細やかな通知だそうです。

そうしているうちに、いつの間にか専用のアプリも登場するようになりました。

- TimeTree
  - 予定をつくって、共有してコミュニケーションできるアプリです
- Cross
  - 夫婦のためのToDo共有アプリ

これらアプリはパートナータスク管理の本質を突いています。たとえば「カレンダーで俯瞰できる」「個別のタスクにアクセスできる」「個別のタスク上でチャットができる」などです。使いやすさが確保されていますし、パートナーの営みでもある「やりとり」の要素もあります。それもタスク単位で行えるので話題が混線しません（LINE だと場所が一つしかないので混ざりますよね）。タスク管理を知らない人でも比較的使いやすく、役に立つのです。

パートナータスク管理の3C

ここまででパートナータスク管理の本質が朧気ながら見えてきたと思います。改めて整理します。

本質は 3C で捉えることができます。

名前	解説
Chat per Task	タスクごとのチャット（やり取り）
Calendar View	カレンダーによる俯瞰
Concept	細かいコンセプトで差別化

タスクごとのやり取り

すでに述べたとおり、パートナータスク管理には「パートナーなので気軽にやりとりできる」「でもやりとりだけだとタスク管理できない」「かといって遠慮も要らないし仕事でもないので、いちいち面倒くさいタスク管理なんて基本的にしたくない」との微妙な機微が存在します。

これを満たせるのが **Chat per Task** で、タスクごとにチャット（などやりとりを行う場）を設ける、というバランスです。

タスクの粒度――細かさはパートナー間で決めればいいでしょう。たとえば修理した時計を引き取ってほしい場合、

- 粒度が細かい場合
  - 「時計を引き取りに行く」タスクを新たにつくる
- 多少粗い場合

- 「2024/05/20の週」タスクでもつくっておいて今週はここを見るようにしておいたとする
- その前提で「時計できたみたいなんで取りに行ってくんない？」などとチャットを撃つ

などとなるでしょう。最も粗いのが LINE のように「場が一つしかない」パターンですが、これだと日常の複数のタスクを扱いづらいですし、過去の情報も探しづらいです。かといって、用件一つごとにタスクを新しくつくるのもやりづらいかもしれません。どの程度の粒度がいいかは対話しながら探りましょう。

カレンダーによる俯瞰

タスク管理のタの字を知らない人でも、カレンダーなら知っていると思います。とても馴染みのある概念です。だからこそわかりやすくて、使いやすい。

仕事でデジタルツールのカレンダーを使っている方は多いでしょうが、パートナータスク管理でも使えます。二人共用のカレンダーがあって、ここに共同で予定を書き込むのです。スマホから使えるのでいつでも見れます。自分の予定が見えるのはもちろん、パートナーに予定を見せることもできます。お互い何をしているか（するつもりなのか）がわかれば、先回りして配慮できます。**タスクの可視化は、思いやりのためにも重要です**。あるいは「外せない時間があるなら予定として入れておけ」「入れてなかったら割り込めると解釈するからな」とルールを決めてもいいでしょう。ビジネスライクに感じるかもしれませんが、親しき仲にも礼儀ありともいいます。礼儀を確保する端的なやり方はルールをつくることです。

細かいコンセプトによる差別化

パートナータスク管理の正解は一つではありません。理想は自分達で模索することですが、仕事ではないゆえにそこまでの努力はしづらいでしょう――というわけで、パートナータスク管理ツールの出番です。ツール開発者は「私たちのツールはこういうコンセプトですよ」とアピールをして利用者を集めます。利用者としては、自分達に合ったものと出会えれば良いですね。

いくつか例を出すと、TimeTree は「予定の共有と相談」を謳っておりカレンダーにフォーカスしています。予定が多いパートナーや、カレンダーに慣れている人には助かりそうです。Cross は ToDo 共有アプリを謳っており、カテゴリやリマインダーや期限などタスク管理としての機能が強かったり、ライフイベントに備えた「やることセット」も拡充しています。（カレンダーもありますが）こちらはタスク単位で捉えたい方には使いやすそうです。家事の名はやYietoのように家事を見える化して（家事と）戦いやすくするものもあります。

その他にも色んなアプリがあります。現時点でパートナータスク管理なるジャンルは存在しませんが、共有アプリや家事分担アプリという言い方はすでに見られます。潜在的にもニーズは多そうですし、今後も盛り上がる余地はあるでしょう。筆者としては、アプリ好きなどマニアックな人以外のマジョリティでもかんたんに使えるデザインを上手く実現できたら、一気に広まる気がします。

盤外戦の重要性

盤外戦

盤外戦 とはタスク管理以外の周辺で何らかの働きかけを行うことを指します。

たとえば「そもそもなんでこんなにタスクが多いんだっけ？」と考えてみたり、タスクを増やしている原因を潰しに行ったりするなどが当てはまります。パートナータスク管理で言えば「話し合いの時間を設ける」が最もわかりやすいでしょう。要は、目の前にあるタスクに取り組むことのみを正義とするのではなく、タスクそのものを疑います。

パートナータスク管理は盤外戦にかかっている

パートナータスク管理は、というよりタスク管理全般がそうなのですが、実は盤外戦にかかっています。盤外戦を行わないと、日々忙しいタスクに忙殺されるだけで終わってしまいます。偶然状況が好転すれば救われますが、そうでなければ最悪壊れてしまいます。昔は精神論や根性論が美徳とされていましたが、もうそんな時代でもありません。

しかし、私たち人間には怠け者であり「目の前のタスクと向き合う方が楽だしなぁ」と思考停止しがちな心理があります。パートナー相手となれば遠慮も要りませんし、仕事じゃないので怠惰も正当化しやすいです。ファミリア・デバフはすでに述べたとおりです。もちろん、お互いに良い年をした大人であり、それなりに忙しくもある――というわけで、忙殺に耐えるシチュエーションはよく起きます。**本当によく起きます**。前章では反応的状況としてその一部を解説しました。だからこそ、意識的に盤外戦を行っていかなくてはなりません。

盤外戦を取り入れるには

では盤外戦をどのように始めればいいんでしょうか。

答えは単純で、**盤外戦用のタスクを管理します**。最も単純なのは思考や対話の時間をつくることで、パートナーに決して割り込まれない・割り込ませない「ひとりになれる時間」をタスクとして確保するとか、二人で話し合う時間を確保する等です。すでに諸悪の根源が見えているのなら、それを潰すためのタスクをつくっても良いでしょう。重要なのは、なあなあやテキトーにせず、**タスクや予定としてきちんと定義すること** です。私たちは外から降ってきたタスクは比較的素直に処理しますが、自分たち主導で何かを行おうとなったときには途端に腰が重くなります。だからといって腰を上げないと盤外戦はできないので、上げないといけません。とにもかくにも、まずはここからです。

以下のマトリックスを見てください。スタートラインに立つために時間の捻出が必要ですが、読者

でパートナータスク管理を行ないたい・行っている方は、どこを確保できるでしょうか。あるいは確保したいでしょうか。

	ひとりで考える時間	二人で話し合う時間
週に1時間	1	2
数日に1時間	3	4

最低ラインは週に1時間です。まずはここを目指してください。ここさえもできないという場合、そもそもそんなビジーな状況（あるいはパートナーとすり合わせられない状況）が論外ですので、何とかして捻出してください。事情は人それぞれですし、特にビジーな当人からすれば苛立つかもしれませんが、盤外戦をしたいのであれば時間の捻出は必要です。捻出するかどうか、それだけです。

ひとりで考えるか、二人で話し合うかについてもパートナー次第でしょう。現代は対等を重んじる価値観に寄っていると思いますので、話し合う方を目指したいかもしれません。一方で、どちらか片方に考えてもらって、他方はそれに従ったりコメントしてフィードバックしたりといった補完関係もアリです。また盤外戦の内容によっても変わってくるでしょう。

イメージとしてはデートあるいは会議です。ひとりの場合はひとりカラオケ、ライブ、整体やジムやエステといった通い事を思い浮かべてください。盤外戦にしても同じことをします。カジュアルに寄せるか、ビジネスライクに寄せるかも人次第です。前者に寄せすぎて思考や対話がおろそかになるのもいけませんし、後者に寄せすぎて盤外戦タスク自体が苦痛になってもいけません。楽しく、長く続けられるようなバランスを開拓したいところです。

そうして時間を捻出できたら、次は何をするかです。対処すべき事項が見えているならそれをすればいいですが、見えていないのなら、まずは見える化から始めます。方法は色々ありますが、アイデア出しや反省会、あるいは相談のようなテイストになります。いくつか挙げます。

- フリーライティング
  - 思いついたことをそのまま文章で書き殴ります
  - ひとり用です
- スキャン＆ライティング
  - 部屋を隅々まで順に見て回る、月曜日の朝から日曜日の夜まですべての時間帯を思い浮かべる、一日の行動範囲を地図ベースで全部辿ってみる、など「スキャン」をします
  - その過程で思いついたことは何でも書きます
  - ひとりで行うか、各人で行ったものを持ち寄るのが良いです
- トピックトーク
  - 1:「～～について～～と考えている」のような話題を一つ決めます
  - 2: その話題について話し合います
  - 3: きりのいいところでクロージングします
    - クロージングとは「次は(誰)が(何)をする」か「保留」のどちらかを決定することです

    - 1〜3 をぐるぐるまわします
    - 二人で行ないます。話題決めは交互に行ってもいいですし、思いつかないなら片方が連続して出しても構いません
- ベストスリー
    - お互いに「言いたいことベスト3」を持ち寄り、交互に一つずつ出して話し合います
- オフィスアワー
    - 片方が「なんでも相談していいよ」時間をつくり、他方が訪問するという体での相談会
    - 元は大学の先生が学生に対して行うものですが、パートナーでも使えます
    - 相談役と先生役など、大げさなロールプレイを心がけると頭の使い方が変わって捗ります
- ディープハグトーク
    - ベッドなどでぎゅっと抱き合いながら、盤外戦に関する話をします
    - 深く抱き合う（ディープハグ）がポイントで、オキシトシンの放出、体温と鼓動の同期、相手の存在を肌で感じることによる配慮の意識、といった作用により落ち着いた対話が可能だと考えます
    - 服か裸かは問いませんが、ムラムラしては本末転倒です。また臭いなど不快感が勝ると集中できないので、必要なら事前に風呂に入るなどの準備もはさみます

と、横文字のいかにもな方法を挙げましたが、方法は問いません。大事なのは盤外戦について建設的に考えたり話し合ったりすることです。何もせずできるならそれでいいですし、できないのならやり方なり刺激のかけ方なりを工夫して何とかして集中して望めるようにしてください。

トーカビリティ

ここまで本章を眺めてきて、「そもそもそんなじっくり話し合えたら苦労しない」と考えた方がいらっしゃるかもしれません。これはそのとおりで、身も蓋もないことを言うと**そもそもパートナータスク管理を行えるだけの素養を持つ人の方が少ない**です。

一つの目安としては、ちゃんと対話を行うための時間を定期的に設けられるか。また、それを受け入れ、実施できるだけの素養があるかどうかにかかっています。これを**トーカビリティ（Talkability）**と呼ばせてください。トーカビリティが高いなら可能ですし、低いなら難しいでしょう。そういう意味では、パートナータスク管理を行ないたいなら**そもそもトーカビリティの高いパートナーであることが必要条件である**とも言えます。

トーカビリティは仕事の場だと高くなり、私生活の場だと低くなる傾向があります。オンとオフを切り替えている人も多いですよね。つまりトーカビリティが低い人は、家庭だから低いだけであって、潜在的には高い可能性があります。この場合、私生活においてビジネスライクなやり方を意図的に採用することで、その潜在性を引き出せます。

またトーカビリティは問題ないが、余裕がなさすぎて話し合える機会を持たないケースもあります。この場合、ご自身でパートナーの負担を軽減させて余裕をつくってしまうことで打開できるこ

とがあります。そうでなくとも、まずは自分ひとりで盤外戦を行うことはできます。パートナーをトーカブルにする、という盤外戦を行えば良いのです。

とはいえ、すでに述べたとおりファミリア・デバフがありますので、思っている以上に手強いことは覚悟してください。正直できたらラッキー、できなくて当たり前くらいの低確率だと思います。

まとめ

- パートナー間で行うタスク管理、という微妙なバランスがある
- 本質は 3C で捉えられる
  - Chat per Task、タスクごとのチャット（やり取り）
  - Calendar View、カレンダーによる俯瞰
  - Concept、細かいコンセプトで差別化
  - 特にコンセプトは様々で、まだ物好きが使う程度のジャンルだが、今後キラーアプリが出るかもしれない
- 実はタスク管理よりも盤外戦の方が重要
  - そうしないと忙殺から脱せない
- パートナータスク管理はそもそもハードルが高い営み
  - ファミリア・デバフが存在しており、ただでさえ通じにくい
  - 鍵はトーカビリティ（ちゃんと対話を行える素養）が有ること
    - トーカビリティがない場合、おそらく成立しない
    - ただし「潜在的には高いが私生活では低い」パターンがあり、この場合は工夫次第で高くできるかもしれない

=== Chapter-6 □第2部「個人タスク管理 基礎」===

個人タスク管理の基礎を扱います。

=== Chapter-7 タスク管理の戦略 ===

タスク管理の戦略とは、どんなやり方や考え方で望むかという方針のことです。これは「何を拠り所にするか」で分けることができます。眺めればわかるとおり、戦略によってだいぶあり方が異なります。自分に合ったものを使っていくことが肝心です。また手札が多いと応用範囲も広くなりますので、余力があれば馴染みのない戦略を練習してみるのも良いでしょう。

本章では各戦略を詳しく取り上げていきます。ツールやメソッドは参考文献をご覧ください。本章ではあくまでも考え方を述べます。

時間で区切る

時間をどのように区切るか、各区域をどう使うかを考える系の戦略です。メリハリをつけるのに適しています。

Dailer

概要

タスクを「日単位」で捉える戦略です。

特に「今日」と「それ以外」に二分し、今日のタスクが全部終わったら今日はもうおしまいにします。そうすることでエンドレスを防いでメリハリをつけられますし、今日やらなくていいものを明日以降に回すことで先送りもできます。明日になったら、その日を今日とみなして同じことをします。

このような概念は **デイリー（Daily）** と呼ばれます。筆者は **デイリーエリア** と呼んでいます。今日やるタスクを集めたデイリータスクリストの他、情報全般に適用したデイリーページやデイリーノートもあります。ソシャゲなどゲームでもデイリークエストなんてものもあったりします。結構身近な概念で、触れたことがある方も多いかもしれません。

別の言い方をすれば、今日というエリアがあって、そこに今日扱うものを放り込むイメージです。できれば全部処理したいですが、無理に完遂する必要もありません。あくまで目安です。なので、今日はもうだるいのでこれは明日でいいや、と明日にまわしてもいいです。もちろん状況次第では余談を許さないこともありますが、人間は所詮脆弱で怠惰な生き物にすぎません。たかが知れています。日単位で毎日仕切り直す、くらいの緩さで良いのです。それだけでも、何も区切らなかった頃と比べると驚くほど生産性が出ます。日という単位は睡眠をはさんだ最小単位でもあって「きりのいい単位」であり、使い勝手は抜群なのです。

応用：エリアの分割

どれだけ分けるかに好みが出ます。

最も単純なのは「今日」と「明日以降」の二分です。何したかのログ（記録）も残したいなら「今日」「明日以降」「昨日以前（のログ）」の 3 つになるでしょう。律儀に分けると「今日」「明日」「明後日」「しあさって」――とカレンダーのように日単位になります。それぞれ **ツイン**、**トリプ**

ル、インフィニティ（無限）と呼ばせてください。

分割数が少ないほど運用がシンプルですが、「今日以降」の方に溜まりやすくなります。TODOリストは肥大化することで有名ですが、特にツインやトリプルだと「今日以降」が肥大化しがちです。これを防ぎたければ、インフィニティが良いでしょう。先送り先を具体的に指定できるようになります。「3日後くらいにするか」、「一週間後でいいか」、「しばらく用はがないが忘れたくはないので、そうだなぁ、20日後くらいに」など柔軟にできます。たとえば今日が 2024/05/01 なら、それぞれ 05/04、05/08、05/21 のエリアに置いておけばいいからです。ただし、分割数が多いほど先送り先の判断にエネルギーを使うようになります（つまり面倒くさく感じます）。

インフィニティが面倒と感じるなら、もう少し減らしてもいいでしょう。「今日」「明日」「今週」「今月」「それ以降」の 5 分割がいいかもしれません。これを **クインタプル** と呼ばせてください――と、これらは一例で、他にも色んな分け方があります。ご自身にあった分け方を模索してみるのも良いかもしれません。

ちなみに、トリプルで取り上げましたが、ログについても好みが分かれます。自分が何したかを振り返りたい場合は、確保した方が良いでしょう。といっても、昨日以前に使ったデイリーエリアをそのまま残しておくだけです。一方で、記録を残したり振り返ったりするのが面倒くさいか、要らない場合は無視しても構いません。邪魔になるなら消したり捨てたりしてもいいでしょう（特に紙や手帳などアナログな手段で行う場合）。履歴や日記としての面白さもあるので、筆者としてはぜひ残すことをおすすめしたいです。

メリデメ

メリット:

- 日単位で区切りをつけることによる恩恵
  - メリハリをつけられる（今日はここで終わり、ができる）
  - 先送りしやすい
  - 日単位でログが残り、振り返りしやすい

デメリット:

- 軽微なメンテナンスが毎日必要になる
  - 今日エリアの切り替え、内容の転記など
- 先送り時やメンテナンス時の意思決定コストが高く、認知資源を食いやすい
  - 特に毎日、その日の始まりに「昨日のデイリーエリアと今日以降のエリアを見て」「今日エリアに何を残すかを決める」との判断

適性

作家、クリエイター、フリーランスなど予定が少なく裁量が大きい人に適しています。このような人達はまとまった時間を使って何をするかを全面的に自分で決めなくてはなりませんし、インプッ

トやアイデアの管理や成果物の振り返りといったメンテナンスも必要になります。Dailer の戦略を使うと、日単位で何やるかを決めたり振り返ったりできるので、何かとおさまりが良くなります。融通も利きやすいので、気分や状況のイレギュラーにも対応しやすいです。

カレンダーが手放せない予定駆動型の人には適していません。一見すると Dailer に見えますが、**Dailer はタスクへの取り組み方に裁量があることが前提** です。だからこそこれは午後にやろう、これは明日に先送りしようといった微調整ができます。予定駆動の場合、調整は効かないでしょうから向いていません。

Calendarer

概要

カレンダー片手に、予定ベースで動く戦略です。タスクも予定として登録します。

最もわかりやすく馴染み深い戦略だと思います。カレンダーを使って予定を俯瞰し、ひたすらそのとおりに動くというものです。

カレンダーは「使ったことがない」人の方が珍しいかもしれませんが、Calendarer とは単にカレンダーを使っている様を指すものではありません。むしろ「重用している」ニュアンスです。カレンダーがないと話にならない、一日に何回も見ている、見ないなんて考えられないというレベルです。率直な言い方をすると、カレンダー中毒者とも言えそうです。

予定とは「開始時間と開始場所を守らなければならないもの」を指しますが、これを忘迷怠なくこなせる手段は限られています。厳しく誰かに管理されるか、予定の接近に気づかせてくれる場に居るか、そうでなければ時間軸に可視化してそれを高頻度に眺めて忘れないようにするか、のいずれかかです（直前にアラームを鳴らすこともできます）。いずれにせよ、カレンダーの形で時間軸を俯瞰する営みは、疑うまでもなく浸透しているものです。だったら、もうこれを重用して、タスクもなるべく詰め込んでしまえばいいのでは、と考えます。それが Calendarer です。

カレンダーは防御網でもあります。時間という有限性のあるものを消費しているので、「これ以上は入らない」がとてもわかりやすいのです。タスクはキャパオーバーになるほど抱えがちですが、Calendarer には無縁です。ただし予定を入れすぎて過負荷になることはあります。

応用1: 行動制御

カレンダーは自分への命令書とみなすこともできます。

ということは、どんな命令（予定）を書き込むか次第で、自分の行動を制御できます。たとえば「休憩」「昼休憩」「個人ワーク」「退社前の準備」といったタスクを明示的に予定に入れておくことで、忘れずこなせるようになります。自分で入れた予定に自分で従う茶番が通じない人は、人や環境を巻き込んでもいいでしょう。休憩タイミングを友人や同僚と合わせたり、定時退社

後にジムやエステに行く用事を入れてしまったりすれば比較的従う気になります。

応用2: 公開範囲の絞り込み

特に仕事の文脈では自分のカレンダーが公開されることがあります。チームメンバーに自分の予定を見せて、空きがあれば会議を入れてもらったりするためですね。このように共同を意識したカレンダーが **スケジュール** です。皆のスケジュールを上手く可視化、共有するツールがスケジューラー（グループスケジューラー）と呼ばれるものです。本章は個人タスク管理の話なので、グループスケジューラー自体については取り上げません。

しかしながら、グループスケジューラーを個人的なカレンダーとして使うことがよくあります。特に仕事だと実質的にグループスケジューラー≒個人の（業務時間中の）カレンダー、になりがちでしょう。これはつまり**「他の人から見える」個人のカレンダー** を使っていることと同義です。さて、Calendarer は、予定をバンバン書き込む戦略でした。他人から見えてしまっては困るような、個人的なタスクも扱えるべきですが、書き込みづらいですよね。じゃあどうするか、書き込まないようにするか、それとも他人が読んでもわからないようにダミーの予定名を書いておくか、はたまた別のカレンダーをつくってそっちを使うか。やり方は色々ありますが、いずれにしても面倒です。

面倒でも越えるしかありません。魔法のような方法は無いと思います（あったらそれを売り出してご飯を食べられると思います）。方法をいくつか挙げます:

- 書き込まない
  - プライベートでは Calendarer をやめる
- 別の私的なカレンダーを使う
  - カレンダーが分散して運用が面倒くさい
  - 自動化してデータを同期できれば楽だが、コンプライアンス的に許されないケースがある
  - Dailer は毎日メンテナンスを行うが、同様にカレンダー間の同期を手作業で行うというメンテナンスを受け入れてしまう手もある
- ダミーの予定を書き込む
  - たとえば定時退社後にジムに行きたい場合、「ジム」とは書きづらいかもしれない。「私用」などと書くはず
  - 牽制として使える
    - この場合、定時間後に予定がありますよ（この日は残業しませんよ）との牽制をしている
- 業務中は Calendarer にならない
  - 業務中はスケジューラーを使わないか、必要最小限でのみ使うようにする
  - 私的なカレンダーの方では、おそらく業務時間部分を「仕事」という単一の予定で埋める格好になる
- フルオープン
  - 経営者など限られた人にしか使えないが、開き直って私的なカレンダーとしても使う

応用3: リマインダー

デジタルなカレンダーツールはリマインダー機能を持っていることがあります。

リマインダーとは n 分前に予定を教えてくれる仕組みで、目覚ましのアラームが最も有名でしょう。ツールとしてはポップアップメッセージの形でお知らせしたり、通知として教えてくれたりするものが多いです。いずれにせよ忘迷怠、特に忘れることを防ぐ意味で重宝します。たとえその予定を忘れていたとしても気付けるからです。

この性質上、忘れっぽい人や作業にのめりこみがちな人には特に重宝します。筆者も日頃から多用しており、リマインダーのおかげで会議のすっぽかしを恐れず仕事できています。ポップアップだけだと怪しいので音も鳴らすようにしてます。

一方で、すべての予定にリマインダーをセットするのはやりすぎでしょう。「別に忘れてないよ、わかってるから」というときでも構わず通知されるとかえって鬱陶しいです。何事にも麻痺はありまして、リマインドされたときにいったん無視して――そのまま普通に予定をすっぽかしてしまう、なんてことも起こり得ます。**リマインダーは「ひとりだけで」すっぽかしを防ぐための最後の砦** なので、麻痺してしまわないよう、使いすぎないことが実はかなり重要だったりします。特に Calendarer は予定をたくさん書き込むだけあってリマインダーも増えがちです。

応用4: オプラン

Calendarer は予定をひたすらこなすマシーンのようなものですが、その予定を律儀に守ると非常に疲れます。頭の切り替えや物理的な移動や準備が要りますし、何より予定≒誰かと話すことなところがあり人付き合いです、人付き合いは（たとえ多少慣れた相手であっても）気を張るため疲れます。メンタルも削れます。そうして疲弊しきってしまうと、ほとんど何もできなくなってしまいます。よくオンとオフの切替とか週末に寝溜めとかいいますが、そういったことが必要なのは単に疲労しきっているからです。疲労しきっている人は、たとえ定時退社であってもそうなります。残業ならなおのこと。それで疲労していて生産的なことができないから、いわゆる夜更かしでカバーしようとします。報復性夜更かしとかリベンジ夜更かしと呼ばれています。夜更かしをすると睡眠不足になりますから――と負のスパイラルです。この諸悪の根源は **日中に予定が多すぎて疲弊しきっているから** のことが多いと筆者は見ています。

一部の鉄人や生き急ぎでもなければ、人は予定まみれの生活には耐えられません（本当に才能がある人もいますが、たいていは単に意思決定や割り切りが人一倍上手ゆえに消耗ペースが緩いだけです、非情とも言えます）。どうすればいいかというと、予定を減らすのです。あるいは「最悪破ってもいい予定」「手を抜いてもいい予定」を取り入れます。後者のような予定を **オプラン(Oplan, Optional Plan)** と呼ばせてください。本当は前者、予定そのものを減らせればいいですが、そうもいかないでしょうから残るは後者、解釈を変えるオプランとなります。

オプランはどのように増やせばいいのでしょうか。

まず、人との約束など本当に破れない予定については **バレないようにサボりを入れます**。会

議でギッチギチの中間管理職を例にすると、前の会議が遅れてますとか、機材トラブルで音声が聞こえないのでとか、ちょっと緊急の電話入ったのでとかいった言い訳を入れて 10 分くらい稼いだりします。あるいは会議には参加しているが内職をしていたり、今はリモートワークも多いですからデバイスは繋いでいるがのんびりトイレしたり家事をしたり、といったこともあるでしょう。体裁は重要ですからバレないように、は前提として、サボること自体は意外と受け入れられます。これもまさに、そうしないともたないからです。

次に自分個人のタスクを予定として追加したもの（Calendarer はタスクも予定として扱うことで予定ベースで動く戦略なのでしたね）の場合は、単に「この予定は最悪破ってもいいものだ」と捉えます。たとえば休日 14〜15 時で「生成AIを勉強する」予定を入れていたとしたら、本来ならちゃんと 14 時から 1 時間勉強するべきですが、この律儀さばかりだと疲弊するという話でした。かわりに、「14〜15時と1時間分勉強できたら理想だけど、別にできなくてもいい」と捉えます。買い物してなかったからちょっと 20 分くらい行ってきて 14:30 くらいから 30 分だけ行う、でもいいですし、なんか気分が乗らないなぁと思ったら開き直ってゲームしてもいいでしょう。オプランはそういうものです。守られないことはよくあります。

ただし、以後ずっとカレンダーを無視します、だと意味がないので、**きりのいいところで、なるべく早めに（予定をちゃんと守ることを）再開してください**。再開を試みて、またオプランをサボったとしても、それはそれで構いません。仮にオプランをサボらずにちゃんとこなせる率が 25% だとしても、4 回に 1 回は守れているわけでして、何も管理せずグダグダになるよりははるかにマシです。このマシを手に入れるために、再開というアクションが必要なのです。

ここで「そんなことに意味はあるのか」、「そもそもタスクを予定として書かなければいいのでは」と思われるかもしれませんが、意味はあります。予定に書かなくても生活がまわるのならそれでいいのですが、Calendarer の人はおそらくそうではなくて、まわしたいからこそ、少しでも多くのタスクを予定として書き込むわけです。ただ、そうして書いた予定を律儀に守るとメンタルが持たないので、オプランとして扱いましょうという話です。予定にすらしなくて何もしないでもなく、逆に予定として律儀に扱って疲弊するのでもない、その間を取るのです。

オプランを増やせるようになると、予定でギッチギチのカレンダーであっても持続できるようになります。一つの目安としては、Calandarer は **睡眠以外の予定がギッチギチの日が一週間続いても生活が持続する状態** が適正です。オプランによりガス抜きが出来ていると、ギッチギチでももつのです。オプランの部分はほとんど守れていない可能性もありますが、それはそれでいいのです。少なくとも何も管理していない形骸化よりも（よりタスクをこなせている、少なくとも着手できているという点で）はるかにマシです。

もし適正に至れない場合、おそらくはオプランが少なすぎます。あるいは Calendarer には向いてないのでしょう（他の戦略を使いましょう）。もしかすると「そもそもタスク管理が要らない」生活がすでにできているかもしれません。それでも仕事や忙しい時期などでカレンダーを多用する時間帯や時期はあるでしょうから、このオプランは覚えていて損はないです。

メリデメ

メリット:

- 直感的で運用しやすい
- 時間という有限性によりタスクを抱えすぎる事態を防ぎやすい
- 行動制御的に使ったり、リマインダーですっぽかしを防いだり、など強力なサポートも使える

デメリット:

- 仕事でスケジューラーを使っていると使い分けが面倒
- 詰め込みすぎると疲弊しやすく、特にメンタルにくる
    - 自分ひとりの予定ならオプランにすると良い
- 予定に書き込まなかったタスクがこぼれやすい
    - ないしは「忙しいから」と確信犯的にスルーしがち
    - 取捨選択や意思決定

適性

予定が多い方に適しています。会議の多い管理職、指定場所への訪問が多い学生や営業職、作業や準備の多い指導者教育者やアスリート、その他も多数当てはまるかと思います。

役職の高い方など権限が集中している方にも適しています。このような人は基本的に「いつ誰に自分を使ってもらうか」の取り合いになるので、スケジュールを開放して空いたところに勝手に入れてくれ方式にするか、秘書など調整役をはさんだりします。予定の選別や判断にかかる認知コストがバカにならないので、最初から諦めて、「私はスケジュールのとおりに動く機械になります」とするのです。

自己管理が苦手な方にも適しています。自己管理とは自分で決めたことをそのとおりに実行する能力のことで、たとえば今日はこれをすると決めてタスクリスト 6 行分を書いたら、基本的にそのとおりに上から行動することです。得意な人は「誰でもできるのでは」と思いがちですが、集中力が無い人や気分屋な人、特に特性上脳内が慌ただしい人などはこれが難しいのです。自己管理が苦手な人は、自分ひとりで自分を御するのはほぼ不可能なので、人や環境の力に頼ります。応用1で挙げたとおり、約束が発生する予定を積極的につくればいいのです。約束であれば（自己管理が苦手な人でも）比較的守ろうという気になります。

クリエイターなど創造性の高い仕事をしている方には向いていません。創造はアイデアとモチベーションの世界ですから、予定などという事前の制約など邪魔以外の何者でもないからです。ただし「作業」を行うときや人からの合意を得るときなど創造性が要らない場合は、その限りではありませんし、（特に仕事だと）たいていはそれで事足ります。創造に頼ったクリエイターは非常に少ないので、通常気にする機会はありません。強いて言えば、普段から時間を持て余している趣味人でしょう。

Slotter

概要

スロット（時間帯という枠）を設け、各枠の中で何をするかを考える戦略です。

Dailer は「日」、Calendarer は「予定」という形で時間を区切っていましたが、Slotter は「時間帯」で区切ります。時間帯とはタスク管理用語では **セクション（Section）** と呼ばれることがありますが、30分～数時間くらいの帯をイメージしてもらえればと思います。

以下に特に有名な時間帯（の分け方）を挙げます。

- 朝、昼、夕方、夜
- 午前、午後
- 出社前、午前、昼休憩、午後、退社後、帰宅後
- 一時間目、二時間目、……

時間帯に注目する理由は何でしょうか。二つあります。時間割などで馴染み深いのが一つ、そしてもう一つは私たちの生活は時間帯的なリズムがつくことが多いからです。特に後者は、自覚できていないだけで、通常はかなりのところまでパターン化します。朝は大体こういうことをしていて、昼休憩前後ではこうしていて……のようなパターンがあるはずです。曜日ごとに違ったり、繁忙期で乱れたりすることはありましが、一週間のスパンで見ると大体パターン化していると思います。

Slotter は、この時間帯的なリズムを捉えます。自分にとって自然な時間帯を自覚し、必要なら設計や調整もして快適に過ごすことを目指します。タスク管理としてはそこまで細かくはなく、この時間帯の間は大体こんなことをしようかと決めたり、普段はこれをしているけど新しいことがしたいからちょっとずらそうかといった調整をしたりするくらいです。よく Calendarer と混同しがちですが、Slotter はあくまでも時間帯という枠を設けているにすぎません。予定のような強制力はありませんし、グループスケジューラーで人に見せる意図もありません。あくまでも自分の、自分による、自分のための時間帯設計にすぎません。

応用1: スロットの把握と運用

今現在の自分の生活にもすでにスロットが反映されています。これを自覚するためには、日々自分が何をしているかを記録して可視化する必要があります。

タスクシュート はまさにそのために設計されたものですが、煩雑かつ神経質な運用を求めるので人を選びます。Toggl などタイムトラッキングツールは比較的シンプルに記録できます。アナログが好みであれば、時間枠を表現できる手帳を使って「ここからこの時間帯まではこんなことをしてました」を記録するのも良いでしょう。仕事は赤、家事は青、趣味は緑など色を使い分けて塗りつぶせば、視覚的にも楽しくなってモチベーションが持続します。いずれにせよ、自分の行動を明示的に記録するという本質的な面倒くささからは逃れられませんので、記録したい方はご自身のモチベーションを保てそうな方法を探してください。

一週間、できれば一週間分の記録を数回（数週間）くらい取ってみたいところです。俯瞰してみると、どの曜日のどのあたりにどれくらいの大きさのスロットがあるか、といったことが見えてきます。おそらくたいていは「思っているよりも余暇が少ない」「仕事や家事に忙殺されてすぎている」という悲しい現実が見えてくると思います。一方で、「木曜日の午前はちょっと空いてるな」などスキマも見えるかもしれません。

自分のスロットがわかったら、あとはこれを尊重する形で、どのスロットで何をしようかを考える・微調整していくだけです。気に入らない場合は再設計することになりますが、スロットの大掛かりな修正は新たな習慣の獲得と同レベルの高難易度なので通常は気にしなくていいです。再設計は引っ越しや転職など環境が無理やり変わったときのお楽しみにしておきましょう。しかし、環境が変わった場合でも結局は元々の自分のスロットに回帰したりします。自分のスロットを自覚できてないと、回帰して落ち着くまでが遅れます。逆を言えば、前もって把握しておけば、環境が変わってもそこに合わせに行くという形で能動的に素早く安定を引き寄せることができます。

ちなみに、稀に自分のスロットに固執しない人がいます。旅行好きな人や、寝たいときに寝て起きたいときに起きて仕事したいときに仕事する人（生活リズムを持たない人）などがそうです。

応用2: 自分の拘束方法

スロットの最中に、いかにして自分を拘束するかには好みが出ます。

最も端的なのは上述した Calendarer――つまりはカレンダーを使うことです。たとえば「朝」「午前」「昼休憩」「午後」「終業前」の 5 つのスロットがあるとして、これをカレンダーに反映し、かつリマインダーも設定します。するとカレンダー上はこの 5 つの太い予定がぎっしり詰まった見た目になります。毎日、朝や午前の直前にリマインダーがお知らせしてくれるので切り替えもしやすいでしょうし、カレンダーをこまめに見ることで「あと 40 分で昼休憩か……」と次のスロットに備えることもできます。この例だと単純すぎていまいち効果がわかりませんが、スロットが 7 個、10 個など増えてくると結構効いてきます。

集中術を使うこともできます。タイムボックス のように「仕事はこのスロットだけで行うぞ」と割り切ればメリハリをつけられます。筆者も「創造的な仕事は午前スロットだけにする」など、体力や精神力をセーブするためにタイムボックスは結構使っています。ポモドーロ・テクニック も使えそうです。それこそポモドーロ・スロットなるスロットを設けて、その時間帯では何か一つタスクを選んでポモドーロを回すことに専念する、という過ごし方もできそうです。スロットの切り方と、その中で何をするかは自由ですが、集中術含めた仕事術は上手く取り入れると便利です。

場所を変えるのもアリでしょう。たいていの会社員は午前と午後の 2 スロットで仕事を行ないますが、出社してから行ないますよね。場所が変わっているからこそメリハリがついていたところがあります。これがリモートワークで崩れてしまい、メリハリをつけられなくなって体調やメンタルを崩す人が増えたとはよく聞くことです――とこれはわかりやすい例の一つにすぎません。出社に限らず、もっと柔軟に場所を使うことはできます。朝活ができて、かつ金銭も許すなら、出社前に「カ

フェ」というスロットをつくって、喫茶店で何か活動してもいいでしょう。あるいは週に1、2回くらい就業後にボーナススロットをつくってどこかに出かけるプチ贅沢を許すのもアリです。

メリデメ

メリット:

- 時間帯という大半の人がすでに無自覚に持つパターンに則れるため、QoL が高くなる

デメリット:

- 自分のパターンを知るためには辛抱強い記録作業を要する
- 自分のパターンから抗うのが難しい
    - 環境が変化してやむを得ず崩れる場合でも、結局自分のパターンに回帰しがち
- 良くも悪くも枠を設定しているだけであり、実際タスクを回せるかは本人の力量次第

適性

生活リズムを持たない人（持てない人というよりも持ちたくない人）には向いていません。旅行家や一部のクリエイターが当てはまります。

話題で区切る

1タスク1ページなどタスクごとに専用の領域を設ける類の戦略です。情報を残しつつ一つの対象にじっくりと取り組み続けるので、継続したい人や寝かせたい人に適しています。

Issueist

概要

GitHub Issues など BTS を使う戦略です。

先に BTS について軽く触れておくと、バグ管理システム（Bug Tracking System）の略です。1タスク1チケット（ページ）でタスクを管理するツールであり、その名のとおりソフトウェア開発におけるバグを一つずつ確実に管理するものでした。Redmine や Backlog などプロジェクトタスク管理の文脈で多数のツールが存在します。開発プラットフォーム GitHub にも Issues と呼ばれる同様の機能がありますが、こちらは特に個人タスク管理としても使いやすい仕上がりとなっており、主にエンジニアが活用しています（※1）。今でこそプロジェクトタスク管理は俯瞰と注視が当たり前ですが、この概念は BTS から来たのだと筆者は考えています。

Issueist は、BTS を用いて俯瞰と注視に専念する戦略と言えます。まず1タスク1ページになっていますので、各タスクとじっくり向き合うことができます。情報を書き残しておけば、2週間後とか1ヶ月後に来ても（それを読めば）状況を思い出せます。チャットのようにテキストや画像を残せるので残すのにも苦はありません。次に俯瞰機能も備えており、フィルタリングやソートも柔軟に行えます。適切に運用されているなら、数百件、1000件以上のタスクの中から必要なものを取り出すこともできるでしょう。また最近のツールですとカンバンのようなボード機能を有していることもあり（GitHub にも Projects というボード機能がある）、リストビューが苦手な人でも俯瞰しやすくなりました。

- ※
  - 1 イメージで言えば、Slack や Discord といった業務用・チーム用のツールをひとり用で使っている感覚です。通常、そのようなツールはひとり用で使うには多機能すぎて辛いのですが、GitHub は使い心地が（特に当時は）群を抜いていて、快適性にうるさいエンジニアでも唸るほどでした。現代的なタスク管理ツールをつくっている人達を含め、多くのエンジニアにインスピレーションを与えたのは間違いないのではないかと思うほどです。筆者もそのひとりです。

応用1: どこまで丁寧に運用するか

丁寧に運用すると、それこそプロジェクトタスク管理になります。個人の生活全般でそれをするのは息切れしますし、オプランの項でも述べたとおりメンタルももたないと思いますが、大きな仕事や用事をスポットで丁寧に運用するのはアリでしょう。特に GitHub はリポジトリ（プロジェクトと同義）という単位で自由につくれますから、引っ越しリポジトリ、書籍執筆リポジトリなどをつくってその中では丁寧にやる、とすればいいのです。

雑に運用することもできます。筆者としてはこちらが多い印象です。やらなくてもいいけどやりたいこと、特に考え事や調べ事や新しく試したいこと等のタスクを雑多にぶちこんでおき、気まぐれに眺めて気まぐれに消化するという使い方をします。いいかげんな分、消化量やペースは安定しませんが、それでも何も管理しないよりは進捗が出ます。

後者と前者はトレードオフです。後者のように雑に管理すると楽ですが、消化の進捗は安定しません。前者のように厳しく管理すると消化は捗りますが疲れます。ご自身の適性と仕事に応じて使い分けると良いでしょう。

よくある折衷案はバックログ（在庫）的な使い方です。今後やりたいタスクを Issue としてとにかくつっこんでおき、余裕がある時にそこを眺めて消化していきます。余裕がない時は一切触れなくても構いません。しかし、やりたいタスクを思いついたら、忘れず追加しておきます。追加しておけば忘れることはありません（取り出せるかは別問題ですが少なくとも存在はする）。永遠に触れないのを懸念するなら、週一や月一などで棚卸しをするといいです。休日の午前に数時間くらい時間を取って、バックログを眺めながら何するか考えるひとときは楽しいと思います。

もう一つ、インボックス的な使い方も可能です。つまり「オープンになっている Issue」が 0 件の

状態を日々目指すという前提のもと、日々タスクとして Issue をつっこみ、消化して、消化できたらクローズにします。別の言い方をすると Dailer のやり方を Issue で行ないます。あるいは Dailer ほど極端でなくとも、仕事や生活の一部のみをインボックスゼロにするだけでも便利でしょう。

応用2: 何のツールを使うか

特に問題なければ GitHub Issues 一択でしょう。リポジトリ単位で区切れますし、動作も軽快ですし、と柔軟性と軽さの観点で優れています。

すでに他のタスク管理ツール（特に BTS などプロジェクトタスク管理寄りのツール）を使っているのであれば、それに慣れているでしょうからそれを使うのが良いでしょう。Issue 系のツールは機能に大差は無いので、最終的には慣れや好みの問題です。ただし個人が使うには料金的に割に合わない可能性があります。

いずれにせよ、どれもエンジニア向けの高度なツールであることが多いので、初学者の方が手を出すのは厳しいと思います。たとえば GitHub は日本語 UI がありません。

メリデメ

メリット:

- すでに BTS に触れてる人にとっては馴染み深い
- 俯瞰と注視に専念できる、シンプルな使い心地

デメリット:

- 初学者のハードルが高い
  - 概念まわりにしてもツールの使い方にしても
- 管理をどこまで厳しくするかのトレードオフが難しい
  - バックログ的な使い方という折衷案に落ち着くか、TODOリストと同様に形骸化しがち
- 細かいタスクや多数のタスクを扱いづらい
  - タスクの操作に手間がかかるので、粒度が大きめのタスクを主に扱いがち
  - また小回りを利かせたい用途には不向き（例: 買い物に関するタスク管理だけ行ないたい）
- 俯瞰を機能させるのが難しい
  - フィルタリングやソートの場合は、優先度やタグやラベルなどを上手く使いこなす必要がある
  - ボードの場合は、タスク（付箋）の置き場所を日々メンテナンスする必要がある

適性

BTS に慣れている方には適している可能性が高いです。一つの目安は GitHub Issues を

知っているか・使えるかどうかです。主にソフトウェア開発者が当てはまります。可能性が高い、と書いたのは、あくまで慣れにより順応しやすいという意味であって、話題で区切る（1タスク1ページ）戦略に合うかどうかがわからないからです。BTS には慣れているが合わない、もありえます。

ナレッジワーカーにも適しています。この人達は知らないことを調べたり、正解が無いことを考えたりといったことが必要ですが、Issueist としては 1-検討事項 1-ページ の形で思考を蓄積していけます。特にしっかりとしたノートや資料ではない、ラフな残し方で済ませたいタスクを扱うのに適しています。たとえば「～～について考えてみる」のようなタスクを気軽につくれます（※1）。

- ※
  - 1 この感覚はナレッジワーカー特有だと思います。タスクとして通常浮かべるのは「外部環境から降ってきたものを忘れず処理する」か「未来の脅威に備える」です。しかしナレッジワーカーは、新しいナレッジをつくりたい・つくることで成していきたい人達であり「別に降ってきてはないし、備えるつもりもないし、何の役に立つかもわからないけど考えたい」ことがよくあります。頭の中だけだと忘れちゃいますし、雑なメモだけだと一度考えただけでおしまいです。かといって、ちゃんとしたタスク管理をするほどではないし、思考は創造的であり管理するものでもありません。バランスで言えば、「～～について考えるためのエリア」をつくっておいて気まぐれでいじくる――くらいのものが欲しいわけです。話題で区切れる BTS はまさにうってつけなのです。

Topician

概要

Wiki やノートツールを使う戦略です。

タスクも含めた「トピック（話題）」を 1-トピック 1ページ で扱います。メインはノート管理であり、その中でタスク管理も行うという位置づけになります。ノートの取り方は色々ありますが、Topician の哲学は 1-トピック 1ページ です。A という話題はページ A に書きます。**トピック指向** と呼ばせてください。トピック指向に話題の混線や脱線を防ぎ、A について扱いたければページ A に行けばいい、と単純化できます。集中や再開がしやすいとも言えるでしょう。その分、A について書くためにページ A にいちいち行かなきゃいけない手間はありますが、この手間を負うだけの価値があるのです。千や万のページを破綻無く持つことすら可能になります。

部分的には他の戦略を含んでいます。たとえば「2024/05/01」「2024/05/02」のようなページをつくる運用をするなら Dailer になりますし、タスクだけを扱うと Issueist になります。ならば Issuiest で良いのではないか、と考えがちですが、Topician はタスク管理がしたいのではなくノートを書きたいのです。そのついでにタスク管理も行ないたいのです。タスク管理は二の次です。実際、用途を限定したいのであれば Issueist が適しています（あるいは日単位でメリハリをつけたいなら Dailer が良い）。そうではなく、自分なりにノートを取りたい、それを貯めていきた

い、そういう営みをしていきたいからこそ Topician を選ぶのです。

Topician がどのようにタスク管理を行うかですが、結論をいうと **何とか工夫してタスク管理の能力を実現する** です。単純なやり方ですと、たとえばタスクとみなしたい行に `#task` のようなタグをつけておき、タスクを見たい場合は `#task` で全文検索をする、になります。この場合は全文検索（テキストエディタの文脈では Grep と呼ばれます）の知識や律儀にタグをつける行動は求められます。単純なやり方であってもこの程度は要求されるのです。この他には、Dailer のように毎日（あるいはもっと高頻度 or 低頻度で定期的に）ノートを見返してタスクを手作業で抽出する「振り返り方式」、hown のように一覧表示や連結表示など表示の工夫や、指定日が近づくと徐々に項目上位に浮かんでくる仕組みなど高度な工夫を凝らした「テキストエディタ拡張型」、最近ですとノートを生成AIに読み込ませて知りたいことを尋ねる「生成AI方式」もあります。DIY と同様、どのように工夫するかは自由自在ですし、筆者が知らない工夫も様々存在するでしょう。もちろん自分でツールをつくる人もいます。共通するのは **プログラミングやテキストエディタのカスタマイズなど、技術的なスキルが要求される** 点です。

応用1: どのツールを使うか

古典的には Mediawiki、Pukiwiki などの Wiki が好まれます。Wiki というと Wikipedia やゲーム攻略 Wiki など有志が多数集まって書き込んで辞典レベルの巨大なコンテンツをつくりあげるものをイメージしますが、Topician はそれをひとりで行ない、自分の自分による自分のためだけのコンテンツをつくりあげていきます。しかし、これら従来の Wiki には右記のデメリットがあります――セルフホストのハードル（自分でサーバーを立ち上げる必要がある）があること。単純な機能しかなく拡張性にも乏しくてタスク管理を成立させる余地が少ないこと。これらを **第一世代** と呼ばせてください。

Wiki のようなノートツールの重要性は近年注目されており、この手の製品も増えてきました。Microsoft の OneNote、Box の Box Notes、Dropbox の Dropbox Paper など商用のものもあれば、esa.io や Qiita Team などエンジニア向け発の使いやすいツールもあります。Notion もこのカテゴリに入ります。また、Workflowy や Dynalist などアウトライナーと呼ばれるツールもここです。これらを **第二世代** と呼ばせてください。第二世代はノートとしては使いやすいですが、やはりタスク管理を行えるほどの柔軟性はありません。あるいは、できてもプロジェクトタスク管理のような「しっかりしたタスク管理」になりがちです。

もう一つ、**第三世代** ともいうべきツールも登場しています。Scrapbox、Obsidian などです。第三世代の特徴はリンクを重視していることで、ページからページにリンクを張ることがかんたんに行えるようになっています。また「このページにリンクしているページの一覧」などリンクベースで関連ページを辿ることで周辺の俯瞰も充実しています。第三世代はノート全体をネットワーク構造と捉えており、ノートツールのパラダイムシフトであると筆者は捉えています。ネットワーク構造は脳の構造でもあり、記憶の特性としても自然で思い出しやすいのです。一覧のフィルタリングやボードのメンテナンスではなく、リンクを辿りながら思い出していく体験になります（どこから辿るかのスタート地点決めには従来の検索や一覧化も使う）。言葉では説明しづらいですが、一度体感すればやみつきになると思います。生成AIの仕組みも脳を模倣していますし、脳の

仕組みは親和性が高いということでしょうか。今後ますます主流になってくると思います。

タスク管理用途でも発想の転換を余儀なくされます。主に前述の「振り返り方式」に頼ることになります。言い換えると、こまめにノートを書いたり、読み直したり、書き写したりすることでタスクに関する情報を継続的にメンテナンスし続けるイメージ、とでも言えばいいでしょうか。このあたりの話は高度になりますし、Topician とも外れていきますので後の章で扱うことにします（文芸的タスク管理）。

最後に、世代ごとに特徴をまとめておきましょう。

- 第一世代
  - ☐ 古くから親しまれている
  - ☐ セルフホストのハードル
  - ☐ タスク管理を成立させる余地に欠ける
- 第二世代
  - ☐ 様々なツールが存在し、自分に合うものと出会える可能性が高い
  - ☐ ツールが多いので選定や試行が必要、定着まで時間がかかる傾向がある
  - ☐ タスク管理を行うにはまだ柔軟性に欠ける or できてもプロジェクトタスク管理的になる
- 第三世代
  - ☐ リンクで繋いだネットワークというパラダイム、自然に辿りやすい・思い出しやすい
  - ☐ 新しい概念であるため学習や適応に手間がかかる可能性
  - ☐ タスク管理として使う場合、タスクに関する情報を継続的にメンテナンスしていく営みになる

メリデメ

メリット:

- ノートの延長でタスク管理を行える
- 情報とタスクを自然に扱う（結びつける）ことができ、情報を得やすいため主体的な行動がしやすい

デメリット:

- タスク管理の実現がそもそも難しい
  - マンパワーによるメンテナンスを前提とした原始的なあり方か
  - プログラミングやテキストエディタ拡張など高度なスキルを用いて自分でつくる、工夫するか

適性

適性は「日頃からデジタルな手段でノートを取る側の人間かどうか」です。たとえばプライベート

でも好き好んで書くか、年単位で継続的に書いているかとか、そういうレベルです。イエスと答えられる人だけが向いています。というより、イエスと答えられない人には（ノートを日々書くことが）苦痛になるため耐えられません。

忙しいとノートを取りづらいという意味では、日頃からノートを読み書きできるだけの余裕があるかどうかも適性に含めて良いでしょう。たとえば 1 日 1 時間、できれば 2 時間以上を恒常的に確保できるでしょうか。「ない」「できない」人は向いていません。あるいは、せいぜい「メモを取る」「あとでテキトーに活かす（大体生かせません）」か「散文的に書いて自己満足しておしまい」が関の山でしょう。

一箇所にまとめる

タスクや物タスクを一箇所にまとめる戦略です。手間暇かけず済ませたい人に適しています。

前提知識：物タスク

先に前提知識を書いておきます。

物タスク とは、タスクを想起するものやタスクの構成要素となっているもののことです。たとえば「壊れた時計」は「時計を修理する」タスクを生み出すので物タスクですし、「上司の顔画像」も「上司から頼まれていたこと」を思い出すという意味では物タスクになりえます。

ある物が物タスクかどうかは、その人の解釈によります。同じ人でも状況が違えば解釈も変わります。

タスクとの違いですが、タスクは「やることが文章化されたもの」です。手帳だったりアプリだったりしますが、何らかのタスク管理ツールに入力されていることが前提です。このあたりはタスク管理の公理として述べました。一方、物タスクとは上記のとおり、タスクになるかもしれない物のことです。

本戦略ではタスクも、物タスクも、どちらもまとめようとします。タスクをまとめるとは、たとえば今日やるタスクを付箋に書いてディスプレイに貼り付けておく等です。物タスクをまとめるとは、たとえば今日出社時に外で済ませたい用事（に必要な物）――壊れた時計と縛ったゴミ袋と終業後オフ会用のおめかしセットを玄関前に置いておく等です。より厳密に言えば、普段から「外で済ませたいものは玄関に置いておけ」というルールを決めておくニュアンスです（その場限りで行うのではなく）。

Richild

概要

物タスクを一箇所に集める戦略です。

イメージは「裕福な子供」で、名前も Rich な Child → Richild としています。偏見になって申し訳ないですが、裕福な子供はたくさんの物を与えられます。一方で、整理整頓に時間を割く必要がないほど空間も潤沢ですが、散らかしているとさすがに不便なので、一箇所に押し込みます。おもちゃ箱は良いイメージで、おもちゃをとにかく箱の中に雑でいいのでぶっこみます。遊ぶときはガサゴソと漁って探します。

Richild の典型例はデスクとデスクトップです。散らかった机や、アイコンで散らかった PC のデスクトップ画面、あるいはスマホのホーム画面、他にもブラウザのブックマークやメールの受信箱やチャットツールサイドバーのチャンネル一覧など色々ありますが、とにかく一箇所にまとめて散らかしているのが特徴です。

Richild がタスク管理として成立する原理は二つあります。

まず、散らかっているとはいえ一箇所にあるので、そこを探せば見つかりやすいことです。一見すると面倒くさそうですが、どこにあるかわからないものを探したり、その整理整頓に時間をかけるよりも、「一箇所の中から探せばいいだけ」の方が実は効率が良かったりします。何より疲れません。探すときはだるいですが、整理整頓というメンテナンスに手間をかけなくても良いことが非常に大きいです。天才は整理整頓が下手、とのイメージがありますが、これは理に適っていて、単に天才と呼ばれるほど突き詰めてる人は意思決定コストの節約にもシビアであり、タスク管理なんかに費やさないからです。

次に、散らかった状態でも気にせず仕事をしたり、散らかった状態から必要なものを見つけたりといった「頭の性能」も必要です。これがないと、ただの管理破綻者、ただの「散らかし癖を持つ無能」になってしまいます。ここで「汚くても見逃してもまあ何とかなるよね、的楽観性なる資質も要るのでは？」と思われるかもしれませんが、筆者としては楽観性は頭の性能次第だと考えています。性能があるからこそ楽観になれるのだと思います。

応用1: おもちゃ箱の文脈と箱数をどうするか

まず Richild には文脈があります。たとえば「仕事」と「私生活」なら二つです。複数の肩書を持つ人は仕事の方は複数あるかもしれませんし、多拠点生活をしている人も拠点ごとに文脈が増えるでしょう。

次に文脈ごとに箱（タスクをまとめる先）があります。仕事の場合、デスクとデスクトップがあるなら二つです。デジタルで完結できる人はデスクは綺麗だがデスクトップは汚いかもしれません――と、これは例の一つにすぎません。デスクとデスクトップはわかりやすいですが、それ以外もあります。特に近代ではチャットなどのワークスペースも増えており、そちらが箱になっていることもよくあります。

このあたりのバランスはどのようにすればいいでしょうか。

基本的には **必要なときに必要な箱をその場のノリで追加して試す** になるかと思います。Richild ならそうなります。一方で、それだと場当たり的で、頭の性能が足りてて上手くいっているならいいのですが、いかないときがあります。「なんか何やっても上手くいかないんだよなー」になりがちです。ここでさらに二通りに分かれます。開き直って諦めるか、腰を据えて考えるかです。後者を行ないたい場合に、文脈と箱を設計すると良いです。

かんたんな手順を挙げておきます:

- 1: 文脈を洗い出す、あるいは決める
- 2: 文脈ごとの箱を洗い出す、あるいは決める
- 3: 複数の文脈を横断する「横断箱」を洗い出す、あるいは決める

最後の横断箱については、フリーランスなど仕事と私生活を絡めやすい人限定の概念でしょう。無理して箱を分けず、いっしょくたにしてしまった方が Richild として快適になることがあります。サラリーマンからするとだらしないと感じるかもしれませんが、Richild 自身が頭の性能でカバーして仕事を上手く回せるとしたら失態もしないため問題ありません。

最後に、例として筆者の運用を軽く紹介しておきます。私生活文脈の箱として「玄関」と「洋室」の二つを用意しています。玄関の下駄箱の上には「外出時に持っていくもの」と「郵便受けに入っていた書類群」が雑に置いてあります。洋室には「今の季節で着る衣類」が下着含めて全部雑に並べてあります。季節外のものはクローゼットに仕舞ってます――と、これは少々几帳面であり Richild らしくないですが、文脈に応じた箱を設定して、そこにぶちこんでいる点に注目してください。

応用2: Richild はタスクではなく物タスクを扱う

鋭い方は気付かれたかもしれませんが、Richild の説明として

> 物タスクを一箇所に集める戦略です。

と書きました。これは意図的です。Richild は **物タスクを** 管理する戦略だからです。

物タスクは「物」であり、物とは現実空間に三次元的に存在し、多種多様な情報をばらまいているものです。一方、タスクとは言語であり、概念的意味的には豊富で詳細な情報は含められますが、物よりも情報の広さ（非言語的な情報の広さ）に乏しく、認知に労力を使いやすいです。

Richild が成立するのは「物」を扱っているからです。デスクトップは電子的な世界ですが、アイコンがありますよね。アイコンは絵であり、非言語的な情報が豊富です。物寄りです。仮に、これがただの文章や単語の羅列だとしたらどうでしょう。成立しないと思います。TODO リストが破綻するのもまさにそうです。タスクという言語的な情報は、一箇所に集めるだけでは中々機能しません。人間にその能力がないからです。だからこそ個人タスク管理が必要で、色々苦心

するのです。

そういうわけで、Richild を心がける人は「物タスク」だけを、あるいはデジタルの場合でもなるべく非言語が豊富な情報を扱うようにしてください。言語で書かれたタスクを扱おうとしても、おそらく失敗します。一応、タスクを一箇所に集める戦略も無いことはなくて、それが次の Monolith だったりします。

メリデメ

メリット:

- 手間暇がかからないし、疲れにくい
- 一箇所に集めているので「この中にある」という安心感がある

デメリット:

- 頭の性能でカバーできないと破綻する

適性

頭の性能が高い方に適しています。頭の性能の定義や測定方法を論じるのは難しいですが、筆者は IQ が端的な指標になると思っています。これも筆者の印象ですが、Richild で仕事ができてて有能と評価される人は、IQ でいうと 110（上位 25%）を超えているイメージがあります。このあたりは本格的に調べてみたいところです。

タスク管理を行う気がない人にも適しているでしょう。一応本書は読んでみているものの本心ではその気がないとか、タスク管理などたかが知れていると考えているとか、タスク管理を下に見ているといった人達を指します。頭の性能があって、それで打開できることを知っている・経験している人達です。このような人達は自身の性能に頼ればいいので、タスク管理などという面倒くさい営みに対する忌避感が強くなります。逆に、そうではない人はタスク管理の必然性がわかるので、（面倒くさいと感じはしますが）忌避感はありませんし、期待もします。前者の忌避感があるとその気がなくなるわけですが、それはそもそも性能が高いからそうなっているのだと筆者は考えます。大体に言えば、忌避感の強さと Richild の適性には相関があると思います。

発達障害、特に ADHD の人にも適している可能性があります。多動性があると自身で脳内の指向性を制御できませんが、散らかったおもちゃ箱の中はそれなりに情報量が多いため、脳内の「持て余し」を上手く費やしてくれます。とはいえ優先順位まではフォローできないため気休めにすぎないのですが、箱やその中身の傾向を工夫すれば「なんかよくわからんけど意外と回っている」を実現できるくらいの余地はあります。

Monolith

概要

タスクを一箇所に集める戦術です。

Monolith とは一枚岩の意味であり、IT の文脈では「何でも詰め込んだあり方」のたとえとして用いられることがあります。この場合は、タスクを一箇所に詰め込んだあり方を指します。具体的には **一つのテキストファイルに** すべてのタスクを書きます。エンジニアやクリエイターなどテキストエディタによるテキストの書き込みに長けた人がいますが、この人達が、自分が扱う文章に対して Richild のような発想を行おうとすると、自然と「一つのファイルに書く」に行き着くことがあります。

例を示します。たとえば stakiran.txt というファイルを一つつくります。stakiran は筆者の名前なので、筆者から見れば私に関するすべてが入っているニュアンスです。内容は ■ で区切って「一つの塊」にしながら書いていきます。

```
■今週のタスク
...

■引っ越し検討
...



■ ---


■2024/05/24
...

■2024/05/23
...


■2024/05/22
...
```

上の方が作業領域になっており、■ --- で区切られた下の方が過去の記録だと想像できますね。基本的にはこれだけです。このファイルを生活導線として、何でも書き込みながら日々過ごしていきます。タスク管理については、原始的に一覧化したり、終わったものは下に移すなり stakiran_old.txt など別ファイルに退避させたりします。原理は単純ですし、これで本当にまわるのかと思いがちですが、意外とまわるのです。特に Monolith は文章入力スピード、言語化のスキル、テキストエディタの操作やカスタマイズの知識を備えているので、手足のように自在に操れます。テキストエディタ自体も強力であり、数百万文字くらいなら耐えられます。管理

対象も stakiran.txt ただ一つなのでバックアップや同期もラクチンです。

上記はあくまで一例です。他にも人の数だけやり方があります。筆者は普段秀丸エディタを使っており、実践入門なる電子書籍を書いてしまうほど傾注しているのですが、一時期は Markdown 記法ベースの Monolith で仕事やプライベートを回していました。見出し記法を区切りとして用いることで目次化や見出し間移動が可能となり、塊単位でザックザック移動していました。これはアウトライン機能とも呼ばれ、デジタルメモ「ポメラ」も搭載しているほど便利なものです――と、このようにテキストエディタとカスタマイズの沼に陥りがちなのも特徴ですが、ともかく、Monolith は一つのテキストファイルにタスクを（タスクも）集める戦略なのです。

応用1: タスクだけか、タスク以外の情報も管理するか

Monolith でどこまで扱うかに好みが出ます。

具体的には、タスクだけを扱った **Task Monolith** にするか、割とあらゆる情報もぶっこんだ **Note Monolith** にするかです。Task Monolith でも、タスクに関する調べ事や考え事はするので色んな情報は書き込みがちですが、Note Monolith はその日ではなく、たとえば毎日の日記や、ネットを見てて気になったこととか、仕事や私生活の問い合わせで使う下書きなども全部書き込むほど重用します。

昔は SaaS がさほど発達しておらず手段が乏しかったため後者が好まれがちでしたが、今では色んなツールがあるため前者派が多いと思います。少しネットで検索してみました（※1）が、

- プレーンテキスト最強に同意 | トドの日記2.0
- 様々なTODOアプリやタスク管理方法を試行した結果最終的にプレーンテキストに行き着いた話 - みんからきりまで
- なにもかもシンプルに。私がいまだにプレーンテキストを愛用している理由 | ライフハッカー・ジャパン

見た感じ、前者の Task Monolith が多いですね。

- ※
  - 1 検索するときは「プレーンテキスト」なるキーワードを使うと良いでしょう。プレーンテキストとはソフトウェアによる装飾の無い、生のテキストのことで、どのツールでも使える、たとえばコピペできる汎用性があります。詳しい話は後の章に譲ります（プレーンテキストタスク管理）

応用2: クラウドツールで Monolith は可能か

結論を言うと、難しいです。

Monolith では圧倒的な操作性――手足に馴染んでおり 0.1 秒の違和感もないことが **前提** です。テキストを自在に扱うためには、それくらいスピーディーかつ快適でなくてはならないので

す。そうでないとストレスでとてもじゃないですがやってられません。もっというと、それを実現するだけのスキルも要ります。

さて、昨今ではクラウドによるツール、つまりは SaaS が当たり前になりましたが、SaaS にはオンラインであることによる微妙な遅延があります。また、テキストエディタほど柔軟なカスタマイズ性も備えていませんし、数万文字はまだしも数十万文字レベルの巨大なテキスト量をスムーズに扱える安定感もまだありません。SaaS が本格化して十年以上経ち、様々なノートツールも出ていますが、元 Monolith の筆者に言わせれば、まだまだテキストエディタの域には至っていないのです。そもそもそんな時代でもないでしょうし、今後は生成 AI により人間は書くよりも（AI が出したコンテンツを）見る・微修正する機会が増えるでしょう。そもそも回線技術が足かせとなっています。今後も至ることはないと思います。

応用3: テキストだけでは扱いづらいタスクをどうするか

たとえば予定や日課・習慣を Monolith で扱うのは面倒くさいです。

通常は別ツールを使います。予定は言わずもがなカレンダーを使うのがベターですし、日課や習慣は Todoist や タスクシュート など「定期的なタスクを扱う能力を持ったツール」、あるいは習慣なら習慣トラッカーを使うのが理に適っています。ちなみに筆者も、Monolith の時代でも別ツールを使っていましたし、何ならつくっていました。

どうしてもテキストエディタでやりたい、Monolith でやりたいというのなら、それができるだけのカスタマイズや機能の自作をご自身で行うことになります。Topician と同様、自分の力で何とか工夫しなければならなくなります――が、試行錯誤してきた先人から言わせてもらうと、かなり分が悪いです。要はカレンダーや定期タスク機能で扱っているようなものを、何とかして単純なテキストの世界に落とし込むわけで、分が悪いどころか無理のレベルです。まるで三次元の立体を二次元の平面に変換しろと言われているような感覚です。できるにしても、まめな振り返りやメンテナンスにより手作業で無理やりカバーすることになります。DIY のようで楽しくはありますが、手段の目的化感は否めませんし、純粋に面倒くさくてキツイです。

メリデメ

メリット:

- テキストファイルで一つで済むシンプルさ
- テキストエディタの扱いに長けたベテランにはフィットする

デメリット:

- テキストエディタ、タイピングスピード、言語化など非常に軽いフットワークが要求されるため、合う人が少ない
- 時代の流れには沿っておらず、今後の発展を見込めない
- 予定、日課や習慣などただのテキストでは扱いづらい性質のタスクを扱いづらい

適性

日頃からテキストエディタを愛用している方には適している可能性があります。目安は「普段文字入力に用いる手段としてテキストエディタが最も多いかどうか」です。そうではない場合、たとえばスマホがメインだったり、クラウドツールがメインだったりする人はおそらく適していません。そのような手段が普及した現在でも、あえてテキストエディタを使っているほどの重用っぷりでなければ務まらないと思います。

エンジニアやプログラマーの人には向いていません。意外に思われるかもしれませんが、エディタや入力速度の強さよりも、自分が書いたテキストを管理、修正していくことを楽しめる感性が大事です。通常、彼らは技術やコーディングには傾注していますが、テキスト自体には傾注していません。そういう意味では、ブロガーや作家の方がまだ向いている可能性があります。

繰り返しの力でまわす

タスクを「繰り返し着手する対象」と捉える戦略です。自分のリズムを尊重したい人に適しています。

前提知識: ルーチンタスク

ルーチンタスク（Routine Task）とは、定期的に実行するタスクを指します。

定期的とはたとえば毎日行う、2日に1回行う、週一で行う、毎日朝と夜と2回行う、などです。

タスクは基本的にルーチンタスクとして捉えることができます。ごみ捨ては毎週特定の曜日に行うでしょうし、メールやチャットのチェックも「1日2回行う」と決めればルーチンタスクになります。日課や習慣もルーチンタスクの一種ですし、新しく読みたい本や始めたい行動なども「3日に1回くらい時間つくってやるか」とすればルーチンタスクです。

わざわざ小難しい概念を導入する理由は、タスク管理しやすいからです。タスク管理ツールには定期性のあるタスクを扱う機能があり、これに頼ることでルーチンタスクをまわしていけます。本節の戦略は、この機能に全面的に頼るものです。

Robot

概要

「このとおりに動けば一日が上手くいく」というレベルで詳細なタスクリストをつくり、それに従って行動する戦略です。

人の生活の大半はルーチンタスクに帰着できるため、Robot におけるタスクリストには多数のルーチンタスクが並びます。最初からそんな都合の良いタスクリストを捉えることはできないため、記録や仮説に基づいて、少しずつ明らかにしていきます。定着まで長い道のりを要し、一ヶ月どころではありません。

例として、仮に筆者が Robot を採用するとした場合、どんなタスクリストになるかを書いてみます。ルーチンタスクには □ をつけており、括弧内に出現頻度を書いています。

-5:00 □（毎日）睡眠おわり、起床
-5:04 □（毎日）トイレとカーテンと顔洗い
-5:07 □（毎日）お湯を沸かす、マットレスを片付ける
-5:10 □（毎日）コーヒーとパンを持ってデスクに
-5:40 □（毎日）情報のキャッチアップ、コミュニティへの返信
-5:50 □（平日毎日）勤務準備
-5:50 □（火・木）可燃ごみ ごみ捨て
-5:55 □（平日毎日）リモート勤務勤務開始
-6:10 □（毎日）今日のカレンダーを見て 10 分前にアラームをセット
-6:10 □（平日毎日）勤務開始連絡
-6:10 □（平日、2日に1度）勤怠入力
-6:10 □（平日毎日）チャットとメールを読む、アクションが必要なものはタスク化だけする
-6:20 □今日やることを組み立てる
...

個人的すぎる内容のため理解は放棄して構いませんが、雰囲気はおわかりいただけるかと思います。見ての通りルーチンタスクばかりです。通常のタスクが登場するのは 6:20 以降です（今日やることを組み立てた結果のタスクが並びます）。決まったリズムで動く様はまるでロボットですね。病気に見えるかもしれませんが、筆者にとっては自然です。むしろ自分の過ごし方を洗い出しただけにすぎません。他の人も、自覚がないだけで自分なりのリズムがあります。Robot はそのようなリズムを明示的に捉えて、タスクリストに落とし込んで、改めてその通りに動くことを目指します。

その結果、Robot が手に入れるのは安定感の継続です。仮に自分のリズムに完璧に沿えることがベストだとするなら、ベストに至るためには「できるだけ沿えるように過ごせ」となりますよね。これを意図的に狙うのです。

Robot を行うためには、ルーチンタスクを扱えるツールが必要不可欠です。「今日はどのタスクを行えばいいのかな」といちいち手作業で判断するのは現実的ではありません。ツール側でタスクごとに頻度を設定しておけば、そのとおりに自動で出現させることができます。国内ではタスクシュートが火付け役となりましたが、Todoist などシンプルなツールでも扱えます。しかし、カレンダー系のツールでは力不足です。Robot は一日何十という（ルーチン）タスクを扱い、何百何千という操作も行ないますので、これができるだけの軽さがないと挫折します。人によってはタスクシュートや Todoist でも足りませんし、筆者も Monolith の延長で Robot を行うツールを

自製していました（※1）。

- ※
  - 1 詳細は割愛しますが Tritask と todaros です。どちらもテキストエディタベースでルーチンタスクを扱います。一応、他者も理解できるよう整備しているつもりですが、万人向けではありません。もっとも、これは筆者が Monolith 寄りのテキストジャンキーで、かつスマホを持っておらず PC で生活を完結させる仙人だからでしょう。スマホを常用している方はスマホアプリでまかなえると思います――が、いずれにせよ、上述したとおり長い道のりになります。

応用1: どこまで固めるか

Robot としてどこをどこまで扱うかには好みが出ます。

- 時間帯
  - 朝起きてから寝るまでのすべて
  - 勤務中だけ扱うことにして、プライベートでは一切扱わないようにする
  - 逆にプライベートだけ扱うことにして、勤務中は扱わないようにする
  - 平日の出勤前と退勤後という慌ただしいタイミングのみ etc
- タスク
  - 掃除、買い物、整理整頓などこまめにやっておかないと首が締まっていく系の日常タスク
  - 日課や習慣全般
  - チャットやメールの確認、RSS や SNS の確認、読書などコミュニケーションやインプット系のタスク
  - 勤務中の定型作業全般 etc
- 実行の記録
  - いつ開始していつ終了したかまで記録する（タスクシュートはここ）
  - いつ終了したかだけ記録する
  - 終わったかどうかだけ記録する
  - 記録すらしない（終わったら消す、筆者の todaros はここ） etc
- 順序性
  - タスクリストは必ず上から順に消化しなければならない、とする
  - 基本的に上から消化したいが、別に飛ばしてもいい、とする
  - 今日行うタスクは表示している（デイリータスクリスト）が、どれをいつ行うかはその場の気分で選ぶ
  - リストによる表示さえもやめて、単に今日やるタスクが集まる状態にする etc

つまり生活全域に適用するのか、一部にのみ適用するかということです。

全域への適用はもはや趣味だと思います。まずは少しずつ適用していくのが良いでしょう。あるいは、どうしても早く突破したい「嫌な時間帯がある」という場合に、最短で駆け抜けるために

Robot を適用するのもアリです。成し遂げたいことがあるため習慣化したい、という場合も向いていますが、Robot は過剰ですので、後述の Tracker が良いでしょう。

応用2: 強いロボットと緩いロボット

Robot を見ると、タスクリストのとおりに厳しく従うイメージがありますが、そうとも限りません。というより自分で選びます。

厳しく従うスタイルを **強いロボット** と呼びます。安定感は抜群ですが、融通が利かなかったり少しでも乱れるとストレスが溜まったりする点がデメリットです。また、人をかなり選びます。強いロボットを運用できるのは、20 人に 1 人もいないのではないでしょうか。おそらくは必要性があって、かつ意思も強くて実践できる人か、そうでなければ単に神経質・几帳面・その他こだわりが強い人だけでしょう。

一応タスクリストは洗っているが、従うかどうかはゆるく決めるスタイルを **緩いロボット** と呼びます。緩いロボットでは、「今日ごみ捨ての日だけどだるいから次でいいか」のような先送りも平然と行います（その可燃ごみ捨てタスクを次回日に先送りする操作は必要です）。強いロボットのような安定感とは無縁になりますが、融通も利きますし、サボることはあっても忘れることはないので精神的にかなり楽できます。

「だったら Robot なんてしなくていいのでは」と思われるかもしれませんが、タスクリストが並んでいること自体に意味があります。Richild や Monolith と同様、一箇所に集まっているのです。**生活において必要なルーチンタスクのすべてが集まっており、しかも適切な頻度で出現するという安心感** は非常に心強いものがあります。逆を言えば、すべて集めることと適切な頻度で出現されるようにするチューニングは、自分で行わないといけません。この点に限っては緩いロボットであっても避けられません。

応用3: リズムにも色々ある

Robot は自分の生活リズムを可視化する戦略とも言えますが、リズムにはいくつか種類があります。

- タイミング
  - いつ実施するか、というものです
  - 可燃ごみのごみ捨てを例にすると、朝 6 時が適している人もいれば、朝 9 時が適している人もいます
  - またタスクによっては制約もあります。可燃ごみは 8:30 までに出してください、などです
- 頻度
  - ルーチンタスクは「n日に1回行うタスク」ということもできます、この n を決めよ、というものです
  - 筆者はメールチェックを 2 日に 1 回としています（n=2にしている）が、毎日行ないたい人もいるでしょう

    - 1日に複数回行ないたい場合は、ルーチンタスクを複製します
        - メールチェックを 1 日 3 回行ないたい場合、n=1 のルーチンタスクを 3 つつくります
    - 頻度も制約によってコントロールされる場合があります（よくあります）
        - リモートワークでは毎日出勤報告が必要と思いますが、これは事実上「毎日行うルーチンタスク」です
- キャパシティ
    - どんなタスクをどれだけ詰め込めるか、というものです
    - 人それぞれにモチベーションや認知資源のクセと限界があり、ここを無視するとタスクリストが形骸化します
        - 忘迷怠でいう「怠ける」がまず発動し、形骸化が常態化すると認識もされなくなるので「忘れる」も生じるようになります
    - たとえば筆者は「業界の情報をキャッチアップする」ルーチンタスクを使っていますが、頻度は 3 日 に 1 回です
        - これ以上増やすと苦しいからです
        - かといって頻度を減らしすぎると、逆に面倒になって形骸化します
        - バランスの良い塩梅が n=3 でした（それでもたまにサボるので実質 4〜5 日に 1 回くらいになってますが）

リズムはすぐにはわからないものですが、仮説検証的に微調整するといいでしょう。タスク管理ツール上では仮決めしておき、そのとおりに動いてみて、よさそうなら採用、ダメそうなら微調整するのです。n=2 だと多すぎるから n=3 に増やすかとか、タスク A と B と C を毎日（n=1）午前でやってたけど、ちょっとしんどいので C だけ午後にして、かつ n=2 に落としてみるかとか、といったことです。

こういう微調整ムーブを息するように行えるかどうかが、Robot 定着の分水嶺になると思います。微調整するのがそもそも面倒くさい、続かないといった場合、まずはツールを疑ってください。次にトレメントを思い出して、トレーニングのつもりで頑張って耐えてみてください。それでもダメなら、向いてないので諦めましょう。Robot はかなり人を選ぶ戦略なので、できなくても不思議ではありませんし、むしろできる方が少ないと思います。

メリデメ

メリット:

- 自分のリズムに則った、安定感のある生活をおくれるようになる
- すべてのタスクが適切な頻度で出現するという安心感

デメリット:

- ハードルが非常に高い
    - 人をかなり選ぶ上に、定着の道のりも長い
- 一生微調整の世界観に身を投じることのしんどさ

- 筆者も一時期 Robot でしたが、これがしんどくて今では（部分的には使ってますが）やめてます

適性

毎日同じリズムで規則正しく過ごしているような几帳面な人には向いています。

仕事と家事育児を両立するような忙しいビジネスパーソンにも向いています。意外に思われるかもしれませんが、この人達は（単に忙しさがひどいだけで）生活リズムが固定化される傾向にあります。自身のリズムを可視化できれば、それに従うことが「模範解答的な過ごし方」となるので、忘迷怠も防げて非常に楽（いかに従えるかというゲームに帰着できます）になります。問題は、Robot を始めたりツールを操作したりする余裕すら取れないことでしょうか。これを取るためには盤外戦が必要です。ハードルはもう一つあって、PC デスクの前に座れるとは限らないので、座れるように盤外戦をするか、スマホで操作できるほど習熟する必要もあります。頑張れば手帳や掲示などアナログな手段でも実現できるかもしれません。

決めたとおりに行動できない人――多動な人やずぼらな人には向いていません。すでに述べたとおり、タスク管理の本質はトレメントであり、これをやると決めたことをそのとおりにこなすのが基本になるのですが、特性上、そういうことが苦手な人がいます。特に Robot は戦略の中で最もトレメントが求められるものですから、絶望的に向いていません。そもそもやろうとも考えないでしょうが。

Tracker

概要

タスクを「習慣タスク」として捉える戦略です。

習慣タスク とは時間を気にしないルーチンタスクのことです。何時に行ってもいいですし、何時に行ったかを記録する必要はありません。必要なのは頻度の設定と実行したかどうかの記録だけです。言い換えると、Tracker は毎日「いつやってもいいけど今日中にやるタスク」が書かれたリストや表と向き合う生活になります。Dailer と似ていますが、先送りの概念はありません。Richild や Monolith とも似ていますが、集めるものが習慣タスクのみという点で違います。

と、概念的には小難しいですが、習慣トラッカーがわかりやすいです。Tracker は習慣トラッカーを使います。これを使って、なるべく多くのタスクを処理することを目指します。通常、習慣化のために習慣トラッカーを使う人の「一日にこなす習慣数」は多くても十数個ですが、Tracker はもっと増えます。20、30、下手すればそれ以上もありえます。

本質的には Robot と同じです。普段の生活において必要なタスクをルーチンタスクとして捉え尽くす点は同じで、これをどう処理するかが違います。Robot は愚直に「定期性を持つタスク」として全部ツールに入れて管理――生涯微調整する道を選びましたが、Tracker はもっと雑

で、単に習慣トラッカーに全部載せてしまえと考えます。

例として、Robot の項でも挙げた筆者のサンプルを、今度は Tracker 用に書き直してみましょう。

	5/27 月	★ 5/28	5/29	5/30	5/31	6/1 土	6/2
トイレとカーテンと顔洗い	✅	✅					
お湯を沸かす、マットレスを片付ける	✅	✅					
コーヒーとパンを持ってデスクに	✅	✅				-	-
情報のキャッチアップ、コミュニティへの返信	✅	✅					-
勤務準備	✅	✅				-	-
可燃ごみ ごみ捨て	-		-	-		-	-
リモート勤務勤務開始	✅	✅				-	-
今日のカレンダーを見て 10 分前にアラームをセット	✅	✅					
勤務開始連絡	✅	✅				-	-
勤怠入力	-		-		-	-	-
チャットとメールを読む、アクションが必要なものはタスク化だけする		✅				-	-
今日やることを組み立てる	✅						

今日が 5/28 火曜日だとして、まだ終わっていないタスクとして「可燃ごみ ごみ捨て」「勤怠入力」「今日やることを組み立てる」がありますね。Robot の時は上から厳密に処理していましたが、Tracker は違います。**その日のうちのどこかでできればいい** と考えます。Robot でいう緩いロボットと同じです。ここで「だったら緩いロボットでいいじゃないか」と思われるかもしれませんが、緩いロボットでも Robot には変わりなく、タスク管理ツールへの深い習熟（と高頻度なタスクの操作）が求められます。もっと雑にやりたいのです。習慣トラッカーであれば、雑にやれます。最初にこのような表をつくるのは面倒ですが、一度つくってしまえば、あとは毎日、やったところにチェックをつけるだけで済みます。律儀に一つずつつけなくても、あとでまとめてこれとこれはやったよなとチェックすればいいのです。雑に運用しながらも、今日は何をすればいいかが可視化されているというバランスが手に入ります。

応用 1: ツールとして何を使うか

大まかに 3 通りあります。

- 1: 手帳などアナログな手段を使う
- 2: 既存の日課・習慣管理系のツールを使う
- 3: 自製する

1 のアナログな手段については、習慣トラッカーの鉄板です。一から自分でつくってもいいですし、時系列軸のある手帳を使ってもいいですし、習慣トラッカー用の手帳を使うこともできます。メリットは手軽に始められることです。デメリットは、タスクの追加・編集・削除といったメンテナンスがしんどいことです。仮に 1 日平均 30 のタスクがあるとした場合、習慣トラッカーの縦軸には 30 のタスクが並ぶことになります。これを全部手で書かないといけません。

2 のデジタルな手段については、既存の日課管理や習慣管理系ツールを使います。この手のスマホアプリは多数ありますし、Habitica のような本格的なもの（習慣トラッカーというよりは Robot 寄りの世界観）もあります。メリットはアナログよりも各種操作や記入が楽に行えることです。デメリットはツール側のコンセプトや広告が煩わしいことです。また Tracker のように何十ものタスクを扱うことが想定されていなかったりします、たとえば今日全部のタスクを一画面に表示しきれず高頻度なスクロールを求められたりします。

と、このように既存で決め手に欠けると、3 の自製案に行き着きます。エンジニアとして自分でツールをつくるのもいいですし、そこまで行かずともテキストファイルや手帳などで自分なりのやり方・フォーマットを確立してしまうのもアリです。たとえば家の壁にマグネットがひっつくボードを設置して、タスクや時間軸をマグネット付箋で書いて並べてしまうこともできますし、デジタルだとMiroのようなホワイトボードツールで同等のことができます。習慣トラッカーは表ですから、Excelでも可能です。いずれにせよ、縦長にはなるでしょうから、それなりのサイズが必要だと思います。

応用2: どこまでカバーできるか

戦略というからには、なるべくすべてのタスクを管理したいものですが、残念ながら Tracker では力不足になりがちです。

まず、その日その時にスポットで発生するタスクや予定――つまり定期性のないタスクは管理できません。十中八九、他の戦略も使うことになります。

次に、あまりタスクを管理しすぎると形骸化に繋がりやすくなります。たとえば 1 日平均 30 個のタスクを Tracker で管理しても、忘迷怠、特に怠けることが発動してあまり消化できません。Robot の項でも少し書きましたが、認知資源がバカになりません。結局、消化率を上げるためには、Robot など厳しめの制約が要ります。本書でも度々強調していますが、人間は怠け者であり、「この中に今日やるタスクが集まってるよ」「どれをいつやってもいいけど今日中に全部終わらせたいな」程度では終わらないものです。30 個中、10 個くらいしか消化できてない日々が続く、みたいな怠けた生活になる可能性も十分あります。

別にそれでもいいじゃないかと思われるかもしれませんが、人間は強くないので、1/3 しか出来てない、チェックできてない空欄が多い、との現状を目の当たりにするとメンタルをやられます。あるいは、やられたくなくて目に入れる機会から逃げて――となって、ついには破綻します。これは重要な原則の一つですが、**タスク管理では形骸化が常態化すると、すぐに破綻します。**穴の空いたバケツのようなものです。そもそも形骸化を防がなくてはなりません（穴空き自体を許してはなりませんし、空いてもすぐに塞がねばなりません）。

そういう意味で、Tracker にもバランス調整が要求されます。習慣化と同様、少なめのタスクから始めて、少しずつ増やしていくのが良いでしょう。あるいは、最初から多数のタスクをこなしたいとわかっているのなら Robot を検討してください。

応用3: 習慣トラッカー以外のビューはあるか

あります。

習慣トラッカーは、ビューの種類で言えばテーブル（表）にあたりますが、ビューとしてはリストやボードもありますし、チャートやネットワークやセンテンスもありえるかもしれません。

いくつか挙げましょう。

- スタンプカード式（ボード）
  - ラジオ体操のスタンプカードを思い浮かべてください
  - あれば「ラジオ体操に参加する」という単一のタスクのみを管理する習慣トラッカーと言えます
  - これを一般化すると、1タスク1トラッカーとなります
  - 仮に 30 のタスクを扱いたいなら、30 枚のトラッカーをつくればいいのです
- センテンス式（センテンス）
  - 一日の理想の過ごし方を日記風に書きます
  - 本文は、
    - 何度読み返してもうっとりするよう、美しく仕上げます
    - 行うべきタスクは「」や『』などで強調して散りばめます
  - このような文章があると、毎日それを見ることで今日必要なタスクがわかります
  - 実施したかどうかを記入する（打ち消し線を引くなど）するために、複製して毎日印刷できると良いですね
  - 曜日ごとに過ごし方が違うのなら、その分の文章をすべてつくりましょう
- スリーエリア式（ボード）
  - 準備
    - マグネット付箋など移動が容易な文房具（デジタルでも可）
    - 付箋を置くエリアを 3 つつくる
    - エリアは「プール」「今日やる」「今日やった」の 3 つ
    - タスクは付箋に書き、プールに集める
  - 運用
    - 1: 一日の最初に「プール」を見て、今日やるものを「今日やる」に移す
    - 2: 日中は「今日やる」を見ながら過ごす、終わったものは「今日やった」に移す
    - これを毎日繰り返す
    - その際、前日残った分が「今日やる」に残ってるので、これは 1: のときに引き続き残すかプールに戻すかを選ぶ
  - Robot にせよ、Tracker にせよ、タスクの実施頻度はツール側に設定しますが、このスリーエリア式は 1: を見ればわかるとおり、自分の記憶と判断を頼りにします

上記は例です。この他にも、自分に合ったビューを考えることはできると思います。

メリデメ

メリット:

- Robot よりも雑な運用で、Robot をメリットを受けられる

デメリット:

- 単発のタスクを扱えない
  - 別にタスクリストをつくるなどして、そちらを使う必要があります
- 専用のツールや事例がなく、整備が落ち着くまでに試行錯誤を要する可能性
- タスク数のバランス調整に苦戦する可能性
  - タスクの消化を自分の意思に任せる構造であり、形骸化→破綻につながりやすいため、そうならないよう抑える

適性

Robot に挫折した人には向いているかもしれません。Robot でも緩いロボットのやり方はありますが、ツールの重用自体は避けられません。一方、Tracker であれば、習慣タスクとして捉えるというクセこそありますが、ツールの難易度自体はたかが知れています。

習慣化が好きだったり趣味として取り組んでいる人達（**ハビタン, Habitan**）には向いていません。意外に思われるかもしれませんが、ハビタンは毎日すべての習慣をこなす完璧性を求めています。一方、Tracker の本質は可視化と手抜きです。頑張るのは最初、習慣トラッカー上にタスク（といつやるかの頻度）を並べて可視化するところまでで、その後毎日どこまで消化するかは雑でも良いのです。「半分以上チェックがつきませんでした」と平然とありえます。ハビタンには耐えられないでしょうし、むしろ「毎日すべてのタスクを全部消化しなければ！」と真面目に取り組もうとするでしょう。正直分が悪いです。Tracker のレベルでちゃんと消化したいなら、素直に Robot を使うべきです。

自分の脳内が正義

ツールの利用を最小限にする戦略です。タスク管理やツールといった細々にとらわれたくない人に適しています。

Inboxer

概要

タスク管理としてインボックス（未消化の対象を溜める何か）のみを使う戦略です。

明示的に新しいツールや方法論を整備するのではなく、既存のコミュニケーションツールや環境

の一部を拝借してインボックスにします。たとえばメールの受信箱、ブラウザやチャットのブックマーク機能、あとで処理する書類を置いておくスペースなどです。最後のスペースについては、律儀に引き出しやトレイをつくるというよりもどこか適当なスペースをそうだとみなすことが多いです。インボックスは一つではなく、必要に応じて複数存在し、Inboxer はインボックスゼロ――つまりは全部処理して中身を空にすることを目指します。

Inboxer は極めて合理的であり、タスク管理を行っているとの自覚はありません。業務上あるいは生活上必要な動線の中で、一時的に手間のかからないマーキングをしている、くらいの手間しかかけません。その心理は繊細で、自身の頭の性能への絶対的な自信と、そうは言っても全部を覚えたり優先順位をつけたりするのには限度があるので楽をしたい気持ちと、それでもタスク管理なるものに頼るのは面倒すぎるしプライドも許さないという信念とが絶妙なバランスを保っています。その結果、普段使ってる動線が持つ機能を使ってマーキングをする程度、となります。メモではなくマーキングです。いちいちメモを残すなんてことはせず、ボタン一発でマークをつけておくくらいの手間がギリギリの許容範囲なのです。

応用1: Calendarerとの使い分け

Inboxer は忙しい人がなりがちです。他に忙しい人向けの戦略としては Calendarer があり、使い分け方に悩むことがあります。

ポイントはカレンダーが動線になっているかどうか、です。動線とは自分が自然に一日何回も通る場所のことです。この場合、毎日自然と何回もカレンダーを見るかどうかです。見る場合で、かつ予定を扱う必要もあるのなら Calendarer も必要でしょう（要はカレンダーが必要でしょう）。このとき、カレンダーをインボックスとして扱うこともできます。たとえば、あとで行うタスクを、普段使わない時間帯に予定として自分でつっこんでおけばいいのです。その時間帯がインボックスであり、ここを空にすれば良いと考えます。

応用2: アクティブレイジー

Inboxer の屋台骨は「既存の」動線を死守することです。自分で新しくインボックスをつくったり、ましてタスク管理のシステムをつくったりなどしません。そんなことはやってられません。たとえば Slack と X と Discord の 3 つを使っている Inboxer がいたとしたら、この人にとっての既存の動線はその 3 つであり、その 3 つが持つ機能のみでインボックスをつくります。手元の手帳でタスク管理しようとか、Notion でノートを取ろうといったことはしません。「新しく仕組みをつくって動線にしちゃえばいいじゃないか」と思われるかもしれませんが、そういう行動も好みません。この人にとっての動線はあくまでもこの 3 つであり、自分でいちいち増やすだなんて真似はしないのです。

そういう意味で、Inboxer は怠け者（Lazy）と言えます。一方で、動線上で多数のコミュニケーションをバリバリこなしており、人一倍優秀だったり立場や収入を得ていたりする傾向もあります。アクティブなわけです。一見すると相反してそうな、このような Inboxer の特徴を **アクティブレイジー（Active Lazy）** と呼ばせてください。

Inboxer の気質はアクティブレイジーです。もし「自分用の新しいタスク管理をつくろう」などと考えるのだとしたら、その人は Inboxer 気質ではないでしょう。他の戦略を採用した方が賢いと思います。

逆にアクティブレイジーだと思われるのなら、既存の動線からは外れないようにしてください。外れようとしても、どうせ続きません。アクティブレイジーは怠け者なので、他の戦略なんてできやしません。もちろん状況次第では既存の動線が廃止されたり、新しい動線が追加されることもあるため、その場合はさっさと勉強して適応してください。

応用3: 場や人の力にどれだけ頼るか

※インボックスとは関係のない話になりますが、タイミングが良いのでここで解説します。

Inboxer はアクティブレイジーであり、タスク管理を好みません。Inboxer として自分で管理するのも良いですが、実は外に任せるのもアリです。人に委譲したり、場に居ることで働きかけてもらったりといった形で楽をするのです――とわざわざ言うまでなく、経営者、管理者、その他権力者などはこのスタイルを使いがちです。秘書を雇うなど雇用とは違う点に注意してください。

外部に頼ることをどれだけ狙うかに好みが出ます。おおまかには以下の 4 パターンがあると思います。

- クローズドな委譲者
  - 人に頼るタイプです
  - 信頼できる少人数（たいていはひとり）に任せます
  - 昔からよく知られた概念で、任さる先として秘書、補佐、マネージャー、パートナー等があります
- オープンな委譲者
  - 場に頼るタイプです
  - 自分の情報を社内に広く公開し、社員の誰かが巻き取る・必要に応じて干渉してくれることを期待します
  - ティール組織など現代的な組織の CEO が、この形態を取ることがあります
- クローズドな独裁者
  - 頼らないタイプです
  - 情報を自分のもとに集め、自分でバッサバッサと判断していきます
  - 関係者のみを集めた会議や、関係者しか見えない DM などクローズドな手段を好みます
  - オープンな場所や機会で発言することもありますが、たいていは決定事項の伝達です
- オープンな独裁者
  - 頼らないタイプです
  - 自分で判断する点はクローズドな独裁者と同じですが、手段はオープンであり、皆に見えるチャットチャンネルやメーリングリストなどでも平然とやりとりします
  - 一見するとオープンなので委譲的に見られがちですが、本質は頼らないタイプであり、

何に反応するかもその人自身が決めます

Inboxer 本人としては、自分がやりやすいやり方で良いでしょう。方法論的に優れているのはオープンなあり方です。クローズドなやり方だと情報格差と政治が発生し、中長期的な目線では分が悪かったり優秀な人材が集まりづらかったりします。どこかで伸び悩みます。組織の力学としては自然ではあります。

Inboxerと関わる側としては、自分に合ったタイプと過ごせると良いでしょう。クローズドなInboxer は IT に疎く、打ち合わせと政治が多発しがちです。懐に入れたら天国ですが、入れなければストレスです。また、基本的に拘束が長くなるためワークライフバランスは通用しづらくなります。オープンな Inboxer は公平で公正でフランクですが、変化や議論が激しく信念等のクセも強くて、合うなら天国、合わなければ地獄です。

メリデメ

メリット:

- アクティブレイジーの気質（繊細なバランス）を尊重できる唯一の戦略であること

デメリット:

- 人を選ぶ戦略であり、採用しづらいこと
  - アクティブレイジーの気質、頭の性能、動線から逸脱しなくても済むことが許される立場等が必要
- 対応が場当たり的になること

適性

経営者、組織長、指導者など人や組織の意思決定を行う立場の人には向いています。Calendarerと同様、誰に言われずともこの戦略になることも少なくありません。

フリーランスなど裁量多め、制約少なめで多くの仕事をこなす立場の人（ラピッド・マルチタスカー）にも向いています。ただし場当たり的な対応しかできませんから、自らを忙しくしがちで、慢性的に不調を抱えがちです。

アクティブレイジーでない人には向いていません。

研究者、作家や芸術家などクリエイティブな人にも向いていません。ただし場当たり的にこなしていける実力とそれで満足できる感性があるなら別です（ラピッド・マルチタスカーの範疇になります）。

まとめ

- 戦略とは
  - どんなやり方や考え方で望むかという方針のこと
  - 何を拠り所にするか、で分類できる
- ===
- 時間で区切る
  - メリハリをつけたい人向け
  - Dailer。今日とそれ以外に二分し、今日は何をするかを考える
  - Calendarer。カレンダーを片手に、タスクを予定化して予定消化マシーンと化す
  - Slotter。時間帯という枠を設け、各枠で何をするか考える
  - 主な用語: オプラン
- 話題で区切る
  - 中長期的に取り組みたい、寝かせたい人向け
  - Issuiest。バグ管理システムを使う
  - Topician。Wikiやノートツールを使う
  - 主な用語: トピック指向
- 一箇所にまとめる
  - 手間暇かけずに済ませたい人向け
  - Richild。物タスクを一箇所に集める
  - Monolith。タスクを一箇所に集める、特に単一のテキストファイルに全部書く
  - 主な用語: 物タスク
- 繰り返しの力で回す
  - 自分のリズムを尊重したい人向け
  - Robot。厳密なタスクリストをつくってそのとおりに行動する
  - Tracker。習慣トラッカーにタスクを載せて習慣としてこなす
  - 主な用語: ルーチンタスク、習慣タスク
- 自分の脳内が正義
  - そもそも管理やツールにとらわれたくない人向け
  - Inboxer。動線上でインボックス（未処理を溜める場所）を使うだけで済ませる
  - 主な用語: アクティブレイジー

=== Chapter-8 タスク管理のスタンス ===

タスク管理のスタンスとは、タスクをどう処理するかという対応パターンを指します。メンタルモデルと言っても良いかもしれません。前章の戦略は「どう扱うか」という管理の話でしたが、本章、スタンスは「どうこなすか」という処理の話です。

抱える量を減らす

自身が抱えるタスクの分量を減らそうとするスタンスです。

- デリゲーター
  - Deligator、自分で処理せず人に任せます
- スプリンター
  - Sprinter、タスク管理をはさまずにその場ですぐに処理します
- スキッパー
  - Skipper、今日やると決めたこと以外はとりあえず先送りにします

減らし方に違いが見られます。デリゲーター人に任せることで減らし、スキッパーはとりあえず先送りすることで今日の分を減らし、スプリンターはタスク管理対象として抱える前に処理してしまうことで管理対象を減らします。

デリゲーター

Deligator、委譲者。

概要

タスクをなるべく委譲しようとします。

委譲とは人に任せることです。委譲先として有名なものはいくつかあります。重役は右腕として秘書を抱えていますし、エンタメ界隈やスポーツ界隈のプレイヤーにはマネージャーがつきますし、共同生活している人がパートナーや両親に仕事や家事を任せていることもよくあれば、寮生活という形で管理人にまかせているケースもあります――と、極めて身近な概念です。

委譲のようで委譲ではないもの

指導、マイクロマネジメント、雇用は委譲ではありません。

まず委譲とは「任せること」なので、デリゲーター自身は基本的に任せた先からの結果を待つだけです。細かく状況を確認したり、指摘や指導を入れたりするのは委譲ではなく、ただの管理（それもマイクロマネジメントと呼べるような細かいもの）や指導です。**委譲とは管理も含めて任せることです。**

また家事代行や執事など、経済的に雇うものも委譲ではありません。これは単にお金の力で解決しているだけであり、いうなれば雇用です（※1）。委譲とは経済力に頼らず、工夫や立場によって誰かに任せることです。とはいえ工夫は難しくて、共感、人心掌握、交渉や説得、場合によって洗脳など高度な諸々が要りますので、通常は立場次第です。**委譲できるかどうか**

は、委譲する・される立場（あるいは役割）があるかどうか で決まります。なければつくることもできますが、つくった立場に従わせるのにも結局工夫か立場が必要です。

- ※
  - 1 委譲先として機能する人材を雇用することはできますが、話がややこしくなるので割愛します。たとえ雇用関係があったとしても、委譲であるなら管理も含めて任せるほどの信頼または器量が発揮されるはずです。そういう意味では、委譲には相手への信頼または自身の器量を必要とします。

スプリンター

Sprinter、短距離走者。

概要

発生したタスクをその場ですぐに処理しようとします。

認知資源の節約

スプリンターの主目的は認知資源をセーブすることです。

GTDでは 2 分ルールなる原則があります。2 分で終わることはタスク管理せずその場ですぐに終わらせろ、というものです。「ボールはすぐに返せ」「持ったままにするな」もよく聞く金言です。また、少しずれますが、フロントローディングという「最初しばらくは全力で取り組め」とするテクニックもあります。

これらに共通するのは、自分で抱えることの負担を減らすことです。抱えているものがあると（どれを選ぶか・いつ構うかといった事を考えてしまうため）判断が生じますが、判断には認知資源を消費します。認知資源は有限ですし、一度消耗したら睡眠するまで回復しません。休憩、昼寝、瞑想などで粘ることはできますが、たかが知れています。判断が鈍ったりできなくなってしまっては元も子もないので、認知資源は時としてお金や時間よりも貴重であり、消費を抑えることは非常に重要です。毎日同じ服を着る経営者の話も、この認知資源を節約したいからです。一般的に言えば、判断の量を減らせば節約できます。抱えているものを減らすことは、まさにその一つです。スプリンターは、タスクを抱えずにすぐ処理してしまうことで認知資源をセーブします。

完璧じゃなくても前進する

スプリンターのもう一つの目的は、とにかく前進することです。

現実はままならないもので、事前に想定した計画はよく裏切られますし、自分が思っている以上にいいかげんに進んだり振り回されたりもします。特に現代は VUCA とも呼ばれ、ただでさえ

先が読めない時代です。

そんな状況下における経験則として、よく挙がるのは「雑でもいいからとにかく進める」ことです。手元で考えたり調べたりして納得するだけじゃなくて、実際に行動に移します。行動に移すことの大切さは昔から耳にタコができるほど金言化されていますが、スプリンターはそれをまさに地で行きます。発生したタスクをなるべくすぐに処理する、という形で行うのです。すぐ処理することで状況がわずかでも進行し、進行すれば状況が変化して新しい何かが生まれます。

副次的なメリットとして、状況が変わったらそれに反応すればいいだけで済むようになる率が高くなることも挙げられます。自分で管理せずとも、見えたことに反応していればいいのでラクチンなのです。すでに述べたとおり、人間は怠け者ですが、怠け者でも比較的取り組みやすいあり方がイベントドリブン――何かイベントが起きたときに受動的にそれに対応することです。スプリンターとしてすぐに処理してしまうことは、いわばイベントを発生させることにも等しい営みです。

スキッパー

Skipper、先送り魔。

概要

今日やると決めたタスク以外は基本的に先送りしようとします。

先送りしても別に死なない

スキッパーの根幹は「本当に急いでやらなきゃいけないことなんてほとんどない」です。実際そうであるかは状況次第ですし、特にリアルタイム性の高い仕事ではそうも言ってられませんが、たいていは単に自分のこだわりの問題にすぎません。たとえば単に自分が気になるからとか、嫌われたくないからとかいったものです。そういったこだわりを捨てて、「とりあえず明日やりますわ」と先送りにするのです。

ただし、前提として今日何をするか（デイリータスク）を決めておく必要があります。毎日デイリータスクを設計するからこそ日々進展が生まれるのですし、デイリータスクに集中するためにデイリータスク以外を先送りすると発想できます。デイリータスク無しの先送りはただの放棄にすぎません。デイリータスクという基準があるからこそ、それとそれ以外に区別することができます。

スキッパーの思想はマニャーナの法則が詳しいと思います。

裁量の獲得が肝

先送りしたところでタスク自体が消えてなくなるわけではありません。曖昧なタスクや状況が慌ただしい場合に消えたりすることはありますが、それに期待するのは無鉄砲です。それでもなぜ先

送りするかというと、デイリータスクという自分のペースを死守したいからです。別の言い方をすると、**先送りとは自分なりのペースを尊重するために（明日以降どこかでたぶんやらないとけないけど）今日はしない、と決めること** です。

そのためには「今日はしない」と決められるだけの裁量が求められます。自分ひとりで進められる作業や資料作成などは比較的裁量が高いですよね。本当の期限までに間に合わせればいいだけです（もちろん先送りしすぎると締切前の追い込みなどという学生症候群に陥ります）し、これをうまく管理するのがまさに個人タスク管理の領分です。逆に、ひとりではコントロールできない場合、たとえば仕事をする≒打ち合わせに参加する、のような拘束の多い仕事だと厳しいでしょう。皆が参加する打ち合わせがあるのに「私はその仕事は今日はしないと決めました」と主張して不参加に倒すのは許されないか、勝ち取れても関係に亀裂が入るでしょう。

学生症候群の回避

学生症候群とは締切直前になってから本腰を入れる現象を指します。名前は比喩的なものであり、ビジネスにおけるプロジェクトでもしばしば見られる現象とされています。

スキッパーはこの学生症候群に打ち勝たねばなりません。そのためにはタスクの本当の締切を見据えた上で、毎日計画的に少しずつ消化していくことになります。あるいは不確定要素が多くて先を読みづらい場合は（スプリンターの項でも少し述べましたが）フロントローディングにてまずは視界を広げることもあります。大げさに言えば、先手を打つことに命を賭けるタイプです。

スキッパーは一見すると楽そうに見えますが、決してそんなことはなく、むしろ「それができれば苦労はしねえよ」と言いたくなるような計画的で行動的、そして野心的なスタンスなのです。

省力化する

タスクの処理にかかる時間や手間を減らそうとするスタンスです。

- オートマトン
  - Automaton、仕組み化（統一化とできれば自動化）します
- リジェクター
  - Rejecter、断る対象やパターンを決めておいて、それらに当てはまるものは断ります
- フォーカシスト
  - Focusist、直近取り組む対象を決めて、それら以外は無視します

オートマトンは処理そのものを減らし、リジェクターとフォーカシストは「やらないタスクを決める」判断の手間を減らします。

オートマトン

Automaton、仕組み魔。

概要

タスクの処理を可能な限り統一化し、できれば自動化もします。

頭を使いたくない

前章の戦略でいう Inboxer、本章でいうスプリンターがまさにそうですが、自身の頭の性能、スキルや経験などで瞬発的にバリバリこなしていきたいとする人達がそれなりにいます。この人達は瞬発から外れるテンポの悪さで頭を使うことを嫌います。たとえば紙やテキストに言語化して書き出して、それを見ながらひねり出す、といった営みです。

なぜかというと、このような営みは認知資源を酷使するからです。また、酷使というほどではありませんが、やり方やタイミングが決まっていない場合も同様に疲弊しがちです。要は頭を使いたくないのです――というと語弊を招きますが、慣れてない使い方で使いたくないのです。たとえるなら短距離選手が持久的な運動を嫌うようなものです。

さて、頭を使わないためにはどうすればいいでしょうか。自分が想定するやり方で周囲をコントロールする（というより想定外の他のやり方を許さない）ことがまず挙げられますが、これは立場や権力を持つ人にしかできません。そこでオートマトンです。

オートマトンとは自分の側で工夫を重ねることによって、タスクを統一的なやり方で扱えるようにします。また、可能なら自動化もしてしまって、そもそも自分で手を動かさなくても済むようにします。

統一化と自動化

統一化とはプロセスやフォーマットを絞ることです。自動化とは手作業を廃することです。

わかりやすいのは、戦略でいう Calendarer でしょう。Calendarer は予定もタスクも全部カレンダーにつっこんで、私はカレンダーにひたすら従うマシーンになりますとする戦略ですが、これはカレンダーという形で統一しています。その気になれば自動化もある程度できます。たとえば「Slack でブックマークしたメッセージを、カレンダーの 0:00 に追加する」との連携をつくった場合、Slack 上でタスクと判断したメッセージをブクマしたら、あとはカレンダーを開いて 0:00 に追加されてるものを適当に修正するだけです。カレンダーへの追加作業を手作業で行う必要はありません。

プロセスやフォーマットというと仰々しく聞こえますが、単に関係者にこのやり方でやりたいと合意を取ることも含みます。打ち合わせでいえば、定例会議はまさにこれです。毎回会議の開催を調整するのは大変ですから、だったら最初から定期的に確保してしまおう、たまたま参加できなかった場合は仕方ないですね追加のフォローはしません議事録を見てください、などと決めてし

まうわけです。

ここまで他者との協調の前提で書きましたが、もちろん自分個人に閉じた話でも通用します。特に日頃から情報収集や振り返りといったナレッジワーク的な営みをしている人は、それらのために使う情報源を固定化したり、情報を集める下準備を自動化したりできるわけです。その結果、手作業だとあれこれ迷いながら 20 分かかっていたものが、統一的なやり方で 2 分で終わるようになったなどもよくあります。他にも安易に SNS やニュースサイトを浴びるのではなく、信頼できる人が集めた情報（キュレーターが集めた情報）だけを見るといったことも統一化の一例です。Menthas などキュレーターによる情報を集める部分を自動化したものもありますし、この分野はキュレーションサイトと呼ばれます。

統一化と自動化は、要は **仕組み化する** と言えるでしょう。

仕組み化には頭を使うという皮肉

才能がある方は瞬発的な頭の使い方で仕組み化ができてしまいますが、珍しいと思います（だからこそ立場や権力や財力を得てパワーで無理やり統一させがちです）。

皮肉なことに、仕組み化を考えたり試したりするのに、まさに瞬発的ではない頭の使い方が求められがちです。頭を使いたくないからオートマトンでありたいのに、オートマトンであるためには頭を使わないといけないのですね。これを **オートマトンのパラドックス** とでも呼びましょう。

このパラドックスを打ち破るには何が必要でしょうか。有名な格言としてプログラマーの三大美徳があります。これは「怠慢」「短気」「傲慢」の三つから成ります。プログラマーはまさに仕組み化を生業とした生き物ですが、そうあるためには短気で傲慢な怠け者とのマインドセットが必要と説いています（いると思います）。そんな心持ちがあるからこそ、強い信念のもとに「怠けるための仕組み」を全力でつくりますし、つくるための鍛錬にも余念がありません。

オートマトンのパラドックスに打ち勝つためには、それくらいの強さが必要なのです。ちなみに、アナログな手段だと自動化できる余地も少ないためデジタルに頼らざるを得ないのですが、そのデジタルがまさに IT、プログラム、ソフトウェアであり、結局 IT スキルやプログラミングスキルと呼ばれるものと対峙することになります。

リジェクター

Rejecter、拒否家。

概要

拒否したいタスクをあらかじめ決めておき、そのようなタスクが来たら問答無用で拒絶します。

Minimize Negatives を地で行く

TODOリストの逆の概念として「やらないこと（Not TODO）リスト」なるものがあります。また、仕事術にせよタスク管理にせよ、優先順位付けの話や取捨選択、特に捨てることの重要性もよく説かれます。

一般的に言えば、ポジティブなことを増やす（Maximize Positives）アプローチと、ネガティブなことを減らす（Minimize Negatives）アプローチがあって、前者を求めがちですが、実は後者もあります。ネガティブを減らすことによって時間や精神を節約するのです。もっとも、何もかもを拒否していてはただの自堕落になってしまいますが、現代人は通常ネガティブを抱えすぎ、もっと言えば捨てるのが下手なので、通常は Minimize Negatives はやりすぎるくらいがちょうど良かったりします。

さて、リジェクターですが、このスタンスでは具体的に条件と行動を定義します。A が起きたら B をする、という形で定義するのです。これを **拒否ルール** と呼ばせてください。単にやらないことリストにテキトーに書くだけでは定着しないので、じぶんルールとしてちゃんと拒否ルールを書くのです。

拒否ルールの例

拒否ルールの例は無数にありますが、いくつか取り上げます。

『見覚えのない人からのインターホンが来たら、居留守する』 性善説の方や優しい方は相手をしたくなるかもしれませんが、見覚えのない人は高確率でセールスや宗教勧誘ですし、稀に（一人暮らしの女性を狙う不審者など）悪意を持っていることもありますので、時間や安全を守るためにも問答無用で拒絶する、と決めます。それでもやりづらいなら「重要な要件なら後日尋ねてくるだろう」「電話など別の連絡手段を入れてくるだろうし、正当な相手なら私の電話番号を知っているはずだろう」など根拠を集めます。あるいは、最初に一言「営業ですか？」を聞くようにして、肯定するか渋る場合は問答無用で拒否する、とのルールを追加してもいいでしょう。ちなみに、ドアにセールスお断り、勧誘お断りの張り紙を設置することもできますが、こちらは仕組み化でありオートマトンの範疇になります。

『SNS で不快な物言いをする人がいたら、ミュートやブロックをする』 例として X を挙げますが、X には指定アカウントやキーワードを非表示にするミュートや、自分との関与自体を断ち切るブロック機能があります。対面の人間関係ですと、多少嫌なことがあってもある程度耐えねばならない（相手に気づかれないように拒絶したり離れたりするのが難しい）場面も多いですが、SNS は違います。リアルタイムで背中を見せるわけではないので、対面ほどやりづらくはないです。それでも（ブロックの場合は）相手からアクセスしてきた場合に気付かれる、など全く気付かれないとは限らないため慢心は禁物ですが、少なくとも対面よりはやりやすいため、積極的に使っても良いのです。特にミュートについてはキーワードやハッシュタグに対しても行えるため、人ではなく話題を対象にするという意味でやりやすいと思います。

『17時以降に通知が来ても、無視する』 夕方以降を穏やかに過ごすために、たとえば 17 時以降は通知を一切見ないようにするのもアリでしょう。といっても、単に心がけるだけでできたら

苦労はしないので、何らかの仕掛けが必要です。毎日 16:50 に「すべての通知をオフにする」予定を、7:30 に「すべての通知をオンにする」予定を入れれば忘れずオンオフができます。一部のコミュニケーションツールには夜間の通知のみオフにする設定もあったりしますので、それを使ってもいいでしょう。また、スマホについて言えばタイムロッキングコンテナ なる商品もあったりします――と、工夫は色々ありますが、このような拒否ルールは「対象が来たら行動する」ができないので（そもそも対象が一時的にやってこないようにするための）仕組み化が事実上必要になります。つまりはオートマトンですね。

拒否ルールのマトリックス

例を3つほど取り上げましたが、一般的に言えば、以下のようにマトリックスで四分類できます。

	反応的	予防的
スルーする	1 反応的スルー	2 予防的スルー
ブロックする	3 反応的ブロック	4 予防的ブロック

まず対処としてスルーする（行動を起こさず無視する）か、ブロックする（行動を起こして相手に働きかける）かがあります。上記の例はいずれもスルーです。張り紙やミュートなどは働きかけに聞こえますが、直接相手に、というより仕組みを整えて間接的に備えているだけですのでスルーの範疇です。ブロックというのは、たとえばインターホンに出て「営業はお断りなのでお帰りください」と返したり、X で当人に「～～の話題は不愉快なのでやめてもらえませんか」と返信したりすることです。相手に直接働きかけています。

セオリーとしては、まずスルーを試します。これで済めば理想です。どうしてもダメならブロックに出ます。直接働きかけるだけあって疲弊しますが、これで済めばめでたしです。それでもダメなら自分から離れます。私はこれを Through → Tackle → Transfer ということで **リジェクターの 3T** と呼んでいます。なお、本当は何もせず様子を見たり我慢したりする Through と、こちらが折れて妥協して何らかの行動を取る Temporize の二つがあり、厳密には Through → Temporize → Tackle → Transfer の 4T です。ここでいうスルーするとは、Temporize のことです。つまり、

- Through（様子見・我慢）
- Temporize（スルー）
- Tackle（ブロック）
- Transfer（自分から距離を置く・逃げる）

となります。リジェクターは Temporize と Tackle を工夫することでタスクを拒否していくスタンスです。

次に、このスルーやブロックのやり方も二つあって、対象が来る度に対応する「反応」と、そもそも対象が来なくなるように干渉するか、あるいは来ても自動で処理されるように仕組み化する（こちらはオートマトンも使える）「予防」があります。一般的に予防の方がコストがかかりますが、

対象が何度も来るようなら予防にしちゃった方が中長期的に見れば楽です。

自動化やドキュメント作成でも同じことが言えますが、どのタイミングで予防に踏み切るかはトレードオフの問題です。一方で、X のミュート機能などは、知っていればすぐにでも使えます。そういう意味で、予防は知識とスキルに左右されます。日頃から調べたり鍛えたりしておけば、予防もしやすくなります。特に普段使っているツールの設定はくまなく見ておくのがおすすめです。案外リジェクターとして使えそうなヒントがごろごろ眠っています。

フォーカシスト

Focusist、焦点者。

概要

直近注力するタスクのみに焦点を当て、それ以外のすべてを無視したり捨てたりします。

許可リスト的

リジェクターは拒否ルールを用意してそれらに従ってなるべく拒否するスタンスでしたが、フォーカシストは逆で、許可対象を用意してそれらだけに従う（それら以外を拒否する）スタンスです。前者はブラックリスト、後者はホワイトリストと呼ばれます。近年では差別的表現を避ける潮流もあり拒否リスト、許可リストと呼ぶこともあります。

以下に対応をまとめます。

- 許可リスト
  - 呼び方は Allow List、Safe List、White List など
  - フォーカシストはこちらの発想です
- 拒否リスト
  - 呼び方は Deny List、Block List、Black List など
  - リジェクターはこちらの発想です

フォーカシストのマトリックス

フォーカシストの運用は 4 種類に分類できます。

	短期的	中長期的
タスクが対象	1 短期的タスク型	2 中長期的タスク型
キャパシティが対象	3 短期的キャパシティ型	4 中長期的キャパシティ型

まず何にフォーカスするかとしてタスクとキャパシティがあります。タスクについては、アイビー・リー・

メソッドが好例ですが、直近はこれらのタスクだけやります（それ以外はしません）、と決めてしまうことです。キャパシティについては時間ややる気や個数といった限界を定めて、それに収まるようにすることです。たとえばタイムボックスは、仕事の枠を 8 時から 11 時、13 時から 17 時と決めてしまって、その枠以外では一切仕事をしないようにします。少しわかりづらいですが、以下に対応を記します。

- タスク型では、「自分」に割り当てる「タスク」を制限する
  - 例: 自分に割り当てるタスクをA,B,C,Dの4個に制限する
- キャパシティ型では、「行為」に割り当てる「資源（リソース）」を制限する
  - 例: 仕事に割り当てる時間を8〜11時と13時〜17時に制限する

次にフォーカスする期間も分かれまして、短期的と中長期的があります。短期的とは一日以内のスパンでフォーカスするものであり、今日はこの 6 個のタスクだけやりますとか、今日の午前中は 8〜11 時まで仕事します（仕事という行為に 8〜11 時の時間を割り当てます）といったものです。明日や今日の午後どうするかはわかりません。一方、中長期的とは日単位以上に伸ばすもので、今週は毎日この 6 個のタスクしか相手にしませんといった考え方をします。

まとめると、タスク/キャパシティと、短期的/中長期的で計 4 つのタイプに分かれます。

どのタイプを使うかは状況次第です。どうしても終わらせたいタスクがあるならタスク型が良いでしょう。普段の処理能力や生産性に問題はないが体調やガス欠が心配であるならキャパシティ型が良いでしょう。つまりタスク型はブーストするためのフォーカスで、キャパシティ型はセーブするためのフォーカスです。期間については、まずは短期的に試すのが良いです。今日だけやってみるとか、午後だけやってみる等ですね。それで要領を掴めてきたら、中長期的に続けてみるようにします。とはいえ、現実は常にフォーカスし続けられるほど単純ではないので、短期的なフォーカスができそうなら差し込む、くらいがバランスが良いでしょう。

きれいにフォーカスできるのか

たとえばタスク型で A, B, C の 3 つのタスクにのみフォーカスすると決めた場合、厳密にこの 3 つのみに従い、他のタスクは厳しく無視する・捨てるべきでしょうか。

答えはイエスですが、実際はそう単純ではありません。もっと言うと「答えとしてはイエスだが、実際できるかはわからない」となります。

まず私たちには日常生活があります。いくら A, B, C の 3 つだけと決めても、食事睡眠排泄を含む一切をやらなくていいかというと、そうではありません。大部分をおざなりにすることもできなくはない（ですしそうするクリエイターやサラリーマンも珍しくない）ですが、長くは保ちませんし、寿命を削っているようなものです。そもそもそうして消耗すると、必ず集中力が切れて A, B, C 以外にも脱線します。一方で、生活に慣れている人・最適化している人・体力のある人は、身体的精神的負担ほぼ無しにこなせるでしょう。この人達は日常生活をタスクとは捉えていません。

次に方向性や視界の問題もあります。当初は A, B, C の 3 つが正しいと思ったからそうフォーカスしたが、実際やってみると実は D の方が重要だったとわかったりします。このときは当然 A, B, C に固執するというよりも、D を加えた方が良いですよね。

ここまでをまとめると、きれいにフォーカスする際の壁は 3 つあります。

- 1: 人間として体力的・精神的な限界があること（また時間やお金などの資源上の限界もあります）
- 2: 慣れているタスクの一部はささっとこなせるが、そうであるとは限らないこと。たとえば日常生活においてやることは意外とある
- 3: 方向性が違うとわかったり、視界が開けてきたりして、当初のフォーカス対象の正しさが変わること

もちろんフォーカシストとしてはなるべくフォーカスを続けたいし、（フォーカスのやり方が最適だと信じている立場なので）続けるべきなのですが、これら壁があるせいで実際はブレやすいのです。ブレを抑えるためには、壁を取っ払っていく――のは難しいので、**壁の存在も踏まえてフォーカスする** ことになります。

1 の限界については、自分の限界を知った上でキャパシティ型のフォーカスを使います。2 の慣れについては、慣れているタスクはカウントしないでいいですし、逆に慣れてないタスクであれば計上するべきと考えます。たとえば、あなたが引っ越した直後で、色々と慣れてない生活上の手続きや整備を行う必要があるとしたら、A, B, C だけやるんだとは考えずに、A, B, 引っ越し後の対応をやるんだ、にしておいた方が良いでしょう（C は諦めて引越し後の対応にフォーカスする）。最後の 3 については、対象として使うタスクや行為の粒度を粗くすることです。たとえば右記のような具体的なタスク――「英語の参考書の第一章を終わらせる」「ゴールデンレトリバーの子犬を飼うための調査」「チームで X さんと Y さんの仲が悪いのを何とかする」を掲げるよりも、「英語の勉強」「犬飼いたいので調べる」「案件を成功させるために」くらいで捉えた方が何かと融通が利きます。

盤外戦をする

タスクそのものの処理の仕方を考えるよりも、タスクの発生や抱え方の方に目を向けて干渉するスタンスです。

- スラキスト
  - Slackist、余裕の確保と死守に専念します
- コンテキストン
  - Contexton、タスクの文脈を知ることに力を注ぎます
- ミキサー
  - Mixer、タスクをまとめて一気に処理しようとします

余裕、文脈、一括処理などアプローチは大きく異なります。

スラキスト

Slackist、余裕人。

概要

余裕の確保と死守が第一であり、タスクをやるかどうか・どこまでやるかといった判断は余裕と相談して決めます。

バッファとスラック

余裕にはバッファとスラックがあります。バッファとはイレギュラーに備えるためにタスクに対して意図的に確保する時間的余裕であり、スラックは時間的・精神的・経済的その他余裕全般を恒常的に確保すること、あるいはしたものを指します。

バッファの確保

バッファについては、たとえば一週間で終われそうなタスクがあった場合、普通は一週間後に締切や評価機会を設定すると思いますが、そうではなく二週間やそれ以上を設定します。

タスクが自分ひとりで完結する場合、自分が決めればいいだけなので問題にはなりません。強いて言えば、そのようなゆとり重視の生き方を自分が許すかどうかです。

問題となるのは他者が絡む場合です。協調にせよ提出にせよ、他者が納得する見せ方をしなければなりません。バッファを設けることをバッファを積むと言いますが、通常、バッファを積むと理由を問われます。あるいははっきりと告げなくとも「なんか長くない？」「もっと短くできないの？」と勘付かれます。このような追及を交わす必要があります。

工夫は無数にありますが、上記を例にしていくつか挙げます。

- 一週間時点で（本当は終わっているが）まだ 50% ですと中間報告する
- 努力の跡を目に入りやすくしてサボっていない旨を周囲に知らせる
    - 例: 筆者は IT 企業のサラリーマンですが、仕事中の検討や成果は基本的に（機密範囲に基づいた範囲で）フルオープンにしています。また日頃から情報共有や議論や雑談もよくやります。周囲にはアウトプットが凄いとかメモ魔とかいった評価をされており、毎日毎日よく頑張ってるなぁと見えるわけです
- 思うように進行しない事情や状況を上手く説明する
    - 仕事の状況を論理的かつ定量的に説明して説得する路線と、育児や介護や病気など個人の事情を持ち出す路線があります

- まずは一週間ください、など少し試してから次を相談するスタイルに持ち込む
    - 単純なタスクでもなければ、やってみて初めてわかることも多いので、最初からそのつもりで望みます
    - 期間は一週間でも二週間でも構いませんが、長すぎるとダレるし相手も気になって仕方がない、短すぎると報告の場が多くてうっとうしい、でバランスが重要です
    - 一週間ごとに報告＆次何をするかを決める、とサイクルにできればなお良しです（バッファを積むというより、一週間の期間に積むタスクの量をコントロールしやすいという話になりますが）

ここで重要となってくるのが、**そもそもバッファを積むことを許される環境かどうか** です。許されない場合、そもそもスラキストのスタンスは通用しません。仕事の性質により許されない場合と、組織や権力者の物わかりが悪くて許されない場合があります。筆者の持論ですが、バッファを積めるかどうかは、職場がホワイトであるかどうかの目安の一つだと思います。

スラックの確保

スラックを確保するためには、地道で継続的な取り組みが必須です。時間、精神、お金の三点を見ていきます。

時間

時間がかかるタスクをいかに抱えないかにかかっています。

仕事の文脈ではキャラクターの演出に帰着されるでしょう。たとえば毎日定時で退社する気難しい人や、育児、介護、その他マイノリティの告白などで「任せづらい」理由を見せている人などは、タスクの量は少なくて済みます。このようなキャラクターを演出できるよう、日頃から取り組む（場合によっては衝突する）わけです。もっともこれは解雇されづらい日本だからこそ通用しやすい話です。また日本であっても、器用に追い詰められたり、下手をするといじめられたりもします。一方的に主張すれば良いというものではなく、デメリットを補えるだけのメリットを示したいところです。

実は筆者もこのタイプで、会社が想定するチームプレイや実作業の要員としてはポンコツです。かわりに、管理されずとも自律的に報連相ができます（タスク管理も武器の一つです）し、他の人が避けがちな新規事業検討や新技術調査ができます、その上（出来はアバウトですが）行動やアウトプットも早いです。メリデメ両方があるわけです。加えて、業界や世の中の機運としても、変化していくことは必然で、組織としても悩んでいます。その結果、筆者には「後者寄りの新しい仕事を任せよう」「たまに進捗や結果を確認させよう」となるわけです。たとえば週に一度や数日に一度の報告用会議体を設定して、残りは自由に過ごせます。通常の社員が行っている慌ただしいチームプレイはほぼありませんし、朝会のような毎日レベルの管理もありません。それだと上手くいかないことや、そうしなくても上手くやれることがわかっているからです――と、これはあくまで一例ですが、自分と環境の双方に配慮した最適化を探っていくのが鍵となります。もちろん探っても一向に成功しないことがあり、筆者も苦労しましたしぶっちゃけ成否は運

にも左右されます。通じない案件が来てしまう度に筆者は苦しんでいます。耐えられないほど慢性的に通じないなら、社内転職や社外転職など環境自体を変えるしかありません。

理想は優れたスキルを持っていて仕事を選べる立場になることですが、そうかんたんじゃないのでここでは想定していません。そうはいってもスキル、特に周囲と差別化できて役に立ちそうなものがあると便利ですので、日頃から鍛えておくことは重要です。筆者の例でいうと、アウトプットの早さやタスク管理はそうでしょう。本書を書いているのも、実はタスク管理という専門性で、より強い立場を掴みたいからです。それだけの力やポテンシャルがあることを示したいからです。

精神

メンタルロード（Mental Load）がかかるタスクをいかに避けるかにかかっています。

メンタルロードとは造語です。直訳すると精神的な荷物・負荷であり、頭の中で高頻度にちらつく不安や不満を指します。物価高いとか将来の年金どうしようといった持続的なものもあれば、A さんが私のメッセージにだけ返事返してないんだけどなぜだろうとか、さっき駅構内でぶつかってきたおじさんマジでうぜえわとかなど突発的に発生するものもあります。通常は突発的で、放っておいてもそのうち忘れますが、中にはしつこくこびりつくものもあります。イヤーワームをご存知の方は、同じようなものだと考えてください。

メンタルロードを抑えるためのポイントは 3 つで、内省と検査と廃棄です。

内省とは遠慮のない自問自答を指し、自分の本当の思考や性質を知るために自分に問い、それに答えることを繰り返します。頭だけでは難しいので、言葉にして書きながら行うのが良いです。嫌なこととも向き合うので非常に疲れますし、敏感な人など不調をきたして続行できない人もいます。その場限りの内省で気付けることはなく、通常は継続的に日記やノートの形で取りながら少しずつ気付いていく営みになります。

検査とは何らかの特性を検査することです。昨今で言えば ASD、ADHD など発達障害、IQ の微妙な塩梅を示す境界知能、LGBTQ などセクシャルマイノリティと、特性を示した概念や体系はそれなりに存在します。特性とは先天的または後天的に刻み込まれた「仕様」であり、変えるのはほぼ不可能ですから、いわば自身の性質として知っておくと便利です。特性上、これはできません、できないから頑張るまでもありません、という形で不向きなことを最初から回避（あるいは観念）できます。検査方法ですが、医療的な手段が用意されている場合はさっさと頼ります。性的指向など手段が確立されていない特性の場合は、能動的な情報収集やコミュニティへの参加などで悟りにいく必要があります。いずれにせよ、日常生活の狭い生活だけでは自覚しづらく、他者からの指摘も期待できないので、そのような概念や手段や場所があるのだという点を知れるかどうかが重要になります。もっと言えば、ありきたりですが日頃からいろんな情報に広く触れて、出会える機会を増やすことです。

そして最後に廃棄とは、内省や検査によって把握した「自分の特徴」に基づいて、メンタルロードの原因に対処を加えていくことです。対処とありますが、実際は **捨てる営み** が求められます。何かをやめたり、環境から離れたりするのはわかりやすいですが、通院したり薬を手に入れ

たりする場合でも「腰が上がらない原因となっているプライドや食わず嫌いを捨てる」ことが必要だったりします。あるいはひとりでできないとわかっているなら、仲間なりパートナーなりが必要で、それを手に入れるための行動が必要で、ここでも人付き合いや金稼ぎに対する何らかの信念を捨てた方が上手くいくでしょう。また人間関係自体のメンタルロードが大きかったり、子供を持つことによるそれが明らかに許容外だとわかっているなら、人間関係や子供の養育を諦めた方が良いです（メンタルロード的には健全です）。

このように廃棄は、単に捨てることだと口で言うは易しですが、非常に難しいのです。子供の養育を諦める、と決断できる人がどれだけいるでしょうか。異性と親しくなる可能性を潰すために意図的に容姿や言動を醜くして遠ざける、との行動ができる人がどれだけいるでしょうか。スラキストとして精神的な余裕を恒常的に確保するためには、このくらいはできないといけません。廃棄という過激な選択は避けては通れません。このような過激な選択を行うために**選択に納得できるだけの自己理解――深い内省と検査が必要なのです**。つまり内省と検査も避けては通れません。別に自己理解ができるなら方法は問わないのですが、内省と検査を経ないで至るのは筆者は不可能だと思います。あるいは至れたとしても、芸術など抽象的・非言語的に表現して発散するのが関の山ではないでしょうか。

と、強い主張を述べたことをお許しください。これはあくまでスラキストとして、恒常的に余裕を確保したい場合の話です。そもそもそこまでして余裕が欲しいか、には議論の余地がありますが、通常はそうではありません。子供の養育についても、幸せと苦労がセットになっていますが、それでも人は選ぶのです。人生、山あり谷ありと言いますし、人間もリズムで駆動していれば、世界は物理的にも波に支配されています。人間は波を好む特性なのかもしれませんね。そういう意味で、スラキストとは波に抗うスタンスとも言えます。「いや、別に山なんてなくてもいいから平坦であってほしいよ。特に谷を減らしたいんだ」というわけです。

お金

金銭的な余裕を確保するには物量と割り切りと生活水準のすべてでバランスを取ることが望ましいです。

物量があった方がいいのは言うまでもありません。貯金が 0 円なのと 1 億円なのとでは、後者の方が安心感が違います。貯金があると、いざというときにお金でササッと対処できるので、この手の心配事がなくなります。しかし、資産が多すぎると、それはそれで管理に意識が向いてしまって本末転倒です。保管や運用も必要ですので、信頼できる人に任せられなければ気の休まる暇がないといいます。またお金を稼ぐために手間暇かけすぎるのも（スラキストとしては）本末転倒でしょう。年収 400 万円で毎日 8 時間の暇があるのと、年収 4000 万で毎日 8 時間睡眠すら厳しいほど忙しくて休日も 2 週に 1 日しか取れないのとでは、前者の方が良いでしょう。あくまでスラキストとしては、ですが。

と、このようにお金はあればあるほど良いとは限らないので、どこかで割り切る必要もあります。年収 400 万でいいのか、1000 万くらいか、4000 万か、1 億か、それともそれ以上か――自分の価値観と能力との相談でしょう。一方で、すべてを自分で決めるのは苦しいので、資本

主義的価値観に浸ってとりあえず上を目指すのが楽ではあります。宗教と同じです。ただ、何度も言うように、スラキストとしては余裕を重視するので、（恵まれていて労無くお金を増やせるのでなければ）あまりガツガツはしません。

もう一つ重要なのが生活水準でしょう。生活水準はいわば恒常的にかかるコストですから、水準が高ければ高いほどお金も必要で、そのお金のための活動も必要になり、余裕から遠のきます。かといって水準が低すぎると、それはそれで不便になってしまい、本人の優秀さや鈍感さがもろに要求されます。極端な話、ホームレスや生活保護やヒモで快適に過ごせる人もいますが、万人には真似できないでしょう。生活水準の厄介なところは、上げるのはかんたんなのに下げるのが難しいところです。

時間の項で挙げた子供もそうですが、一度手を出すと撤回しづらいものがあります。子供せよ、生活水準にせよ、やっぱやーめたと軽率にやめられないわけです。これを **不可逆要素** と呼ばせてください。スラキストは、スラックの確保という観点では、不可逆要素との戦いと言っても過言ではありません。

Minimize Negative 再び

そこまで余裕を求める理由は何でしょうか。

つまるところ、何にも縛られない自由な時間を愛しているからだと思います。その自由を何に使うかはその人次第です。ゲームや読書かもしれませんし、スポーツかもしれませんし、散歩やDIYかもしれません。家族やペットとのんびりするだけでもいいですし、別に寝転がってスマホやテレビをダラダラ見るのでもいいでしょう――と、使い方を見ても人次第としか言えないので本質には迫れません。

どちらかといえば「嫌なこと」や「嫌なことが起こり得ること」を回避したい、という消極的な動機が強いです。前述しましたが Minimize Negative ですね。FIRE（Financial Independence, Retire Early）がもてはやされているのも、労働という「ネガティブの根源」を丸ごとなくせるからです。自己実現とか成長とか野望とかそういったものよりも、ネガティブを根絶できることの方が重要なのです。精神の項でスラキストは波に抗うスタンスと書きましたが、筆者はこれを **フラティズム（Flatism）** と呼んでいます。平坦で静かな人生でいいよ、というわけです。

余裕も持て余しすぎると病む

スラキストも人間であり、人間は持て余しすぎると病みます。

書籍『暇と退屈の倫理学』では、人間は元々遊動生活民であり狩猟や移動を伴う慌ただしい生活をおくる生き物だったが、農耕により定住生活民への移行に成功。しかし回路は遊動的であり、定住では使い切れないため今度は持て余すようになり、暇と退屈が問題となった――のようなことが書かれています。筆者もそう思います。人間のメカニズム自体が遊動的で、だからこそ持て余してしまい、これに対処するため様々な宗教や文化が生まれたのでしょうし、この流れは現在も変わりません。

現代の変化の早さと情報の密度は言うまでもなく、むしろ過剰ゆえに、いかにセーブするかというカウンターも色々提唱されていますよね。スラキストもその一つです。だからといって余裕を確保できたら、今度は遊動的ゆえの持て余し問題に回帰します。スラキストはこの問題とも向き合わねばなりません。

筆者もまだ解は知りませんが、その人その人にあったやり方以上に、人間の仕様に則った活動もある程度は行うことが必要ではないかと考えています。たとえば人は孤独感が強いと生理的に警戒態勢になりますが、その間、心肺に負荷がかかります。慢性的な孤独感が強いと、慢性的に負荷がかかっているようなものであり、健康上よろしくありません。孤独は1日15本の喫煙にも等しいとのキャッチーな言い回しもあります――と、少なくとも孤独感の解消は何とかしないと危なそうですよね。他にも考慮すべき事項は多数あります。

同書では退屈の第二形式を勧めています。これは何らかの場に参加して一応気を紛らわせてはいるが、それなりに暇であり、色々思うところがあって悶々としている状態です。これが（程々に潰せている上に思考もできていてバランスが良いので）一番マシだと説いています。ちなみにその他には第一形式と第三形式があり、第一形式は状況に急かれていて思い通りにならない分が「待ち」になるというものです。第三形式は「なんとなく退屈だ」的な空虚であり、仕事中毒や自分探しに陥りやすい（そして無事対象が見つかると第一になる）とされています。

他には、Twitter でバズったネタですが、イギリスにはオールドバチェラーと称されるあり方があるそうで、元ツイートを引用しますと、

> 独身おじさん友達いないの件。英国ではそういう人はオールドバチェラーと称され、“With a dog and a few good books” をなかば合言葉としてカントリーサイドで楽しく暮らす場合が多いようです。ワンコとお気に入りの本数冊こそ最高の友という発想です。

スラキスト的にスラックを担保しながらも、犬という生物を飼うことで孤独感を和らげる、触れ合うことでオキシトシンを分泌する、散歩により健全な生活リズムになるといったカバーができていて、なかなかバランスが良さそうに聞こえます。

と、雑多な紹介になってしまいましたが、スラキストは余裕が手に入ったら終わり、ではないのです。**むしろ手に入ってからが本番です**。この営みができない人はスラキストには向いていませんし、向いてない人はおそらくまた慌ただしい仕事なり何なりに飛び込んでいきます。成功者が常に忙しくしていたり、無駄に散財したり目立ちたがったりするのも、結局暇が嫌だからではないでしょうか。

タスク管理というよりは人生哲学

スラキストについて解説してきましたが、タスク管理というよりは、もっと上位の目線に立っていることがおわかりいただけるかと思います。このように、タスクそのものをどう処理・管理するかではなく、その上位の目線で、そもそもタスクが発生しないように動くことを盤外戦と呼びました。

コンテキストン

Contexton、文脈家。

概要

タスクに安易に飛びつくことを良しとせず、タスクのコンテキスト（文脈）を収集・理解することを重視します。

解釈する者

コンテキスト（Context）とは、和訳すると文脈です。前提、背景、周辺情報といった言い方をしても良いでしょう。

通常は仕事にせよタスク管理にせよ、発生したタスクは基本的に「やることが確定したもの」であり、（優先順位は考えねばなりませんが）疑うことなく処理すべきものです。コンテキストンはそれを良しとしません。タスクには文脈があるはずであり、それを知り、尊重することが真の価値につながっていくと考えます。文脈がわかれば納得感も出ますし、優先順位の判断――特に今やらなくてもいいとか、どうせこれ以上考えてもわからないからさっさとやってみようとかいった判断もしやすくなります。

ここまでだとコンテキストンは文脈を探る、さしずめ探偵のようなイメージを思い浮かますが、違います。本題はその後で、把握した文脈を踏まえた上で何らかの解釈（そして行動）を行ないます。**行動するところまでがセット** であり、コンテキストンの本懐でもあります。文脈を知って楽しみたいだけの観察者ではなく、タスクと向き合っていきたいのです。そのスタンスが「文脈をちゃんと踏まえてから動こう」「文脈は一瞬ではわからないことが多いからちゃんと見よう」というだけの話です。

ちなみにコンテキストンは新規事業との相性も良いです。新規事業や業務改善など創造的な営みには（参考はあれど）正解が無く、初手を打てる程度に文脈をさっさと集めて、あたりをつけて、さっさと行動することが求められます。それでまた色々情報が得られるので、文脈として追加で解釈して、また行動して――を繰り返します。仮説検証とも呼ばれますが、仮説とは暫定的な文脈とその解釈に他なりません。調べるべき文脈がそもそも「無い」ので、暫定的に定めて、試してみることで何とか生み出そうとするわけです。特にそのサイクルを高速で回さないとらちがあかないためスピードを重視します。一見するとコンテキストンには見えませんが、文脈を重視するからこそ、その導出に力を注ぐわけです。

日本との相性は悪いのでゲーマーになる

日本では文化的にハイコンテキストなコミュニケーションを行ない、包括的思考といって周囲の誰が何を言っているかをよく見る傾向もあります。皆がそう言っているのだからそうでしょ、という

わけですし、その「そう言っている」も明示的に示されているとは限りません（ハイコンテキスト）。

コンテキストンはこのようなあり方とは相性が悪いです。文脈をはっきり探りにいく点はハイコンテキストに逆行しますし、理解した文脈に基づいた判断や行動も、周囲によって形成された暗黙の了解とはズレがちです。むしろ問題児と扱われるでしょう。

そういうわけで、現実的には露骨な行動はせず、場を乱さないようにこっそりと把握しながら、こっそりと行動に反映していくことになります。微妙な制約を踏まえながら動く様はまるでゲームであり、コンテキストンとはゲーマーと言えるかもしれません。

たとえば根回しという形で水面下で攻めていったり、飲み会に参加して飲みニケーションの形で本音を聞き出したり（文脈を集める）します。これらは日本文化におけるセオリーであり、従えるかどうかが勝敗の分かれ目となります。ゲームは勝ち筋をちゃんとなぞれるかどうかがすべてですが、同じことなのです。文脈を知れたら勝ったようなものだ、あとはどうとでもなる、ではないのです。むしろ文化的制約に従うことの方がはるかに重要です。

ノートテイキングの重要性

タスクの文脈を把握した結果、「今すぐはやらない」と倒すことも多いです。しかし、その場で判断しただけですと、次はいつ思い出せるかわかりません。あるいはそのタスクを思い出せたとしても、当時の文脈や判断までは思い出せないかもしれません。脳内だけでは限界があります。この限界を越えられないと、タスクが溜まって形骸化するか、あるいは溜めるのは性に合わないと判断して「今を生きる」系の生き方、特に目先のタスクを消化しまくれば済む作業的な生き方になりがちです。

越えるにはどうすればいいでしょうか。ノートテイキング、つまりはノートを取ることです。

戦略の章では話題で区切るとして Issuiest や Topician を紹介しましたが、タスクとその文脈を律儀にノートに書いておくのです。かつ、このような営みを続けます。そうすることで何十、何百、下手すれば何千ものタスクとその文脈、そして自分がどう判断したかといった情報が貯まっていき、第二の脳になります。忘れても見れば思い出せます。どうせ全部に取り組むのは無理ですが、脳内だけに頼って何もできないのと、ノートに頼ってまがいなりにも処理していけるのとでは全然違います――と、まるで人生のすべてをノート化するかのような言い方をしましたが、もちろん限定的な適用もできます。毎日日記を書いたり、たまに読み返したりするだけでも違いますし、仕事の案件 A についてのみノートを取るだけでも（A については）違うでしょう。

現代は何かと慌ただしいですし、多様性も進んでいて「自分の生活や考え方も尊重しやすい」時代です。その分、考えることも増えますし、一方で人間のキャパシティなどたかが知れていますから、一度にできることは限られていて取捨選択が要ります。文脈を把握したところで、今すぐ生かせるとは限らないのです。だからこそ外部に保存して、第二の脳として蓄えていく必要があるのです。そうしないと前述したとおり形骸化するか、今を生きる他はなくなってしまいます。

わかりやすいのは医者でしょうか。患者ごとに記録を取ってますし、経過も取りますよね。言わ

ば患者ごとの文脈をノートに取っています。そうしないととてもじゃないですが覚えていられません。コンテキストンは、同じことをタスク全般に対して行うようなものと言えます。

コンテキストンのようでコンテキストンでない者

よくある勘違いが **エセコンテキストン** です。

じっくりと対話や交流に時間をかけるタイプの人がいますが、この人達がコンテキストンであるとは限りません。特に関係者の感情を把握したり、少しずつ変えに行ったりしようとする行為はただのひとたらしにすぎません。この人達は非言語情報を駆使します。その性質上、対面と口頭によるコミュニケーションを重視します。この力を極端に悪用したものの一例がカルトや詐欺です。

感情も文脈の一つではないか、といわれると、一応イエスなのですが、それだけではありません。文脈とは情報であり、言語的で論理的なものです。情報と言語をもって文脈はこうだよね、と明らかにして、文脈がこうだから行動としてはこうするのが良さそうだよね、と繋げていくのです。ここを避けて、単に感情の制御に帰着させるのは、コンテキストンの風上にも置けない、まがい物にすぎません。

私たちは人間であり、人間は感情的な生き物で騙されやすいので、エセコンテキストンに騙されてしまわないよう注意が必要です。「この人はちゃんと文脈を把握しようとしてくれている」とか「文脈を知りたい私にちゃんと応じてくれる、ありがたい人だ」と信じる前に、本当に文脈を見ている・出してくれているのか、エセではないかをしっかりと見定めたいところです（※1）。

- ※
  - 1 一方で、ホストなどまさにエセコンテキストンを生業とする職業も多数存在します。個人で適度に浸るのは自由ですが、タスク管理の文脈では悪手、というより搾取でしかないので避けたいところです。こう書くと、自分には縁がないと思われるかもしれませんが、感情的に制御しようとするエセコンテキストンは意外と多くいます。まともに付き合うと時間の無駄なので、程々にしましょう。一方で、そうすることでしか信頼関係を育んでくれないパターンもあり、難しいところです。

ミキサー

Mixer、一括家。

概要

複数のタスクをまとめて一気に処理しようとします。

束ねるか仕込むか

ミキサーは主に二通りあります。

一つは関連するタスクを束ねた上で一気にこなすことです。目的はスイッチングコストの最小化であり、要は頭の切り替えを最小限にしたいのです。行動としては、たとえば午前中はクリエイティブな仕事をして、午後は雑務とコミュニケーションをして、夕方以降は人と会う用事とその準備をして――という風にタスクをグルーピングして時間帯に当てはめます。行き過ぎると家事や運動、読書やネットサーフィンといったこともそうします。家事は空いた時間に少しずつ行う人も多いと思いますが、ミキサーはそうではなく、昼休憩でご飯食べた後に残り時間いっぱいまでやる（リモートワークの場合）とか、日曜日の午前中にやるなどと決めます。

もう一つは「得られるものが大きい機会」を積極的に探して、そのための準備に手間暇をかけることです。普段地方に勤務している人が２ヶ月ぶりに東京本社に行く場合、都会でしかこなせないようなタスクも洗い出して、当日いつ何するかも設計します。時間が足りないようなら自費覚悟で滞在を延長するとか、休暇を入れる（それよりも土日にくっつけるのがてっとり早いですが）とかいったことも厭いません。イメージはまさに旅行です。得られるものが大きいので、準備もしっかりします。他には「イベント」もそうでしょうか。資料つくって共有して済ませるのではなく、わざわざイベントなる場を企画して、宣伝もして、本番に向けた準備や根回しも余念なく行っていきます。

したがってミキサーの思想は以下の二つです。

- 頭の切り替え（コンテキストスイッチング）はできるだけ抑える
- 然るべき場で一気に処理することこそが最善

また、ミキサーの過ごし方には「本番」が登場しがちです。ミキサーは一括でこなすということで一括家と書きましたが、本番を設けてそこに全力投球するという意味では本番家とも言えるでしょう。

ミキサーは割と常に束ねる・仕込む

上述した営みはミキサーに限らず、割と誰もが行うことではありますが、ミキサーはこれを割と常に行ないます。スタンスというくらいですので、黙っていても自然とそうします。一見するとすぐには動かないので、外から見るとノロマに思われがちです。

だからといってミキサーが何もしてないかというとそうではなく、旅行における旅行計画や各種手配と同様、行動はそれなりにしています。というより、トータルで見ると、安直にタスクをこなす人達よりも多くの行動をしています。オートマトンの項にて、オートマトンのパラドックスに打ち勝つにはそれなりの強さが要ることを示しましたが、ミキサーも同様で、強い信念――上述した思想に関する強いこだわりを持っている事が多いです。

この点も旅行がわかりやすいです。旅行が趣味の人は息するように準備を余念なく行ないますし、旅行先でうまくやるための鍛錬（たとえば言語の勉強）も惜しまなければ、日頃から次はどこに行こうか何食べようかとアンテナを張っていて感度も高いです。本人は下手すれば自覚して

いないくらいに当たり前に行動していますが、他の人から見ると相当頑張っているなぁと見えるレベルの行動をしています。ミキサーも同様です。

最高の体験がしたい

ミキサーの根源は体験にあると筆者は考えます。

何かを手に入れたいというよりも最高の体験がしたいから、その準備や演出を重視します。旅行やイベントはわかりやすいですが、個人的な仕事においても同様で、たとえば「午前は冴えた頭でクリエイティブな仕事がしたい」「愛用のデスク環境に囲まれて、コーヒーを嗜みながらつくる」「その前に早起きして、朝日も浴びて身体のスイッチを入れる」といったこともミキサーです。午前中にクリエイティブな仕事を一気にこなすためにあれこれ頑張っているわけですが、単に生産性を最大化できればいいというわけではなく、過ごし方も重視しています。

まとめ

- スタンスとは
  - タスクをどうこなすか
  - 戦略はどう扱うかという管理の話だが、スタンスは処理の話
- ===
- 抱える量を減らす
  - 例: 人に任せるデリゲーター、その場ですぐ処理するスプリンター、先送りをするスキッパー
  - 人に任せる立場、すぐ処理できる行動力と性能、先送りに確信を持てる計画性とメンタル、など能力的な適性が求められます
- 省力化する
  - 例: 仕組み化するオートマトン、拒否ルールに従うリジェクター、注力対象以外を無視するフォーカスト
  - 処理そのものを省力化するか、処理対象の選別（特にやらないものの判断）を省力化します
  - 「省力化を手にするためには、一時的に省力化とは無縁の努力をする必要がある」とのパラドックスこそ手強いですが、やり方や適用対象には比較的融通が利きます
- 盤外戦をする
  - 例: 余裕を死守するスラキスト、文脈を知りに行くコンテキストン、一気にこなすミキサー
  - タスク管理というよりも人生哲学の話に寄っており、一貫した行動を継続的に行える信念が求められます

=== Chapter-9 □第3部「タスクのようでタスクでないもの」===

タスクのようでタスクでないものはいくつか存在します。わかりやすい例は「予定」や「メモ」です。これらはタスクとして扱っても上手く行かず、扱い方には工夫が必要です。

このようなものを本書ではオルタスクと呼んでいます。ここからは主なオルタスクとその扱い方を詳しく見ていきます。

=== Chapter-10 コンテナ ===

コンテナの概要

概要

コンテナ（Container）とはタスクを分類したもの、あるいは分類の各項目を指します。

書類をクリアファイルで分類したり、本を本棚や仕切りで分類したり、パソコンではファイルをフォルダで分類したりしますが、タスクも同様に色々な分類に分けることができます。物理的に物を分けるというよりは、論理的に意味を分けるニュアンスが強いです。

また、分類というと厳密な分け方をイメージしがちですが、そうとも限りません。むしろタスク管理は各個人各組織それぞれにあった分類を自分で設計するものなので、雑な分け方もありがちです。

コンテナの例

タスク管理に登場するコンテナには色々な名前がつきます。ツールやメソッド、あるいは人によって使い方が変わります。

よく使われるのは以下のような名前です:

- フォルダ
- カテゴリー
- グループ

- タグ
- ラベル
- ワークスペース
- プロジェクト

中には小難しい単語もあり、身構えてしまうかもしれませんが、所詮は何かを（少なくともタスクを）分類しているだけです。各用語の細かい意味については後述します。

リッチコンテナ

コンテナにはタスクのみを分類したものと、タスクも分類するしついでに他の情報や機能も入れたものとがあります。後者を**リッチコンテナ（Rich Container）**と呼ばせてください。前者を明示的に指すときは、純粋なコンテナということで**プレインコンテナ（Plain Container）**と呼びます。

上記のよく使われるコンテナ名を筆者の肌感覚で一般的に分けてみると、こんな感じになるでしょう。

- プレインコンテナ
  - フォルダ
  - カテゴリー
  - タグ
- リッチコンテナ
  - グループ
  - ラベル
  - ワークスペース
  - プロジェクト

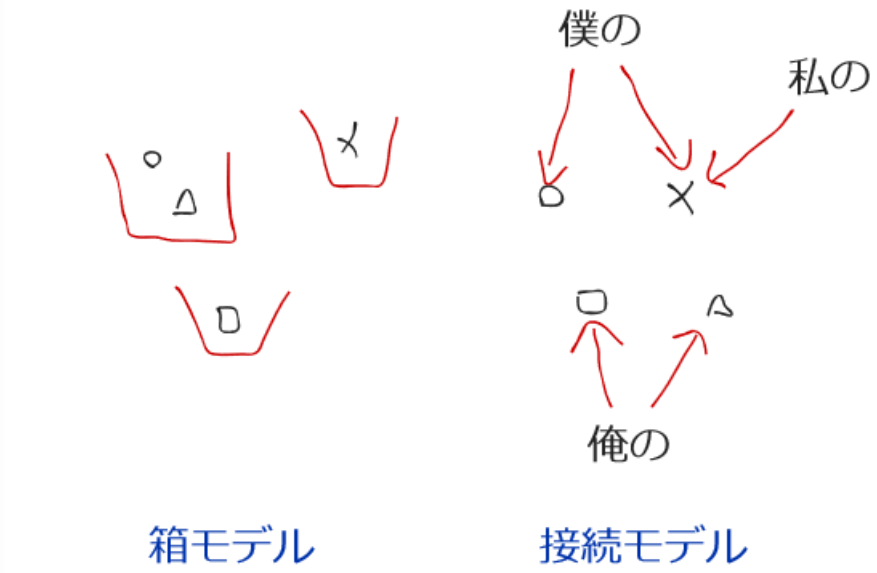
フォルダやカテゴリーはプレインコンテナとしてわかりやすいと思います。

一方で、ワークスペースやグループといった大きめの単位は、単にタスクを分類するだけでなく、メンバーの情報だったりアップロードしたファイルの情報なども持っていたりしますよね。これらはリッチコンテナです。ラベルは色つきのタグであり、色という情報も持っているので一応リッチの扱いです。

箱モデルと接続モデル

コンテナとして「何かを入れる箱」を思い浮かべがちですが、それだけではありません。

何かを入れるような世界観（箱モデル）の他に、何かと何かを繋ぐ世界観（接続モデル）もあります。フォルダやカテゴリーは前者、タグやラベルは後者になることが多いです。



箱モデルは、ある対象を同時に一つの箱にだけ入れられます。必ずどこかの箱に一つだけ入っている（どこにも入れてないと見つかりませんが）ことが保証されるのでシンプルではありますが、こだわりだすと一貫性を保ちにくくなります。「どっちにも該当するけどどっちに入れたらいいんだろう」のような問題もよく起きます（※1）。

接続モデルは、ある対象に目印をつけるイメージです。上図では誰のものかという目印をつけています。つけるだけなので一つの分類項目（この場合は「誰」）が複数の分類対象につけることもできますし、ある分類項目と別の分類項目（この場合は「僕」と「私」）が同じ対象につけることもできます。箱モデルよりも柔軟性は高いですが、つけすぎると形骸化します。

- ※
  - 1 こうもり問題と呼ばれます。こうもりは哺乳類だから獣にも分類できるし、翼もあるから鳥にも分類できるし、とどちらに分類すればいいか迷いますよね。こうもり問題をなくすのは一般的にかなり難しいです。図書館で見かける日本十進分類法のようなものもありますが、あのような分類をつくるには専門家が長い時間をかける必要があります。また、そうして出来上がった分類はそもそも複雑であり、分類作業自体にも苦労するでしょう。個人のタスク管理において、このレベルを確立できるかというと、おそらくできないのではないかと思います。筆者もできませんでしたし、正直人類が挑むものではないとさえ思います。もちろん、これは分類にこだわりすぎた場合の話です。雑に分類する分には困りませんし、それだけでも何も分類しないよりは便宜が得られます——一方で、かえって混乱して害悪だという場合もあるので、一筋縄には行きません。

コンテナはどう扱うか

タスクとして扱わない

まずコンテナはタスクとして扱わないべきです。

仮にタスクとして扱ったとしても十中八九うまくいきません。コンテナは分類であり、分類とは粒度の粗い事柄ですので具体的に何をすればいいかがわからなかったり、あれもこれもやらないといけなくて決断ができなかったりします。一方で、タスクは粒度が細かいものであり、何をすればいいかが明確です。明確だからこそ管理もできます。

粒度

粒度 という言葉が出ましたが、事柄の細かさだと考えてください。よく「具体的じゃないタスクは分解して具体的にしろ」「大タスクは中タスクや小タスクに分解しろ」といいますが、これは粒度の粗いものを細かくしろと言っています。なぜかというと、繰り返しますが粒度が粗いと理解や行動がしづらいからです。

別の言い方をすると、**コンテナとは粒度の粗いタスク** だとも言えます。ただし、粗いがゆえにタスクとして扱うのは難しいから分類として扱うのです。

コンテナの扱い方は 4 つ

ではコンテナをどう扱えばいいかというと、以下に分かれます。

- 1: ツールのコンテナ機能を使うし、ツールの通りに使う
- 2: ツールのコンテナ機能を使うが、独自に使う
- 3: ツールのコンテナ機能を使わないが、何らかのコンテナは使う
- 4: 何も使わない

上に行くほど制約がきつくなりますが、普通は 2: が多いです。ツールとしても「コンテナをつくる機能はあるけど、どんなコンテナを使うかは皆さんに任せます」となっていることが多いです。状況は人それぞれだからですね。たいていのツールにはカテゴリー、フォルダ、タグといった機能がありますが、実際どんなカテゴリーやフォルダやタグをつくるかは皆さん次第です。元からつくられていることもありますが参考程度であり、そのまま使うことはあまりなく（とっかかりとしてとりあえず従うのはアリ）、それらは消した上で自分なりに作り直すのが定石です。

最も制約のキツイ 1: は、筆者は見たことがありませんが、例を挙げるとGTDです。ツールの話からは離れてしまう（GTDはメソッドです）のですが、GTD ではインボックス、プロジェクト、いつかやる、連絡待ち、ネクストアクションといった箱がいくつか登場し、いつどれをどのように使うかも定まっています。私たち利用者はその辺をちゃんと勉強して、そのとおりに従う必要があるわけです。勝手に一部の箱を使わなかったり、新しい箱を追加したりしてもいけません。設計されたとおりに従わなかった場合は GTD 本来の効果が得られないからです（※1）。

すでに良さそうなコンテナのあり方と運用を知っている場合は、おそらく3: になるでしょう。つまりツールの機能を使わず、独自に管理するのです。たとえばカレンダーを重用している人は、他のツールを使っていてもそのコンテナを使わず、わざわざカレンダーの方に予定として追加すると思います。この場合もカレンダーというコンテナ（タスクを日や週で分類している）は使っています。

と、このように何らかのコンテナは使うはずですが、中にはそもそも分類なんてしたくない人もいます。その場合は4: です――とはいえ、そのような人でも実は雑な分類は使っていたりします。戦略の章では一箇所にまとめるを紹介しましたが、これはコンテナをただ一つだけつくってそこに全部ぶっこんでいるとも言えます。そういう意味では3: の一種です。

- ※
  - 1 余談ですが、ソフトウェア開発では「設定より規約」という言葉があります。開発で使われる道具は通常、汎用的につくられていて、どう使うかは開発者が決めるのが主流でした。設定項目が豊富なわけですね。一方で、そうじゃなくて、道具の開発者が「設定の余地はなくしたぞ。この道具が言うとおりに使え。ちゃんと使い方を学べ。そしたら便宜は保証してやる」とある種傲慢なスタンスで道具をつくるケースも出てきました。設定の余地はなく、ただただ俺のやり方に従えというわけですね。扱い方の1: は、いわば設定より規約です。

それはタスクか、コンテナか

そもそもタスクとコンテナの区別はどうやるのでしょうか。

結論を言うと人次第であり、粒度をどう捉えるかです。

筆者なりに一つの捉え方を書きます。タスクとは具体的で、実行可能で、始めれば数日以内で終了できて、終わったかどうかがはっきりとわかるものです。一方、それ以上の粒度になるとコンテナ――複数のタスクを束ねる単位になります。たとえば「引っ越し」はタスクというよりもコンテナになるはずです。タスクというのは「全体スケジュールを組む」「引越会社をリストアップする」「引越会社各々に見積もり依頼をする」「すべての見積結果を比較してどの会社を選ぶか決める」などです。人によってはもうちょっと細かいかもしれません。筆者ですと「会社Aに見積もり依頼」「会社Bに見積もり依頼」レベルで分けますね。

以下に箇条書きで階層的に書いてみます。コンテナには□、タスクには□、どちらか怪しいものには□をつけます。

- □引っ越し
  - □スケジュール組む
    - □試す
  - □引越会社の選定
    - □リストアップ
    - □見積もり依頼 A社

        - □見積もり依頼 B社
        - …
        - □選ぶ
    - □荷造り
        - □クローゼット系
            - □試す
        - …
    - □廃棄
        - …
    - □新しい家を探す
        - …

これはあくまで筆者の場合です。筆者は頭の中で考えるのが苦手なので、細かく洗い出してタスク管理しがちです。スケジュールとクローゼット系には□がついてますが、これは「一見するとすぐ出来そうだけど」「たぶん洗い出すとタスクが色々出てくるんだろうなー」「たぶんコンテナになると思う」のようなことを考えています。まだ確定ではないのでいったん□をつけています。こういうときは考えても仕方がないので、とりあえず試してみるのが良いです。それを「試す」というタスクとして書いています。

また、タスクを属性や性質で分類したいときも、同様にコンテナになります。たとえば自宅のデスクで調べ物をする系のタスクに「調べ物」タグをつけたい場合、この「調べ物」はタグにするべきであって、タスクにするべきではありません。「調べ物」というタスクを管理したところで、あらゆる調べ物タスクを適切に管理・遂行できるかというと、できないと思います。一方、タグにしておけば、（日頃から適切に調べ物系タスクにこのタグをつけているのなら）調べ物タグから各種タスクを辿れるようになります。

分類の話

コンテナとはタスクの分類ですが、そもそも分類とは何でしょうか？

分類の一般論を、タスク管理も適宜絡めながら見ていきましょう。

分類のメリット

分類のメリットはコストの軽減 です。具体的には探すコストと切り替えるコストの二つです。

探すコストについては、図書館などがわかりやすいですが、種類ごとにエリアを設けて同じ種類のものを集めておくと探しやすくなります。Aに関するものは分類Aの中に入っている、を保証したいわけですね。分類というと厳密で几帳面なものをイメージしがちですが、雑でも構いません。それこそ「タスクは全部ここに入れます」という箱を一つつくって、全部ぶちこんでしまうのも立

派な分類です。

切り替えるコストについては、頭の切り替え――コンテキストスイッチングの話で、同じものをまとめておけば「同じ頭の使い方で扱える」ため楽できます。逆に違うものが混ざっていると、頭の切り替えに苦労します。極論ですが上司として部下を厳しく指導する立場と、子煩悩な親として子供を甘やかす立場は違うわけで、リモートワークの会議中に子供を甘やかしながら部下を厳しく指導するのは難しいでしょう。通常は両方とも独立した時間を確保するでしょうし、間に休憩もはさむと思います。フィクションでは切り替えを一瞬で行う「集中の化け物」が描かれることもありますが、現実的ではありません。

分類の対象と分け方は様々

上記メリットに対して「何を当たり前のことを……」と思われたかもしれませんが、意外とタスク管理では盲点になりがちです。というより分類を応用できる余地は意外と多いのです。

すでに「分類は雑でいいこと」と「頭の使い方を分けること」を挙げています。分類というとタスクを「仕事」「家庭」「趣味」と分けていくことだよねと考えがちですが、それ以外の分け方が色々とあります。何を分類するのか（分類対象）とどこに分類するのか（分類先または分類項目）は様々なのです。

自分で分類をつくる

分類とは「誰か偉い人がつくったものを使う」イメージを持つかもしれませんが、一方で自分でつくることもあります。日常生活では基本的に誰もが何らかの分類を少しはつくっているはずです。

タスク管理も同様で、後者のように自分でつくることが多いです。一方で、「こんな分類をすれば上手くいくよ」という事例やパターンもあり、適宜参考にすると捗ります。それでも自分に合った分類を自分でつくるのが基本だという点はぜひ押さえてください。

逆を言うと、自分で分類をつくっていく気がない人や従うつもりがない人は、雑に向き合う程度で良いでしょう。世の中は分類、特に階層の思想がまだまだ強く、各種ツールやメソッドにも何かとこの概念が現れがちですが、別に無理して従う必要はないです。（タスク管理からは離れますが）この潮流に抗ったのがScrapboxで、階層的な整理をせずに書いても破綻しないWikiとなっています。その本質はリンクであり、ページとページをリンクにより繋ぐことによって、あとから辿ってたどり着けるようにします。人間の脳と同じネットワーク構造なだけであって、これが意外と成立するのです。筆者も23000ページ超えのScrapboxを持っていますが、破綻することなく続いています。この世界観には感動しまして、Scrapboxの解説本を書いたほどです。

と、少々脱線してしまいましたが、分類が絶対的ではないこと、自分なりにやればいいこと、何ならやらなくてもいいことはぜひ押さえてほしいです。分類作業が目的になってしまっているパター

ン――分類に適応できない自分を責めたり、維持するために苦労して作業を続けたりといったことがあまりにあるあるだからです。タスク管理という言葉も堅苦しくて助長しがちですが、惑わされてはいけません。

コンテナの例とニュアンス

コンテナの例としてプロジェクトやフォルダなどいくつか挙げましたが、本項ではそれぞれの一般的なニュアンスをまとめます。

フォルダ

フォルダ（Folder） は、主にファイルやノートを分類する概念として使われている言葉ですが、タスク管理としても登場することがあります。

たとえば Wrike です。以下引用します。

> タスクをフォルダーに追加することで関連情報を1箇所にまとめます。また、スペースの組織構造を構築し、情報の検索や共有を容易にします。フォルダーはタスクやプロジェクトとは異なり、実行可能な項目ではありません。また、フォルダー自体に一連の属性はありません。チームや部門全体をカバーするスペースよりも、より小規模で詳細なレベルで使用できます。

フォルダーという字面からは箱モデルを思い浮かべますが、むしろ接続モデルだとわかります。「引っ越し先の家を探す」タスクがあるとして、これを「引っ越し」フォルダーと「待ち時間中にできること」フォルダの双方に入れることもできるわけです。本質的には「引っ越し」タグや「待ち時間中にできること」タグをつけているのと変わりませんが、あえてフォルダーという言い方にしているのは階層性を持たせたいからです。実際にサブフォルダーとの用語もありますし、結局タグと呼ばれますとの説明もあります。

そういうわけで、フォルダーという言葉には階層的に分類するニュアンスがある、と捉えておくと良いでしょう。

カテゴリー

カテゴリー（Category） は、コンテナとしてよく登場する言葉の一つです。直訳しても「分類上の区分」「ジャンル」「範疇」などとなります。

ニュアンスとしては箱モデルです。カテゴリー A に入れたタスクは、カテゴリー B に入れることはできません（あるいは入れるなら A からは取り出すことになる）。1-カテゴリー 1-タスクを守って、きっ

ちり分類するための機能と言えます。フォルダーと同様、階層性を持つこともありますが、持たないこともあります。

グループ

グループ（Group） は、人を束ねる概念とのニュアンスが強いです。タスク管理においてもタスクグループといった言い方で使われることがありますが、あまり例がありません。Yahoo!カレンダーくらいでしょうか。

意味的にはカテゴリーと同義だと捉えれば良いでしょう。つまり接続モデルというよりは箱モデルです。

タグ

タグ（Tag） もコンテナとしてよく登場します。直訳は「札」であり、タスクが性質 A を示す場合に A というタグをつけておく、というニュアンスです。

モデルとしては接続モデルであり、1つのタスクに複数のタグをつけられます。タグもたくさんつくることができるので、雑に運用していると何十何百というタグがすぐに出来てしまいます。これをちゃんと運用するのは不可能に等しく、形骸化します（あるいは良くて「よく使う少数の選ばれしタグ」が運用されている程度）。

X のハッシュタグなど、デジタルツールとしては非常に一般的な概念でもあります。こちらには性質の表明とのニュアンスもあります。ポストに #タスク管理 と書いてあれば、そのポストはタスク管理に関するものですよ、と表明しているに等しいですし、検索もできます。

そういう意味で、分類というよりは性質の表明と捉えるのがわかりやすいでしょう。

ラベル

ラベル（Label） は、タグと同義ですが、色の指定もできることが多いです。

たとえばGitHub Issuesではタスク（を指す単位である Issue）にラベルをつけられますが、このラベルには名前と色があります。重要性の高いラベルを赤色にしたり、廃止や中断を示すラベルを灰色にしたりできるので視認性が上がります。一方で、乱用するとちかちかして見づらくなります。

プロジェクト

プロジェクト（Project） はタスク管理においてトップを争うほどの多義語です。

ざっくりと言えば、すぐには終了しない、ある程度きりのよくまとまった一連のタスクの集まり、とでも言えるでしょう。生活でたとえるなら「引っ越し」はまさにプロジェクトという言葉で捉えられるものだと思います。一方で、ビジネスでは非常に粒度が大きいことがあります。たとえば数年以上継続する大規模な事業もプロジェクトと呼ぶことがあります。

メソッドや体系によってはさらに制約が加えられることもあります。体系として PMBOK では一時的であること（永遠には続かないこと）や独自の成果を生み出すことなどが指定されていますので、延々と続けているルーチン的な作業はプロジェクトとは言えません。メソッドとしては GTD もこの言葉を使いますが、PMBOK とは定義がずいぶんと違います。

と、状況によってだいぶ意味が違いますので、タスク管理として捉えるのであれば、単に「活動」あるいは「事業」の一言で問題ありません。ただツール（とその背景にあるメソッドや体系）次第では何らかの含意が込められていることがあるので、できれば知っておくと良いかな、というくらいです。たとえばツールとしては規模の小さなプロジェクトのみ想定されているから、規模の大きなプロジェクトを管理しようとしてもなんだか上手くいかない、といったことが起こります。前者の想定がわかれば、後者の管理は諦めがつきます。

ワークスペース

ワークスペース（**Workspace**）は作業場、作業空間といった意味になり、典型的なリッチコンテナです。つまり単にタスクをまとめる単位というだけでなく、タスク以外の情報も色々とまとめます。というより、まさに「場所」を示すニュアンスがあり、仕事中は高頻度に皆が集まってくる場所になりがちです。

タスクについても、ワークスペースに直接ぶら下げるというよりは、さらに別のコンテナが使われます。たとえば、

- ワークスペース
  - プロジェクト
    - タスク
    - タスク
    - ……

のような形です。

様々な分類対象と分類項目

先ほど分類対象と分類項目は様々だと書きましたが、コンテナとしてよく使われるものを紹介します。

大中小タスク

大きなタスクを小さなタスクに分解するシチュエーションはよくあります。すでに引っ越しの例は述べました。

どこまで分解するかは自由ですが、あまり深すぎても全体像を把握しづらくなるため、3段か、多くても4段がいいでしょう。大・中・小の三段階はわかりやすいと思います。書籍でも章節項の形で目次を構成したりします。PARAメソッドではノートツールが 4 段階までサポートしていることに注目しています。4 Based と呼んでおり、人間のワーキングメモリ含め色んな研究結果を踏まえているそうです。

3 段にせよ 4 段にせよ、このような階層的分類において重要なのは、各階層のニュアンスを固定することです。たとえば小タスクは一日以内に終われる具体的なもの、中タスクは小タスクの集まりで特定の目標を表現したもの、大タスクは中タスクの集まりで特定の方向性を表現したもの、とすれば、どのタスクを大中小どこに入れるかもわかりやすいですよね。方向性は大タスク、目標は中タスク、そして具体的な個々のタスクは小タスクで捉えて、中タスクを満たすにはどんな小タスクをこなしていけばいいのかとか、大タスクを満たすにはどんな中タスクを集めていけばいいのか――といった感じで当てはめていけるようになります。

タスクと情報の仕分け

特にメールがわかりやすいですが、受信箱に届いたメールを色んなフォルダに分類すると思います。このとき、特にメールは「タスク」と「情報」に分かれます。大半は情報ですが、たまにタスクも混ざっています。タスクを表すメールついては「タスク」フォルダをつくってそこに入れたり（箱モデル的）、あるいはフォルダには入れないが、タグやマークをつけたりします（接続モデル的）。

と、このようにタスクと情報を区別するシチュエーションもよくあります。割合としては情報の方が明らかに多く、逆にタスクの方は少ないので、後者のタスクの方をコンテナとして特別扱いすると良いでしょう。箱モデルで分けるのか、接続モデルで付けるのかは人次第です。人によって箱モデルが適していたり、逆に接続モデルで雑に付けた方が上手くいったりします。ここは本当に人次第ですので、自分がどちらに合うかを探りたいところです。

いずれにせよ、重要なのは **タスクだとわかった時点ですぐにコンテナに入れる（つける）** ことです。あとでまとめて行うのは苦しくて、たいてい形骸化しますので、日頃からすぐにコンテナに入れる癖をつけておいた方が良いです。こう言うと「通知や受信が来る度にすぐ分類するのか」と思われがちですが、そうではありません（慌ただしいとそれしかできなくなりますが）。どちらかと言えば、チェックする時間を設けてそこで一気に行ないます。メールでいうと、たとえば一日朝、昼、晩と三回くらいチェックするタイミングを設けて、そこで一気に仕分ける（タスクをコンテナに入れる）のです。

と、このように、タスクと情報の仕分けはそれなりに奥が深いです。特に情報は多いですから、い

かに省力化してタスクという本質だけをさっさとコンテナに入れられるかがポイントになります。メーラーの話になりますが、HEY など、まさにこの仕分けの部分を明示的に強化したツールもあったりします。

コンテキストタグ

筆者は コンテキストタグ（Context Tag） と呼んでいますが、タスクにつけておく「いつ実行できるのか」「どんな性質なのか」を示す用のタグを使うと便利です。

たとえば通勤中、電車内やバス内でこなせるタスクには「通勤中」タグを付けておきます。すると、車内にいるときにスマホで通勤中タグの一覧を見ることでタスクをこなせるわけです。あるいはもっと単純な例は「あとで読む」タグです。より生々しい例で言うと「夫がいないとき」タグや「上司がいるとき」タグも使い所が結構あります。特に忙しい人は対人関係が多い人は、この手の制約が色々あるはずですから、コンテキストタグとして上手く反映してやると、スキマ時間などでガンガンタスクを消化しやすくなります。

セクション

戦略の章で少し述べましたが、時間帯のことを セクション と呼びます。人はどの時間帯に何をするかが大体決まっているので、セクションをコンテナと捉えて、タスクをどのセクションに入れようかと考えると生活が円滑になります。

筆者の場合、生産的・創造的な仕事は午前中が捗るとわかっているので、午前セクションに入れるようにしています。もっと言えば「1: 起床後から出勤までの間」、「2: 自分が出勤してから皆が出勤するまで」、「3: 皆も出勤してきた後の午前残り」と 3 つのセクションがありまして、1 にて一番やりたいこと、2 にて仕事で集中してやりたいことをやるようにしています。3 は同僚からの割り込みがあったり、会議があったりするのであまり集中できません。

コンテナのパピプペポ

コンテナのパピプペポとは、コンテナとしてありがちな使い方をパピプペポで整理したものです。

1文字	カナ	英語	意味
パ	パーパス	Purpose	目的や目標。大タスクや中タスクになる
ピ	ピープル	People	人。どの人に何を任せるか、何を言うか（インプットしておくか）
プ	プレイグラウンド	Playground	遊び場。遊びたいこと、やりたいことを雑多に入れておく。おもちゃ箱
ペ	ペンディング	Pending	待ち。何らかの事情で待ち状態になるものを入れておく。GTDでいう「連絡待ち」
ポ	ポスト	Post	受信箱と送信用下書き箱。届いたものや外に出す（発信や投稿や提出）ものを一時的に溜めておく

パーパスやポストやペンディングのような真面目な用途もあれば、プレイグラウンドのような不真面目に見えるものもあります。

ピープルは半ばこじつけですが、人をコンテナとみなして、誰に何を入れておこうかと捉えています。人はツールではないので不確実性が高いですが、同時に人という存在はツールよりも覚えやすく親しみやすいので、案外役に立ちます――というより、ツールよりも人で捉える人の方が多いかもしれません。筆者は「コミュニケーション1.0」と呼んでいますが、人には「常に特定の具体的な誰かを想定する」というメンタルモデル（行動特性）があると思っています。情報共有の観点では皆に見える場所に情報を置くことが重要ですが、これが出来ない人が多いです。なぜかというと、1.0のメンタルモデルでは扱えない行動だからです。タスク管理もそうで、特定の人を想定するとは限らないため、1.0にとらわれている人にはやりづらいと考えられます。なので、開き直って「誰に何を言うか」に帰着させてもいいわけですし、ツールでやりたいなら、たとえば上記のコンテキストタグを使って「Aさん」タグをつくって、Aさんが絡みそうなタスクにこのタグをつける、とかをしてもいいわけです。ピープルコンテナは、1.0の人には親しみやすい考え方だと思います。

と、このように、コンテナの扱い方に決まりはありません。真面目だろうと不真面目だろうと、あるいは厳密だろうと雑だろうと、やりやすいやり方でいいのです。

まとめ

- コンテナとはタスクを分類するもの、あるいはしたもの
- 本質的には分類の性質が当てはまるが、タスク管理の文脈ではさらに特徴がある
  - タスクのみを分類したものと、タスク以外も含めたもの（リッチコンテナ）がある
  - 箱に入れるモデルと、性質を付けるモデルがある
  - 「それはタスクか？コンテナか？」の判断は人次第、状況次第であり、正解はない

    - 自分で分類をつくってもいいし、雑に分類してもいい
- 既存の用語や活用事例も色々あるので、押さえておくと生かしやすい
    - フォルダ、カテゴリー、プロジェクトなどの用語
    - コンテキストタグやセクションなどのセオリー
    - コンテナのパピプペポ

=== Chapter-11 イベント ===

イベントの概要

概要

イベント（Event）とは開始時間と開始場所の少なくとも片方を拘束され、かつ終了のタイミングも指定されているか調整が必要となる事柄を指します。いわゆる「予定」のことです。別の言い方をすると「参加が伴うもの」とも言えます。

イベントは拘束が重たいため、基本的に事前に調整するべきです。アポを取るともいいますよね。一方で、割り込まれたり雑談から延長したりといった形で不意に発生することもあります。ちなみに雑談自体もイベントと捉えることができます。開始まわりの拘束はありませんが、終わらせるために空気を探ったりしますよね（暗黙の拘束）。

イベントの本質は儀式です。言ってしまえば、「今からここでこのようなことを始めるぞ」と決めて、皆がそれに合意して参加しているわけです。参加自体にそれなりのコストが発生します（指定時間に指定場所に集まる等）し、参加中も拘束され続けますし、終わるときも指定の終了タイミングまで待たないといけないか、あるいは自分からタイミングをうかがう必要があります。要は**面倒くさい事柄なのです**。

イベントの種類と対処

イベントはどこが拘束されるかで分けることができます。イベントでないものは□をつけます。

No	開始が拘束される	最中に拘束される	終了が拘束される	例
1	y	y	y	予定全般
2		y	y	雑談の延長で打ち合わせを始める
3	y		y	リモートワーク
4			y	閉館の概念を持つ場所
5	y	y		デスマーチ、監禁
6		y		雑談など会話全般
□7	y			-
□8				-

本項では各種類の概要と、タスク管理としてどうアプローチすればいいかを解説していきます。

なおタイプ 7 と 8 はイベントではないため扱いません。7 については、開始だけが拘束されるもの、というイメージしづらいカテゴリなので割愛します。8 については、何の拘束もない事柄――つまりイベントではない事柄なので扱いません。

1 予定全般

No	開始が拘束される	最中に拘束される	終了が拘束される	例
1	y	y	y	予定全般

基本的にはすべて拘束される 1 が多く、予定という言葉もこれを指すことが多いです。開始時間があって、開始後も終了時間までは拘束されます。

タスクとして管理するのは難しいので、カレンダーで管理するべきです。開始を忘れないことが何よりも重要であり、そのためにはいつ何があるかを俯瞰して頭に入れておく必要があるのです。必要に応じてリマインダーを使ったり、そもそも誰かや雰囲気が教えてくれる場に居たりすることでも防げます。つまり **自分が忘れていたとしても誰かが or 何かが思い出させてくれれば良い** とも言えます。

実は集団で同じ場所で過ごす意義はまさにここにあって、自身に予定どおりに動ける力がなくても、誰かが教えてくれたり皆が動いているのを見れば思い出せるのです。学校生活の移動教室にせよ、職場の会議室への移動にせよ、極めて身近なやり方でしょう。

2 延長による割り込み的な開催

No	開始が拘束される	最中に拘束される	終了が拘束される	例
2		y	y	雑談の延長で打ち合わせを始める

次に、開始のみが拘束されないケースは 2 であり、たとえば雑談しているときに打ち合わせを始める等のケースです。これは事前に決められた開始時間や場所に行くというよりも、その場で「今から始めようか」「わかりました」と始まっています。あるいはいつの間にか始まってしまっていたり、明示的に同意したわけではないが始められてしまったりといったこともあります。

タスク管理としては予定外のイベントであり、いわゆる **割り込み** と呼ばれるものの一種です。できればその場で参加するか、やめるかを判断して伝えたいところです。難しい場合は、いったん休憩をはさんで（体裁を保つために言い方や言い分は工夫が必要かもしれません）一時的に離れます。しかし、人やチームによってはそのような行動を「空気が読めない」とみなす場合もあります。良くも悪くも読み合いです。もちろん、心理的安全性の担保された職場など信頼関係が十分な場合はこのような心配は要りませんが稀でしょう。特に一方は「信頼関係がある」と思っていて、他方はそうは思っていないこともよくあります。

最も悪手なのが、この割り込みをとりあえず受け入れてしまうことです。受け入れてしまうと以後の予定がすべて狂いますし、この割り込みによるイベント自体もいつ終わるかがわかりませんし、そもそも急に始まったものなので集中もあったものではありません。生産性はもちろん、単に QoL の観点でも望ましくないのです。このタイプ 2 のイベントが多い職場やチームは要注意だと筆者は考えます――と書くとわかりづらいかもしれませんが、要は割り込みが多いわけです。割り込みが多いのは好ましくない、と言えばしっくりくるのではないでしょうか。

例外として、日頃から積極的に雑談を行ない、そこから仕事に関するイベント（打ち合わせなど）につながることも許容する、むしろ狙うといった合意が取れているのなら問題ありません。例としては GitLab 社の Coffee Chat があります。社員はいつでも誰にでも雑談ミーティングを申し込めるようになっています。2024/06/11 現在、ハンドブックを見ると「週に数時間程度を推奨」「日頃は仕事の話ばかりだからこそ雑談や共有を」とありますが、いつでも誰にでも雑談を申し込むとの部分は汎用的に使えるでしょう。うちはいつでも誰でも雑談していいことにするよ、喫煙室や給湯室での雑談も別に一日複数回とか一時間とかしてもいいよ、そこから仕事の話に繋がることも多々あるからね、などと決めてもいいわけです。もっとも注意深く設計しないと、一部の権力者とその取り巻きによるノリが形成されてしまって政治的になってしまいますが。

3 リモートワーク

No	開始が拘束される	最中に拘束される	終了が拘束される	例
3	y		y	リモートワーク

最中のみ拘束されないケースは 3 であり、わかりづらいですがリモートワークが当てはまることがあります。

リモートワークのあり方として「勤務開始時と終了時にのみ連絡を行う」「その間どこでどう過ごすかは自由」というものがあり、このタイプ 3 はこれのことを言っています。実際はちゃんと仕事しなければならないわけですが、少なくとも職場という場所には拘束されていません。自宅でもいいですし、ワーケーションとして避暑地や観光地にいてもいいわけです。

このようなあり方は昨今珍しくはなく、裁量が大きくて自律的に働ける仕事やチームでは実現できます。ただし、単にリモートワークやフルリモートをしているからといって、これであるとは限りません。少なくともオンライン会議が多い場合はここには当てはまりません。目安はミュートデイ――一日全く打ち合わせがない日が当たり前にあるかどうかです。雑談やちょっとした相談などは問題ありませんが、それらもすべて廃した、純粋に一日誰とも喋らない日というものをやろうと思えばできるかどうか、です。これにイエスと答えられない場合、3 は満たしていません。

さて、タスク管理ですが、まずこのような自律的な働き方を支えられる程度に、各自が各自の管理を行わなくてはなりません。筆者は以前、同僚から「吉良野さんはチャットのすべてのメッセージに全部答えてくれるからすごい」と言われましたが、3 のあり方を実現するならこれくらいは当たり前にできなくてはいけません。要は、すべてのメッセージを理解して、現実的な時間で返事を出せる程度のタスク管理を「自分で」しているということです。

次に、こちらの方が重要ですが、自律的に働けるよう働き方や仕事自体を調整する必要もあります。たとえば普段から検討内容は Wiki に書き込んでいこうとか、投票や議論は 1-トピック 1-ページで書いて、全員必ず意見（特になければ無い旨）を書こうとか、孤独に耐えられなくて迂闊に打ち合わせをしたがるけどそういうのはやめよう、あるいは必要に応じて誰かに申し込んで全員同意した場合に限りやろうとか、そもそも耐えられない人向けの耐え方や満たし方もナレッジとして貯めよう、必要なら教育もしよう――といったことです。これらはやり方の話ですが、場合によってステークホルダーとの調整や交渉も要求されます。すでに何度か盤外戦との言葉で述べていますが、望みの過ごし方を実現するためには、それを阻む根本自体への対処がどうしても必要です。

4 閉館時間を持つ場所

No	開始が拘束される	最中に拘束される	終了が拘束される	例
4			y	閉館の概念を持つ場所

4 は終了のみ拘束されるケースであり、これもわかりづらいですが、閉館時間や閉店時間を持つ場所が相当します。店舗でもなんでもいいですが、いつ来てもいいし、どう過ごしてもいいが、終わりの時間はあるため永遠には居られないというものです。

タスク管理的には、このような場所は「終わりの時間まで使うことができるリソース」と捉えて、終わりが来るまでに最大限使うには、と考えるといいでしょう。たとえば空いている時間帯に使えれば快適だ → 空いてる時間帯に行くには？ → その時間帯を空けるには、とブレイクダウンしていけば、自分のタスクを調整して空ければいいのだとわかります。

筆者の例で言うと、人混みが嫌いなのと、朝型体質なのとがあって、普段は 5 時起き 21 時就寝、仕事も朝 6 時から 15 時くらいで勤務にしています。この生活リズムですと昼休憩が 9〜10 時、終業後の夕食が 15〜16 時になるので、店も空いているのです。この生活をキープするためにはタスク管理をフル活用しています。制約として拘束の強い仕事はできません、これはつまりチームプレイも全般的に不得手というわけで、仕事はかなり選ばなくてはならなくなります、それでも仕事を手に入れるための鍛錬、鍛錬のために計画もまたタスク管理して――とそれなりに苦労しています。

と、筆者の例は極端ですが、一般的には**「終わりの時間が決まっている場所」の使い方を自分なりに調整する営み** と言えます。タイプ 1 の典型的な予定は「とにもかくにもまず開始を守る！」と注力していましたが、このタイプ 4 はそうではなくて、終わりの時間までそれなりに時間があって、いつどれだけ使うかが自由に決められるので、自分に適した使い方を模索しましょうとなるわけです。

かんたんな例を言うなら、昼休憩で通ってる店に混む前に行きたいが、普通に 12 時の休憩を待つと確実に間に合わない、15 分くらいフライングできれば混む前に席を取れるとわかった、なら実際フライングを取るためには――とかはどうでしょうか。おそらくあなたは 11 時 45 分までに午前の仕事を終わらせる要領を身につけたり、15 分フライングしてもいい旨をチームや上司に共有して合意してもらったりが必要でしょう。裁量が高いのであれば堂々とできるかもしれませんし、ルール的には OK だが職場の雰囲気的に難しいならこっそりやれば見逃してもらえるかもしれません。ちなみに、閉館の概念はここでも存在するはずで、おそらく店が 13:00 で閉まるとか、昼休憩が 12:59 で終わるとかいったことがあります。4 のケースとして捉えるなら、これらは守らねばなりません。

と、このようにタイプ 4 のイベントに勝つための調整行動には、そもそもその時間帯に行動できるように過ごし方を変えることと、その過ごし方を実際になぞれるように自己管理することの二点が求められます。

5 エンドレス

No	開始が拘束される	最中に拘束される	終了が拘束される	例
5	y	y		デスマーチ、監禁

開始も最中も拘束されているのに、終わりが拘束されていないシチュエーションとは何でしょうか。「いつでも終わってもいいですよ」とのパターンは考えづらいし、あったとしても好きに帰れるなら問題無いため割愛します。現実的には終われるまで帰れないという過酷なシチュエーション

でしょう。

例としてデスマーチ（恒常的な過重労働）を挙げていますが、プロジェクトが炎上しているケースや、要領が悪かったり優柔不断だったりするメンバーや顧客と重要な仕事をしているケース、終了タイミングが横暴な権力者の気分によってコントロールされていて嫌がらせや洗脳を受けているケースなどがあります。いずれにしてもブラック企業のブラックの形容詞が似合う有り様ですし、より率直で極端な例として監禁も入れてみました。

このタイプ５のイベントは地獄でしかなく、タスク管理でどうにかなるものではありません。盤外戦として根本に働きかける――そもそもこのように陥らないようにする、近寄らないようにする、もし陥ってしまった場合は早々に逃げるか、刺し違えでも戦って抗うかといった行動が必須です。偶然終わることもありますが、神頼みでしかありませんし、その前に心身を壊しかねません。盤外戦をできるかどうかがすべてです。そして、盤外戦をされると逃げられることがわかっているからこそ、ブラック企業なりカルト集団なりマルチ商法なりホストクラブ（など一部の客商売）なりはこれを封じてきます。情報を遮断し、他の人に頼ることも許さず、閉鎖的な環境でアメとムチを使い分けて報酬を制御し、そもそも思考させないためにノルマや通いを課して疲弊させ、としてくるのです。人間の感情に訴えるために非言語情報を浴びせてくることも特徴で、そのためには現地で実際に会う必要もあるため、場所的な拘束も入りがちです。

こう書くと非日常的で自分には無縁と思われるかもしれませんが、意外とそうでもありません。軽めのシチュエーションではよくありますし、そうでなくとも昭和の時代は長時間労働が常態化していました。その名残は今でもありますし、その世代の人達が権力を握っているので、ともすると「終わるまで残業してでも働きなさい」なんてことになりかねません。盤外戦としては、そもそも残業を常態化させないと務まらない状況からしておかしいから改善するべきだ、となるのですが、この視点が通じないことはざらにあります。第一、自分が望んで仕事を受けてその状況になってしまった場合は、仕方ないと耐える道を選ぶ人も多いのではないでしょうか。

と、このタイプ５のイベントは意外と逃れにくいので、日頃から予防的に行動することをおすすめします。自分の限界を把握してそれを越えないように生活を整えたり、タイムボックスなど働く時間をあらかじめ決めておいたり、あるいは自分の感覚が麻痺しないように日頃から色んな業界や人や働き方を調べておくとか、定期的に内省と振り返りの時間を設けて自分がおかしくないかに自分で気付けるようにするとか、もちろん気の置ける人とおしゃべりしてもいいでしょう、と方法は色々あります。過激な方法で言えば、気難しい人や変人を演じたり、そのような人と仲良くなったりなどして周囲の方から遠ざけてもらう手もあります。予防的な行動は面倒くさいものですが、そうしなければかんたんに飲まれてしまいます。

6 不明瞭な終わり

No	開始が拘束される	最中に拘束される	終了が拘束される	例
6		y		雑談など会話全般

開始も終了も拘束されず、最中だけ拘束されるというイベントがあります。雑談など会話全般がそうです。突然始まったりしますし、いつ終わるかもわからずだらだら続いたりします。

タスク管理としては、開始と終了をいかにしてコントロールするかの問題となります。といってもアプローチは 3 つくらいだと思います。

- 1: メリハリ制。自由に会話していい時間帯と、させない時間帯を設けて、メリハリをつける
- 2: 明示的提案。いつ始めるか、いつ終わるかをちゃんと明示的に提案する
- 3: ダミーイベント。架空の予定を入れて無理やり終わらせる

メリハリ制は、会話してもいい時間帯（グッドタイム）とそうではない時間帯（バッドタイム）を両方用意して、グッドタイムのときは別に拘束を気にしない、とするものです。友人や親しい人がいる場合は、十中八九こちらになります。通常は明示的に会う予定をつくったり、空き時間をつくったりして、そこで楽しみますよね。逆にグッドタイムを確保できないと（会話ができないので）関係に傷が入っていきます。時間の捻出はまさにタスク管理で行えることですし、ここを捻出したいからこそタスク管理をするものですが、万人に行えることではありません。そこで節目の儀式が重宝されます。誕生日やらお正月といった節目の儀式は、グッドタイムの典型例です。人々がこういった儀式を好むのはグッドタイムを求めているから、そして日頃から自分でグッドタイムを確保するのが苦手だからでもあります。もっと言えば、グッドタイムを意識的に設計して捻出できると強いです。まめな人はモテると言いますが、それはグッドタイムを意識的に確保しているからもであるのだと筆者は思います。

明示的提案は、始めたいときや終わりたいときにはっきりとその旨を言います。味気ないとか、ビジネスライクで嫌だとか思われるかもしれませんが、慢性的なだらだら会話によるストレスも馬鹿になりません。そうでなくとも、だらだらしたいときともういいというときはあるもので、どちらもできるに越したことはありません。明示的提案を使うには、明示的に提案できるほどの信頼関係と、提案しても通じるかどうかとの素養が鍵です。パートナータスク管理の章にてトーカビリティを取り上げましたが、まさにそうです。トーカビリティがある場合は、明示的提案を使ってもいいかどうかとか、いつ使おうかといった話し合いができます。この過程でメリハリ制が登場することも珍しくありません。

最後にダミーイベントについては、終わりを導くための方法ですが、単にダミーの予定をでっちあげてその場を抜けることです。電話がかかってきたとか、そういえば用事があるんだったとか、実世界でもフィクションでもよく見る言い訳ですね。発展的なテクニックとして、普段から忙しいキャラクターを演じておくことでもダミーイベントを適用しやすいです。両親やパートナーからの門限が厳しいとか、寮の門限があるからとか、私はストイックな生活リズムを踏んでいる変人なんです（筆者はこのタイプ）とか、ペットがいるので……などキャラクターのバリエーションは無数にあるで

しょう。

まとめ

- イベントとは開始、最中、終了を拘束される事柄
  - どこが拘束されるか次第でいくつかの種類に分類できる
- イベントの種類は豊富で、その性質も違えばタスク管理としての対処も違う
  - 予定。開始時間と開始場所を守ることが最優先。カレンダーを使うべし
  - 雑談から打ち合わせに移行する等のケース。本質的には割り込み。一息置いて冷静に対処するか、最初から許容してもいいとの文化をつくるか
  - リモートワーク。自律的に働けることとその許容が必要だが、大半の人には縁が無いほど高度でもある
  - 閉館時間を持つ場所。いつどのように利用するか、と自分の側を調整する必要がある
  - エンドレス。要はブラックなのでできるだけ回避したい
  - 会話などいつ終わればいいかが不明瞭なケース。メリハリをつける、明示的に言う、ダミーをでっちあげるなどの対策がある

=== Chapter-12 モットー ===

モットーの概要

概要

モットー（Motto）とは心がけるもの全般です。本来の意味は「標語」ですが、本書では「心がけるもの全般」を示すタスク管理用語として使います。

モットーの例

モットーの例を挙げます。人生全般に関わるものから、仕事において明日から行うことまで様々です。

- 標語
- 座右の銘

- 格言
- 金言
- 人生哲学
- 今年の抱負
- 今月の抱負
- 次から気をつけること
- KPT など振り返りで導いた「次やること」

応用と格言問題

モットーの性質はただ一つで、正しく使うことが難しい――この一点に尽きます。

たとえば仕事で上司から「上位上司や幹部や社長にいきなりメッセージを送るな」と怒られたとします。モットーとして「直属上司の先の、目上の人とは勝手にコンタクトを取らない」を掲げたとします。これを守るにはどうしたらいいでしょうか。言葉で書くと、然るべきときに思い出して、然るべき行動を取り続ける、となります。この場合は外の目上にメッセージを投げるときに、このモットーを思い出して、そのとおり投げる行動をやめる、となるでしょう。そのためにはメッセージを投げるときにちょうどこのモットーを思い出せなくてはなりません。また、思い出しただけでは意味がないので、投げない、との行動もちゃんとしないといけません。

個人タスク管理の章にて忘迷怠を取り上げましたが、人は忘れるし迷うし怠ける生き物です。特に普段とは違ったことや気が進まないことは中々行動に移せません。この例については、「それくらいできて当たり前でしょう」と考える方も多いかもしれませんが、人によっては決してそうではないのです。

特にモットーは当てはまるシチュエーションがいつ来るかがわからず、事前に備えることができません。タスク管理の場合ですと、タスクという言語化された情報を上手く仕込んで目に入れればオッケー（あとは行動すればいいだけ）でしたね。目に入ったときに行動できるとは限りませんが、それは行動しやすくなるように生活やタスクを組み立てていけばいいだけのことです。それがまさに個人タスク管理という営みです――が、モットーに対しては通じないのです。

かといって常に頭の中で「直属上司の先の目上の人とは勝手にコンタクトを取らない直属上司の先の目上の人とは勝手にコンタクトを取らない直属上司の先の目上の人とは勝手にコンタクトを取らない……」と唱え続けるわけにもいきませんし、紙に書いて貼り付けてそれをずっと読み続けるわけにもいきません。タスク管理的に言えば「直属上司の先の目上の人とは勝手にコンタクトを取らない、と10回音読する」タスクを毎日朝に仕込む、なども可能ですが、やはり意味がありません。せいぜい暗唱できるようになるくらいですし、このやり方はそのうち麻痺して形骸化します。

モットーの適用は難しいです。当てはまるシチュエーションが来たときに、適切に思い出して行動できる必要があるからです。これを **応用** と呼ばせてください。勉強や試験でも応用問題と言いますが、まさに同じことで、何が使えるかを適切に思い出せないといけません。**基礎ができ**

ているからできるとは限りません。基礎ができていても、その基礎を思い出せないと意味がないからです。前の章にてタスク管理はトレメントゆえに難しいとの話をしましたし、正直言って筆者はある種の才能が必要とさえ考えていますが、モットーについても同じことが言えます。応用という才能が必要です。

このようにモットーを使いこなすのは難しいのです。これを **格言問題** と呼びます。格言を使うのは難しいのです。

アスリートとアプライア

タスク管理には 2 種類の才能があると考えます。

一つはトレメントと愚直に向き合える才能です。たとえば毎日朝に 1 時間の散歩をやりますと決めて、実際そのとおりに行います。面倒くさくても、苦しくても、基本的にはやります。誰に言われるとか報酬があるとかは関係なしに、自分の意志で行ないます。自己啓発ではよく「意志や決断ほど役に立たないことはない」との言い回しと出会いますが、それはトレメントと向き合う才能がない人の言い分です。トレメントと向き合うタイプの人を **アスリート** と呼びます。

もう一つが、上述した格言問題をこなせる才能です。一度モットーを掲げたら、あとは状況に応じて、必要に応じて適切に思い出し、そのとおりの行動をしていくことができます。応用力が高いとも言えるでしょう。格言問題を容易くこなせるタイプの人を **アプライア（Applier）** と呼びます。

筆者の持論ですが、アスリートとアプライアは両立しないと感じます。

筆者は典型的なアスリートであり、その空気感は本書をここまで読んできた読者の皆さんも感じていると思います。個人的にも毎日一定の生活リズムで過ごすのは苦ではないですし、むしろリズムが乱れることを嫌うまであります。毎日同じ起床時間と就寝時間を守ったり、律儀に1日3食食べたり、風呂に入ったり、掃除したりといったことは難なく行えますし、むしろできない人達がなぜできないのかがわかりません。こんなかんたんなことなのに一体何に苦戦しているのだろう、と宇宙人を見るような気持ちです。逆に格言問題は苦手で、上記の例も実は筆者の実例なのです。筆者が上記の格言問題に対処できるようになるためには異文化理解力という本との出会いを待たねばなりませんでした。同書では日本を含む各国の文化や価値観を言語化しており、なぜ日本では直属上司を越えた先に直接コミュニケーションしにいってはいけないかも書いてあります。なるほど、そういう文化があるからなのか、文化ならまあ仕方ないなと理解して納得できたことで筆者はようやく腑に落ちました。ちなみに同書では「階層主義」という言葉が使われています。階層的な社会であり、直下と直近だけを相手にしなさい、その先まで行くのは越権にあたります、とのモデルになっています。

話を戻しましょう。一方でアプライアの方も数多くいらっしゃいますし、体感的にはこちらの方が多い気さえしています。仕事の場はよく忙しくなりがちで、OJT という言葉もありますが、これはアプライアの価値観に基づいています。アプライアは応用が得意なので、忙しいシチュエーション

でも一度学んだことは応用していきやすいのです。逆に、そのような人は忙しさに溺れたり場当たり的にこなすことを好みがちで、アスリート的な過ごし方は苦手です。応用できればどうとでもなるからトレメントに従う必要性がそもそもない、ないから納得感もないわけですし、そもそも応用できる自分の強さに従ってばんばん仕事や予定を突っ込むので基本的に忙しくて、トレーニングを行ったりといった余裕がなくなるのです。

もちろん何事にも例外はあります。特にプロと呼ばれる人達は、アスリートでありながらもその種目や仕事についてはアプライア的にガンガン応用していきます（できないとプロとして通用しない）し、逆にアプライアであっても規則正しいトレーニングや生活リズムを踏んでいたりもします（こちらは場やチームやトレーナーなどの力に頼っていることが多い）。

ですが、プロは例外的なものであり、筆者としては才能と呼んでもいいものであり両立はしない――基本的にはどちらか片方に偏るものだと考えます。

モットーはどう扱うか

アスリートとアプライアとで分かれます。その前に両者に共通する扱い方もあるので先に取り上げます。

モットーを扱う際の原則

アスリートにもアプライアにもどちらに当てはまるものです。

まずはありきたりですが、詰め込みすぎに注意します。モットーが多すぎたり、状況があまりに忙しかったり、あるいは単に疲れていたりするとさすがに応用しづらくなります。モットーを掲げすぎている（or 仕事など軽微なモットーが発生する立場が多数ある）ならある程度は捨てた方が良いですし、忙しいときは慣れた行動や反射的な思考が主戦力になるのでモットーどうこうとのんびりしてはいられません、つまり一時的にモットーの適用を諦めた方がいいかもしれませんし、疲れているならもちろん休むべきです。ありきたりですね。

それから納得感も大事です。結局私たちは納得してないことはできません。納得したふりをして、とりあえずモットーにしてみることはできますが、どうせ定着しません。納得感を増やすためには調動脈――調子と動機と文脈を追求するのが良い、とはすでに述べました。特にアスリートの人は「正直全然納得してないけど、皆が言っているし、とりあえず従ってみるか」と抱えたがりがちですが、アンチパターンです。どうせ（上手く応用できないので）定着しません。納得感を得るには、モットーとして抱えるのではなく、その調動脈に目を向けた方が良いです。特に重要なのが文脈で、なぜそのモットーが必要なのか、そのモットーを掲げている人達はなぜ納得できているのか、といったことを理解できるかが鍵になります。そのためには落ち着いて内省したり、対話したり、あるいは本などで勉強してみたりできればいいのですが、ここでも余裕が要ります。特に自分の価値観や傾向を知る自己理解がポイントで、ここがないと納得感は醸成できません

（感情に振り回されるか他者の言いなりになってしまいます）。そういう意味でも、やはり詰め込みすぎには注意したいのです。

というわけで、総じて詰め込みすぎを避けて余裕をもたせましょう、と言えます。

アプライアから見たモットー

アプライアには応用の能力がありますので、格言問題に悩まされることは通常ありません。

強いて言えば、粒度の粗いモットーは行動しづらいので分解しましょう、というくらいです。たとえば「誠実な人になる」とのモットーはざっくりしすぎて、さすがに行動しづらいでしょう。誠実の定義や構成要素を洗い出して、もっと具体的にした方が良さそうです。「約束は必ず守るし、守れない場合はその旨を必ず伝える」はどうでしょうか。例外が欲しいなら「心身上の危険が迫っている場合は問答無用で逃げる」とか「最悪あとで誠心誠意謝ればいい、壊れたらそれすらもできなくなる」とかいったモットーも追加すると死角がなくなります。

とはいえ、アプライアは応用の能力が強いので、どちらかといえば粗めが良いと思います。粗めの方が何かと融通が利きやすいし、アプライアにとっては理解もしやすいのです。上記の例はアスリートの筆者による具体化であり、おそらくアプライアの人にはあまり刺さらないでしょう。よく使われるのは、具体的な人（フィクションでも構いません）を思い浮かべて、この人ならどう捉えるだろうか、と考えることです。誠実な人としてＡさんが思い浮かぶのなら、Ａさんをモットーにします。言語化されているわけではありませんが、自分が応用できればそれでいいので問題ありません。

アスリートから見たモットー

アスリートはおそらく応用の能力が乏しいので、工夫が必要です。

一番かんたんなのは環境の力に任せてしまうことです。上司でもチームメンバーでもルームメイトでも何でもいいですが、高頻度に素早くフィードバックをくれる存在と一緒に仕事をすれば、応用できなくてもその場で教えてもらえます。度が過ぎると仲がこじれてしまいます（よくあるのは「何度も同じ失敗をしやがる」ですかね）が、孤軍奮闘よりは応用しやすくなるので意外と何とかなります。こう書くと、最初から環境に頼っているようで他力本願ですが、応用に向いていないので仕方ありません。むしろ、アスリートはいかに早く環境に、特に人に頼れるかが勝負なところがあります。上手くいけば、自分でモットーを運用することなく、指示に従うだけで過ごせるような快適な環境が手に入ります。ここも行き過ぎると「管理」「隷属」「思考停止」になってしまいますが。

次にモットーを一つずつ相手にして仕組み化していく作戦も使えます。モットーを上手く仕組みに変換し、その仕組みを定着できたら、もう忘れません。上記のコンタクトの例を再び使うと、たとえば「仕事中に私が自己判断で連絡してもいい人リスト」なるものを導入します。ここに自分

の直属上司とチームメンバー、また許可してもらった人や利害関係者外の人（一緒に入社した同期やイベントで知り合った社員など）などを追加していき、普段は常にこのリストを見て連絡を行うようにします。要は頭で判断するのをやめて、リストの中に入っている人にだけ送っても良いルールにするわけですね。ちなみに、リスト内にない人と連絡したいときは、まずは上司に確認して、それで許可をもらったらリストに追加します。追加したら以降は自己判断で送れるようになります、なぜならリスト内に記載があるからリストを見た自分が送っても良いと判断するだろうからです——と、かなり機械的に仕組み化していますが、こうすることによって仕組みに従うことに帰着できています。このようにして定着できたら、もう忘れませんし、一度定着したら仕組みを多少端折っても上手くいきますので融通も利きやすくなります。ここまでできたら、次のモットーに着手します。このようにして、一つずつ定着させていくのです。

筆者としてはもっと上手いやり方を模索しているところです。モットーをランダムでリマインドするツールはどうかとか、モットーの適用を映像化したり小説家したりして「豊富な情報」として記憶するのはどうかなどを検討していますが、まだまだ未完成です。アプライアのようにバリバリ応用できるようになりたいものです。

モットーをどうつくるか

モットーには他者からもたらされるもの（**エモットー, External Motto**）と、自分が掲げるもの（**イモットー, Internal Motto**）があります。

どちらもモットーには違いありませんが、どちらが適しているかは人次第です。

行き来する

エモットーだとわかりづらいから自分なりにアレンジしたり（イモットー化）、逆にイモットーをつくったけどいまいちわかりづらい、どうしよう、あ、このエモットーはわかりやすいし似ているからこっちに寄せよう（エモットー化）といったことも可能です。

すでに述べたとおり、モットーには納得感が大事ですが、その納得感は自分なりに理解できるかにかかっています。そしてその理解は言語化にかかっています。**エモットーとイモットーはどちらも使うのが望ましいですし、むしろ行き来するのが望ましいです。**

どうつくるか

エモットーについては、古典や偉人の格言に限らず、マンガだろうと周囲の人だろうと何にでも目を向けた方がやりやすいと思います。前者の優れた格言は抽象的すぎますし、自分の生活とも紐づいてなくてピンと来づらく、応用しづらいです。それよりもXでバズっているツイートとか、身近な人に言われたはっとすることなどの方が理解もしやすく、納得もしやすいことはずです。かと

いって、後者ばかりに依存していると視野が狭くなってしまうので、前者の格言とも見比べてみたり、関係を整理してみる（例: これって結局「誠実であれ」ってことだよなー）などして自分なりに洗練させていきます。

イモットーについては、とにもかくにも言語化です。まずは日記などで自分の感情や思考を書くところから始めます。慣れてくると、こんな感じの考え方なるものがあるから仮に XXX と名付けよう、みたいなこともできるようになってきます。向き不向きがあり一向にできない人もいますが、重要なのは言語化です。頭の中で思っているだけでは何も変わりません――と言いたいところですが、それでも応用できてしまう感覚派もいます。向き不向きですね。

イモットーは個人的なものですから、遠慮は要らず、自分にとっての納得感があればそれでいいです。遠慮して言語化しないのはもったいないので、迷っているならとりあえずやりましょう。あとから洗練していけばいいのです。とはいえ、こちらも溺れすぎると認知が偏ってしまうので、エモットーと比べて微調整していきたいところです。ここで「他人に相談してフィードバックをもらってもいいのか」と思うかもしれませんが、おそらく通じないし、むっとすると思うので控えた方が良いでしょう。イモットーはあくまでも自分の内にとどめておいて、でもそれだけだと偏ってしまうから、エモットーも参考にして微修正する、とのバランスになります。

エモットーは参考程度に

エモットーは実はかなり漂白されています。本当はその人自身の個人的な思いや細かい例外などがたくさんついているはずですが、その辺は綺麗さっぱり取り除かれていて、口当たりのいいフレーズだけが残っています。

そういう意味で、エモットーは、それ単体では実はあまり使い物になりません。モットーを聞いてもよく「ふーん」となって終わると思いますが、まさにそうなります。特にことわざはそうで、「早起きは三文の徳」と聞いても「ふーん」「そりゃそう」としかならないでしょう。

「いや、中にはものすごく共感できるものもあるよ」と思う方もいるかもしれませんが、それはモットーというよりも共感事項ともいうべきもので、モットーとは違う可能性が高いです。昨今は SNS などで注目を集めてお金を稼ぐ時代ですし、自己啓発や人生に関するトピックは依然として需要もあるため、モットーの皮を被った共感事項はよくあります。特に「男は」「女は」「日本は」「海外は」「東京は」「田舎は」「若者は」「老人は」といった主語の大きなものには要注意です。

大事なのはエモットーを踏まえた上で、それを自分なりに解釈することです。「早起きは三文の徳」にしても、早起きとは何かとか、三文とはどういう意味なのかとか、徳とは何か、得じゃないのかなどはすぐにでも調べられますし、自分の経験と照らし合わせると色々見えてきます。答えは唯一ではなく、仮説として定めることもありますが、それはそれでいいのです。あとでまた修正すればいいのです。夜型人間の人が「時間帯は関係がない」「起床直後のリフレッシュした頭を有効活用することが大事なのだ」と解釈してもいいわけですし、ここまで来ると「起床直後は最もやりたいことに時間を使え」のようなイモットーに昇華できます。これが正しいかはわかりませ

んが、正しくなかったらまた直せばいいだけの話です。

まとめ

- モットーとは心がけるもの全般
  - 標語以外にも格言、抱負、次から気をつけることなど色々ある
- モットーは適切に使うのが難しい
  - 然るべきときに思い出し、かつ行動に起こすことが必要
    - これを応用と呼ぶ
  - タスクを認識した後に行動すればいいだけのタスク管理とは異なり、応用は難しい
  - この問題を格言問題と呼ぶ
- 実はタスク管理の才能は 2 種類あり、アスリートとアプライアに分かれる
  - 格言問題はアスリートが直面する問題
- 格言問題に対処するには
  - まずは納得感を増やす
  - アプライアなら基本的に困らない
  - アスリートの場合、環境の力に頼るか、一つずつ仕組み化して定着させるか
- モットーをつくる際は、継続的な営みにするとよい
  - 外からも借りる（エモットー）し、内からも考える（イモットー）
  - 一度掲げて終わりではなく、継続的に微修正する

=== Chapter-13 メモ ===

メモの概要

概要

メモ（Memo） とは、あとで処理することを想定した一時的な記述を指します。

今すぐ処理はできないが、何もしないと忘れてしまうためささっと書いておくのです。もちろん、書いただけでは意味がないため、あとで読み返さなくてはなりません。また読み返すだけでは意味がなくて、そのメモが想定しているはずの行動もちゃんとするべきです。最後に、メモがいつまでも残っていると邪魔ですから処分も必要です（問題ないなら放置も可能）――と、言葉にするとなんだか大げさですが、メモを正しく扱うためにはこの流れを押さえておきたいです。

メモのフロー

メモの流れを図でも示します。

memoflow

書く → 読み返す → 行動と処理、の 3 段階になっています。

読み返すときは一度の読み返しで次に進むのが理想ですが、そうならないときもあります。つまり同じメモを複数回読み返すことがあります。たとえば「読んだけどよくわからなかったのでスキップしよう」と明日以降に先送りしたりします。

行動と処理については、行動とはメモが表す内容に取り組むことであり、処理とはメモ自体をどうするか（捨てるか消すか放置するか等）という話です。たとえば新しい水筒を買いたくて「水筒」というメモがあったとき、行動というと Amazon で水筒を探してブクマしたり、今すぐ近所の店に出かけて買いに行ったり、今日の退勤時に買いたいからとリマインダーを仕掛けたりといったことを指します。処理というとそのメモを消すのか、放置するのかといったことです。付箋で書いて貼り付けているならその付箋を捨てるでしょうが、日記に書いているとするならおそらく放置するでしょう、あるいは打ち消し線を引くかもしれません。

メモは任意

メモのフローにおいて重要なのは、行動はしないが処理するケースもあれば行動はするが処理しないケースもあるということです。もっと言えば行動しなくてもいいし、処理しなくてもいいわけです。別の言い方をすると、行動できたらいいな、処理できたらいいな、です。**できたらラッキー**の気持ちですね。

メモとタスクは違います。タスクはやらなきゃいけないものもありますが、メモはそうではありません。もしやらないといけないメモがあるのだとしたら、それはタスクとして扱うべきです。

一般的にメモはたくさんつくられます。慣れてない人や必要ない人はほとんど書かないと思いますが、慣れてくると「書けば書くほど便利」になるので、メモはどんどん増えます。たとえば後でやりたいことが 1 日平均 10 個浮かぶとしても、メモを適切に扱えれば **全部回収できます**。1 年

で 3650 個のメモを扱えることになります。もちろん全部に対して行動できるわけではありませんが、それでも 3650 個を扱えているわけです。全く扱わない人との差（たとえば QoL の差）は歴然でしょう。さて、3650 という数字を見てもわかるように、メモを全部律儀に処理できるかというと、まず不可能です。よって、できたらラッキーくらいの気持ちで割り切ることが求められます。そうでないと不出来な自分に落ち込んでしまい、メンタル上よろしくありません。完璧主義な人は挫折しがちですが、そもそも完璧主義を捨てねばならないのです。

メモの扱い方

ではメモはどのように扱えばいいのでしょうか。またタスク管理はどう役立てればいいでしょうか。フローの各ステップごとに解説します。

メモを書くとき

最も重要なのは素早さです。一秒でも早く書き始めることができて、一秒でも早く書き終えられるのが望ましいです。一秒どころか 0.x 秒でさえも突き詰めたいです。それくらいのつもりで、とにかく素早くメモを取ることを目指してください。

まずはツールです。手帳とペンなのか、付箋や紙とペンなのか、スマホのメモツールなのか、PC 上でテキストエディタなのか、あるいは何らかの Web アプリなのか。デジタルの場合、入力方法もフリック入力なのかタイピングなのか、タイピングの場合はローマ字入力なのか、かな入力なのかなど様々です。ボイスレコーダーだったり、作業風景を常時録画したり（「今からメモを喋ります」と明示的に言ったり、人差し指を立てている間に喋るようにしたりすれば、後で見返したときにわかります）といった方法もあります。どんなやり方が最適かは人次第、かつ状況次第ですので、自分で模索することになります。

次に言語化をどの程度行うか、にも傾向が出ます。基本的にちゃんと書けば書くほど後で思い出しやすいですが、その分、面倒くさいです。逆に単語だけ書くなど手抜きをすれば書くときは楽ですが、後で思い出せなくなる可能性も上がります。どれくらい書けば最低限思い出せるかは自分次第なので、自分のクセをつかめると良いでしょう。また、そもそも言語化するのかという話もあります。本質は行動したいことをあとで思い出すことですから、写真を取っておいたり、印をつけておいたり、イラストをささっと描いたりといったこともありえます――が、これらはメモの領域からは逸れるため本章では扱いません。**メモはあくまで言語化して書くことです**。描くことはメモには含みません。ということは、言語化だったり書くことが苦手だったりする人は、そもそもメモも苦手だということになります。メモという営みを諦めるか、言語化や書くことの苦手意識を減らすかが先です。

メモをどこに書くかですか、複数用意するのが望ましいでしょう。たとえば自宅と外出先と会社、と 3 つの文脈があるならこれら 3 つ分のメモ環境をつくるのが良いですね。メモはとにもかくにも、

まず素早く書くことが第一です。書いた後の取り扱いはどうとでもなるので、まずは書くことだけを――素早く書き残せるよう整備することを考えます。**自分が置かれた文脈の数だけ環境を用意しましょう。**あるいは、「スマホを常に持っているのでそっちにメモする」が可能ならそれでもよいです。というより、慣れたツールでメモするのが一番なので、狙えるならそうしたいところです。ただ実際は業務中に私有のスマホでメモできないとか、通勤中に満員電車で手帳を開くスペースすらないとか、あるいは人目があって素直にメモを取るのが恥ずかしいとかいったことがよくあり、単一ツールでは厳しいことが多いので、文脈ごと・環境ごとにメモ手段を確立してしまった方が早いのです。ここで環境を用意すると書きましたが、インボックスを用意すると読み替えても構いません。

メモを読み返すとき

メモを扱う際の最大のハードルがこの部分、読み返しだと思います。具体的には二つあります。

タイミングの壁

一つは、いつ読み返すかという**タイミングの壁** です。メモは書いている最中は最もモチベーションが高いですが、その後は（書いているときの）文脈を失ってしまうのでモチベーションが湧きません。加えて、何を書いているか一見では読み取れず粘り強く解読にあたる泥臭さも求められることがあります。気が重いですよね。ですので、「毎日 18 時に今日書いたメモを全部読み返そう」などとしても、十中八九成功できません。アスリートタイプの人なら比較的楽勝だと思いますが、それでも仕事の後で疲れていて解読や行動に繋げられない、といったことはよくあります。

つまりタイミングの壁を越えるためには「モチベーションに左右されず定期的に振り返る（メモを読み返す）」ことと、「読み返す際の解読やその後の行動に繋げる」ことの二つが必要ですが、これは要は **メモを読み返す時間を、疲れていない時間帯にしっかりと確保する** ことに他なりません（※1）。たとえば 2 日に 1 回、朝一に 15 分、メモを読み返す時間を確保できるでしょうか。もっと言うと、それを行うだけの価値がメモを読み返すという行動にはある、と思えるでしょうか――そうです、結局のところ、メモにそれだけの価値があると考えるかどうかという納得感の話に帰着されます。

とはいえ、一見すると価値がないと思える人でも、状況を限定すれば価値が出ることがあります。特に仕事では、あとでやるべきことをすっぽかすのはよろしくないことで、たいていの人が「何とか頑張ってやる」と決めて行動できるでしょう。たとえば昼休憩の前後でメモを 5 分チェックする、くらいなら採用しやすいのではないでしょうか。まとまった休憩の前後、特に直後に確保するのは自然なのでおすすめです。ただ、日頃から会議で忙しい人はそのせいで忙殺されがちなので、まとまった休憩の後、10 分くらいのゆとりは確保したいものです。あるいは開き直って休憩時間を 10 分削る手もあります。

- ※
  - 1 ちなみに「いや、単に状況が落ち着いたら速やかに読み返せばいいのでは」と思われ

るかもしれませんが、それで済むならもちろんそれで良いです。しかし、そうかんたんにはいかないと思います。一方で、単に上から降ってくるタスクを消化するだけの過ごし方をしていて、かつタスクが多すぎるからメモしておく機会が多い、という人であればこのやり方で済む可能性が高いです。もっと一般化すれば生活を「タスクを処理する時間」と「今抱えてるタスクを見返すプチ整理時間」に二分して、メモは前者の処理時間中にサクッと行っておき、後者のプチ時間で読み返す営みをする――となります。別の言い方をすれば「仕事」「休憩」の二分（WR Life）を「仕事」「休憩」「プチ整理（読み返し）」の三分にする、とも言えるでしょう。Work, Rest, Reread ということで筆者は **WRR Life** と呼んでいます。休憩の概念は言うまでもなくよく知られており、適宜はさむと思いますが、これと同じ概念として「プチ整理」も新たに加えるイメージです。仕事でもなく、休憩でもなく、その中間くらいのプチ整理という営みを新たに追加するのです。

解読の壁

もう一つは **解読の壁** です。書いてある内容を理解できなければ行動しようがないので、理解できるまで粘る必要があるのですが、ここも難しいのです。前述のとおり、メモを読み返すときはモチベーションなど死んでいますから、中々粘ろうとは思えません――し、より残酷なことを言えば、粘ったところで解読できないことも多いです。解読できない自分を責めて、と負の作用に繋がることもあります。手強い壁です。

この解読の壁を超える方法は、実は一つしかなくて、**そもそもすぐ理解できるようなメモを書いておけ**、これに尽きます。別の言い方をすると解読が不要なくらいわかりやすく書け、です。つまり読み返すときの努力ではなくて、その前の、書く段階の話になります。自分が理解できる程度に、しかし（素早く終わらせるために）最小限に済ませる、というバランスを見つけます。これは自分自身にしかできません。**メモはスキルです**。経験を積んで、少しずつ養っていくことになります。この現実と向き合えない人は、一向にメモを扱えるようにならず形骸化します。

これ以外の方法も無いことはないです。解読をやめて、解釈をします。メモとして書かれた内容を正確に解読するのは諦めて、今現在そのメモから何を読み取れるかを解釈し直すのです。その結果、メモを書いた当初とは異なる意味で理解するかもしれませんし、やっぱり意味わかんねえやと破棄することになるかもしれませんが、ともかく開き直って再解釈してしまうわけです。正解の無いアイデアを扱うクリエイターの方には適しています。

行動するとき・処理するとき

書いたメモを読み返して理解できたら、あとは行動と処理が待っています。

Act first, Manage second

行動については **Act first, Manage second（可能なら行動してしまう、無理そうならタ**

スク管理に回す）が良いでしょう。

戦略の章でスプリンターを紹介しましたがすぐ終われるタスクならさっさと行動してしまった方が良いです。2分で終われるものは終わらせてしまうとか、ボールが自分にあるのならさっさと投げてしまうなど、できることは色々あります。すぐ終われるものは管理した方がかえってコストがかかるので、さっさと処理してしまったほうがいいのです。

とはいえ、何でもかんでもすぐ終わらせられるわけでないので、たいていはタスク管理に回す必要があります。あとで適切なタイミングでこなせるよう、ツールに入れておきます。タスクリストに追加しておくとか、カレンダーに予定として入れておくとか、あるいは直近行う必要がないものならもっと低頻度に目に入る場所に置いておく程度でもいいかもしれません。

この部分、要はメモから読み取れた「行動」をどう管理するかの判断は、できるだけ素早く行ないたいです。メモを書くときと同様、ここをもたつくとメモの運用はかんたんに形骸化します。特に「目に入ったものを一つずつ行動していく」という条件反射現象はあるあるで、好例は通知が来るたびにそれをチェックして返事をする人達です。メモを適切に運用できていると、確認と行動は分離できます。確認したときにメモだけ取っておいて、あとでメモを読み返せばいいからです。これができない人が確認と同時に行動もしてしまいます。まるで条件反射のように、目に入ったものから行動していくのです。

つまり**行動とは実質的にはタスク管理（ツール）に追加すること**です。ただ、何でもかんでも追加すると後で苦しいですから、すぐ終われるものは追加せずにその場ですぐ終わらせてしまおう、というだけの話です。行動という言葉のニュアンスを裏切っているようにも聞こえますが、それでよいです。何でもかんでもその場で終わらせようとするから条件反射現象になってしまうのです。いったんタスク管理に突っ込んで、あとから落ち着いて処理する、くらいの心持ちを持ちたいところです。もちろん、そんな余裕など許されず、条件反射的に次々捌かないといけないケースもありますが、そういうケースではそもそもタスク管理など役に立ちません。あるいは盤外戦をして、そのような忙しい状況そのものを何とかしましょう。

処理≒捨てる

次に処理――行動し終えた後のメモの残骸をどう処理するかについては、あらかじめ処理方法を決めておくのが良いでしょう。その場で残したり残さなかったり、どこにコピーするかとか考えたりすると疲れるので、「この場所に書いたメモはこう処理する」と最初から決めておきます。

さらに言えば、行動し終えたメモはただのゴミですので、残さず捨てた方が良いです。付箋であればぐしゃっと丸めてゴミ箱に捨てましょう。紙のメモなら該当部分を打ち消し線などで打ち消します。デジタルであれば該当の記述を消します。私たちにはもったいない精神があり、特に日本では八百万の神や付喪神といった宗教観の影響で「なんでも大切にしがち」「取っておきがち」なので、意識的に抗うくらいがちょうどいいでしょう。

この処理フェーズはできるだけ頭を使わず、かつなるべく捨てるように倒したいところです。そうしないと、ここで使う頭の消耗がバカにならず首が締まってきます。特にデジタルだと「別に残しても

空間を占有するわけじゃないし」と残しがちですが、残していると後々目に入るたびに判断が入って認知資源を消耗します。一つ一つは軽微でも蓄積すると馬鹿になりません。「使い終えた付箋」ならわかりやすく「まあ捨てるよね」と判断できると思いますが、この感覚を大事してください。付箋だろうと、紙だろうと、デジタルで書いたテキストだろうと、行動し終えたメモはゴミですので、さっさと捨てましょう。ただし捨てなくていいシチュエーションであれば放置で構いませんが、高度になるので補足に飛ばします（※1）。

- ※
  - 1 捨てなくてもいいシチュエーションは主に 3 つあると思います。
    - 1 つ目は、デジタルにおいて「後で目にいれる可能性もほとんどない」場合です。たとえばメモ領域を日付ごとに分けている場合、過去の日付のメモは（所定の読み返しタイミングを除けば）ほぼ見ることはないと思います。毎週日曜日に今週分のメモを読み返す運用をしているなら、読み返すのはその週の日曜日だけであり、それ以降はありません。この場合、消す行動はせず放置してもいいでしょう。
    - 2 つ目は、クリエイターの人です。クリエイターは発想が大事であり、発想するためには色んなヒントをいかに目に入れるかが肝心ですが、たまに「一から考え直したい」場合があります。この場合は行動した結果（おそらくよりわかりやすいキーワードや文章を書いている）を見るよりも、それを生み出すもととなったメモを見て再解釈を試みます。再解釈するためには、行動し終えたメモそのものが必要ですので、あえて残しておくのです。とはいえ、そもそも情報の総量が多くて、行動し終えたメモをいちいち読み返す暇や余力など通常はないですから稀でしょう。1 つ目が当てはまる場合で、かつ再解釈したいなと思った場合使う程度でしょう。筆者もメモを多用する人間ですが稀です。
    - 最後、3 つ目は生成 AI です。そもそも行動し終えたメモを消すのは人間の事情にすぎず、人間の処理能力が限られているから対象を節約しよう、そのために要らないものは消そうというだけの話ですが AI には当てはまりません。何百万、何千万、それ以上のテキストを学習することなど朝飯前ですし、すでに NotebookLM や Claude など大量のテキストを食ってそれに基づいて回答してくれるサービスもあります。現時点では PDF の入力のみで、文字数として数十万文字程度ですが、今後はもっと増えると思います。そうなると自分が書いた情報をメモ含めて全部渡して、それら情報に基づいて回答してもらえるようになります。現時点で ChatGPT は一般的な回答しか出せませんが、自分の情報も全部食わせてやれば、自分にパーソナライズされた回答を出せるようになります。自分よりも自分に詳しい AI 秘書のような存在も、もう現実に迫っています――と、これは筆者の想像でしかありませんし、生成 AI サービス側としても私物化されると儲からないので食わせられる量は意図的に抑えていますが、何ともわくわくする話ではあります。一方で、メモは AI に食わせるエサとしてもマズイ（役に立たない）可能性もありますが、そこは将来使えるようになってから判断するとしましょう。

まとめ

- メモとは後で読み返して行動を起こすための一時的な記述
- 原則２つ
  - 完璧主義は捨てる
    - メモは大量に発生しがちなため、すべてを扱うのは不可能
    - とはいえ諦めたり形骸化したりするとメモの恩恵を丸々捨ててしまうことになる
    - メモ自体は扱うが、全部は扱わなくていいという塩梅
  - できるだけ早く書き、しかし後で読み返しても理解できるというバランスを模索する
    - 読み返し時に「解読」が必要な場合は、おそらく情報が足りないのでもっとちゃんと書いた方がいい
- メモのフロー
  - 書く → 読み返す → 行動と処理、のステップから成る
  - 1 書くときは素早さが重要、1秒でも早く書き残せるよう最適化したい
  - 2 読み返す際はタイミングが大事だが、結局は納得感なので、メモを運用するだけの価値があると思える状況を見出すのが良い
  - 3 行動は Act first, Manage second（可能なら行動してしまう、無理そうならタスク管理に回す）が良い。処理、つまりは行動し終えたメモをどうするかは「捨てる」一択

=== Chapter-14 タスクソース ===

タスクソースの概要

概要

タスクソース（Task Source） とはタスクを生み出す何らかのもの――特に言語化されたものを指します。ソース（Source）とは源泉を意味し、まさにタスクの源泉というわけです。

序盤の章で解釈の公理について述べましたが、ある事柄がタスクであるかどうかは人次第です。その人がタスクだと解釈すればタスクですし、タスクではないと思うならタスクではありません。また同じ人であっても、状況次第でタスクだと解釈したりしなかったりするでしょう。

タスクソースもまさに解釈が絡んでいるもので、ソースからいつどんなタスクをつくるか（解釈するか）はその人次第です。引っ越ししたいという「やりたいこと」があるとして、ここからいつどんなタスクをつくるかは人次第ですよね。遠い願望くらいに考えてる人なら当面は何もつくらないでしょうが、極論近所トラブルやストーカー被害など喫緊の理由により今すぐやりたくなるかもしれませ

ん。いずれにせよ、「引っ越ししたい」なるソースがあるからといってタスクも絶対に存在するかというとそうではなく、あくまでタスク化しているのは私達自身の解釈にすぎません。

さて、この「引っ越ししたい」のような事柄をタスクとして扱っても上手くいきません。すでに述べたとおり、粒度が粗すぎて具体的に何をすればいいかがわからないからです。そもそもこんな雑な事柄を扱うだけで上手くいくなら、タスク管理など要りません。かといって、じゃあその場で事細かに具体的なタスクに分解するかというと、仕事でもなければ中々できないでしょうし、仕事でやったとしても上手くいくとは限りません（プロジェクトタスク管理は難しい）。そもそもすでに書いたように、同じ自分であっても状況次第で解釈は変わるのです。

ここでタスクソースの考え方を導入すると比較的やりやすくなります。具体的には、ソースはソースとしてリストアップなり掲示なりしておいて、これらから必要に応じて必要なタスクをその都度つくります。

タスクソースの例

タスクソースの例を示します。

- 目標事項
  - 達成可否が存在する、道しるべとなるもの
  - 例: 目的、目標、大タスクなど粒度の粗いタスク、プロジェクトなどのコンテナ
- 維持事項
  - 恒久的に維持したいこと
  - 例: 日課、習慣、一部のモットー
- 連想事項
  - 何らかの事物を思い出す・思い起こすためのヒントとして使うもの
  - 例: トリガーリスト、ブレストなど発想法で出力したもの全般、何らかの俯瞰全般
- 発想事項
  - ひらめきや思いつきを促すもの、インスピレーションのもととなるもの
  - 例: 人によって著しく異なる

いずれも言語化されたものを想定していることに注意してください。なので具体的な人、場所、物タスク、画像や動画といったものはタスクソースではありません。これらリッチな事物もソースとみなして、タスクを生み出すことはできますが本章では扱いません。

以下、分かりづらい部分をもう少し解説します。

コンテナは目標事項とみなすこともできます。コンテナは通常、すでに存在するタスクや今後発生しうるタスクを束ねる単位であり、ソースからタスクを生み出すというよりも、既存のものを捕まえてくるイメージですが、実は前者として扱うこともできます。仕事においても、仮説や想像に基づいて先手を打っておく・先回りをしておくことは珍しくないと思いますが、これはいわばコンテナをソースとみなして、自分なりにタスクをつくっているわけです。

維持事項は、タスクソースとして最もわかりやすいかもしれません。たとえば「体重を 50kg 以内でキープする」との維持事項があった場合、そのために日々何をするかは別途タスク化するべきです。決意だけでは何も変わりませんし、もちろんタスク管理ツールに「50 kg 以内をキープする」など書いておいてもおそらく何もしません。普通は「毎日 2 km 走る」のような定期的なタスクをつくりつつ、脚を痛めたなら通院するでしょうし、シューズが敗れたなら買い替えるでしょうし、梅雨が近づいたら別のメニューに変えるなりレインウェアを購入するなりとスポットでもタスクを（適切なタイミングで実行できるよう上手く）入れて調整します。

連想事項と発想事項はわかりづらいですが、連想事項の方が身近です。アイデア出しとか発散とも言いますが、連想的に色んな記憶や知識を引っ張りたいことがよくあり、連想事項とはその際に目にいれるものです。たとえば「人生について考えたいときに見るといい 10 のリスト」なるものをつくっておけば、人生について考えたいときにそれを見ることで連想の助けになります。一から発想するよりははるかに楽なことです。またフィクションでも、主要人物の顔写真や相関関係を視覚化してそれを眺めているシーンがよくありますが、これも連想のヒントとして顔写真を使っている（そして並べることで俯瞰している）のです。

発想事項については、連想とは違って、ひらめきや思いつきといった「ゼロから急に何かが浮かぶ」感じを狙うものです。通常、発想はタスクソースの形で狙って出すことはできず、アイデアの 3B（入浴中、乗り物で移動中、寝るとき）ともいいますが、リラックスしてぼーっとしているときに降ってくることが多いです。これを意図的に狙うのが発想事項で、インスピレーションと呼ばれることもあります。発想事項とはインスピレーションを得るための言語化された何かです（※1）。何が発想事項であるかは人によって著しく異なり、詩などの美しい文章の人もいれば、格言やことわざなどの人もいるでしょうし、国語辞典の人もいるかもしれません。筆者の場合は「デジタルツールのヘルプ」や「個人的に気に入っていて、あまり難しいことを言わず、でもタスク管理など仕事術に関することを言う人の投稿」などが発想事項です。体感ですが、強張らず疲れてもなくてリラックスした頭でニュートラルに読めるかどうかが肝心という気がしています。ひらめいたものは、そのままだと見逃すのでタスク化します（その前段としてメモ化することも多い）。

- ※
  - 1 インスピレーションというと芸術的な文脈を浮かべると思いますが、実際そのとおりで、言語的な情報よりも非言語的な情報、何なら体験の方がよくもたらされます。言語情報にとらわれる必要性はありません。ただ、本書ではタスクとは言語化されたものであるとの立場を取っているため、本章におけるタスクソースも（発想事項を含め）言語化されたもの、との定義にしています。

タスクソースの運用

目標事項と維持事項の運用

メモのフローと同じような流れに沿います。タスクソースの定義 → 読み返し → タスク化、です。

まずタスクソースを定義するところは、いきなりですが一番大事な部分です。目標は達成するまで続きますし、維持事項に至ってはともすると一生続きます。長期的に続けても耐えられるほどの何かが必要で、何かとは納得感または信念です。アスリートタイプの人なら「とりあえず半年続けてみるかー」もできますが少数派でしょう。納得感も信念もない場合は、おそらく続かないため、最初から諦めた方がいいでしょう。またひとりでこれらが手に入らない場合は、誰かに従ったりどこかに行ったりしてとりあえず従ってみる、くらいに割り切った方がやりやすかったりします。ただし依存しすぎるとハメられたり抜け出せなくなったりするので一歩引いた姿勢も欲しいところです。次の読み返しも良い防波堤になります。

タスクソースを定義した後は、日々読み返す行動が必要です。タスクソースを見て、現状もかえりみた上で、じゃあ次は何しようかと考える（そしてタスク化する）のです。あるいは支障がない範囲で思い出せるなら、明示的にソースを見るなどという行動は要りません――が、そうかんたんにはいかないので、通常は意図的に読み返す機会を確保することになります。かといって、読み返しすぎても今度は麻痺してしまって形骸化する（確信的にサボることも含む）のが難しいところです。目標事項の場合は、進捗という形で 100% に近づいていくことを可視化できればモチベーションは維持しやすいです。維持事項はそうもいかず、事実上日々の読み返しやタスクを楽しめるかどうかが争点となります。

読み返すだけでは意味がないので、タスク化も行ないます。毎回行う必要はありませんが、毎回行うかどうか判断するのも面倒くさくて形骸化に繋がりがちです。かといって「毎日 2km 走る」のように予定にしてしまっても、これはこれは苦しくなります。どういうバランスだと続くのかは自分で微調整していかないといけません。別の言い方をすれば、微調整という営みは避けられません。微調整を行うことから逃げると十中八九形骸化します。「微調整する努力ができるかどうか」は一種の目安です。できない場合は、タスクソースの定義の部分でも述べたように、納得感や信念が足りないと思います。あるいは微調整という細かい営みを行えるだけの資質がないか、です。

この一連の流れを上手く言語化、体系化しているのがGTDやPARAメソッドです。どちらもタスクソースをリストアップしておくことと、定期的に読み返して必要ならタスク化を行うこと（特にGTDではレビューと呼んでます）を説いています。

もう一つ、納得感や信念を足らせるためには、自分についてよく理解しておく必要もあります。自己理解、自己認識、自己分析など色々な言い方がありますし、自己啓発の文脈でもよく語られますが、大事なことです。自分のことを理解しているからこそ、これは合う、これは合わない、こっちは合いそう、こっちは合わなそうといった判断や行動ができるのです。もはやタスク管理からは離れていますが、タスクを管理するのは私たち自身であり、かつ私たちは人間であって意志や事情や好き嫌いがあります。自己理解は実は非常に重要でして、ここができてないと自己判断もできず、言いなりになるか多忙に身を置いて誤魔化すかしかできなくなってしまいます（このような生き方が悪いと言っているわけではない）。

連想事項の運用

方向性は 3 つあります。

- 1: 連想リストの運用
- 2: 発散と収束の営み
- 3: 俯瞰のつくりこみ

連想リスト

連想リスト（トリガーリスト） とは、何らかの連想を引き出すための自問自答集・ヒント集のことです。チェックリストとも似ていますが、各項目を全部消化しなければならない制約はなく、もっと雑に使ってもいいものです。1 つの項目から 10 個連想してもいいですし、「何も浮かばないからこれはいいや」と無視しても構いません。とにかく連想を引き出すために、テキトーに使えばいいのです。

この連想リストをどれだけ揃えられるかが、日々どれだけ連想の質を上げられるかの分かれ目になります。私たちの人生は、よほど難しい仕事をしているのでなければ、基本的に答えは単純です。ただそれに至ったり、納得感を見出したりするのに苦戦しているだけです。ここをショートカットできるのが連想で、別の言い方をすれば「考え抜く」「思考の隙や死角をなくす」とでも言えましょうか。正解かどうかはわからないし、正しいかもわからないが、自分としては考え抜いたぞ、これ以上はないぞ――と自信を持って言えるようにしたいのです。特に現代は私たちの目も肥え、VUCA で先が読めない時代でもあるからこそ重要だと思います。とはいえ、何も無しに考え抜くことはできないので、連想リストなるヒントを揃えておいて、これベースに膨らませていきましょう、とするのです。

連想リストの例を挙げます。

生活:

- やりたいことリスト
  - いつできるかはわからないが、いつかやりたいことをリストアップしたものです
  - GTD でも「いつかやるリスト」が存在します
  - 項目数が数百、数千以上にのぼることもざらにあります
  - 重複も肥大化も気にせずとにかく追記することと、眺めるときは完璧主義を捨てて肩の力を抜いて眺めることがポイントかと思います
- 人生について考えるときに見るリスト
  - 書籍『28歳からのリアル』で取り上げている以下 8 要素を並べたものです
  - 1. 仕事、(2) 結婚、(3) お金、(4) 住まい、(5) 健康、(6) 親、(7) 趣味、(8) 常識
  - これを見ながらだと「結婚についてはどうしようか」「住まいは……」などと連想がしやすくて、筆者としても重宝しています
- 生活のコツ128

    - 生活のコツを128並べてみる - 山下泰平の趣味の方法
    - 明治時代の「簡易生活」を前提に、生活のコツを128個並べています
    - 合う合わないはあるでしょうが、部分的にでも取り入れてみたいなど考えながら読むと生活の足しになります
        - 筆者は道路で過ごすときの QoL の低さが長年悩みでしたが、043 を見て自分の方が気をつけるしかないと決断できました（そのために普段使うルートや時間帯を設計し直すタスクを追加したりした）
- 自己啓発リスト
    - 自己啓発 - Wikipedia
    - Wiikpedia のページで、自己啓発の主要なテーマがリストアップされています
    - 自分に足りない視点やソフトスキルを探したい、何とかしたいときに見ると連想が捗ります

仕事:

- プロジェクトリスト
    - 自分が抱えている仕事や用事をリストアップしたものです
    - どんな粒度で並べるか、読み返しの頻度はどうするかなど奥が深いですが、これを眺めれば「そろそろアレやっておかないとな」など思い出しやすくなります
- ピープルリストやポジションリスト
    - 自分が普段絡んでいる人や役割の名前をリストアップしたものです
    - 仕事で次何をするべきかとか、組織内で今後どうしていこうか等を考えるときに役立ちます
    - 人によっては図の方が連想しやすいかもしれません（連想リストからは外れますが）
        - プロジェクトではよく「体制図」が用意されていると思います
        - ポイントは自分にとって見やすく連想しやすいように作り直すことです
- オズボーンのチェックリスト
    - アイデアを練るときに使うフレームワークとして有名だと思います
    - 転用（他の用途は？）、応用（他の事例は？）、変更（新しい非言語的側面は？）、拡大（大きくできる？）、縮小（小さくできる？）、代用（別の何かで補える？）、再配置（構成要素の組み合わせを変えたら？）、逆転（逆にしたら？）、結合（いっしょくたにしたら？）

また（連想リストとは限りませんが）色んなリストを集めたコンセプトの書籍やウェブサイトもあるので、探してみるのも良いでしょう。

- Amazon.co.jp: 仕事と自分を変える 「リスト」の魔法 (角川書店単行本) eBook : 堀正岳: Kindleストア
- Awesome List
    - ソフトウェア開発や IT の文脈になりますが、「そのジャンルの定番を集めたリスト」があります
    - 本家: https://github.com/sindresorhus/awesome

- 解説: GitHubのawesomeリストが本当にawesomeなものばかりだから一度見てほしい #GitHub - Qiita
- たとえばリモートワークネタを扱った awesome-remote-job なんてものもあります

ネット上でも色んな人が色んなリストをつくっていることがありますし、最近では ChatGPT にお願いして洗い出してもらうこともできます。

連想リストと付き合うコツ

とにもかくにも、まずは **雑でもいいので使ってみる体験をする** ことです。個人的にはこれができるかどうかがすべてだと思います。上記でプロジェクトリスト、ピープルリストやポジションリストを紹介しましたが、これらを実際に自分なりにやってみるのです。自分なりにプロジェクトや人や役割を洗い出してリスト化して、そのリストを眺めてみて、色々連想してみて、そのうち行動に起こしたいものをタスク化（あるいはすぐ行動できるならしてもいいですが）して――と、実体験することで連結リストが身近になります。特にリストは、所詮はただの箇条書きですので、わかった気にもなりやすいし、スルーや形骸化はもっとよく起こります。このハードルを越えるには、実際に手と頭を動かして、連想まで体験して「意外と悪くなかった」「楽しかった」「すぐに何とかなるわけではないが何とかなりそうと思えた」といった実感を得ることが最重要なのです。

次に、連想リストはできるだけ自分でカスタマイズ、メンテナンスしていきたいところです。思っているよりも合う・合わないが激しいので、自分に合うようリストの中身を微調整していくのが望ましいです（目標事項や維持事項ほどではない）。逆を言えば、微調整しなくても使えるならそれでも構いませんが、リストの活用はただでさえ形骸化しやすいので、違和感は少しでも取り除いていくことをおすすめします。

最後に、わからない単語をちゃんと調べることもおすすめしたいです。連想リストの項目内あるいは連想した事柄の中に「わからない単語」が混ざることがあります。わからないまま扱っていると、目に入る度に無視をする、という余計なノイズが発生します。これが積もるとバカになりませんし、筆者は **連想に頼るのが苦手な人の大半はこのノイズに潰されているからではないか** とさえ思います。ノイズは無視せず、ちゃんと一つずつ調べていくのが良いと思います。といっても完璧主義的に把握するのではなくて、「なんとなくわかったからこれでいいや」とか「なんか面倒くさそうだから調べるやーめた」でも構いません。自分でそう判断したという実績が大事です。実績があればノイズはスルーしやすいです。

発散と収束

方向性の２つ目は **発散と収束** です。

発散とは、アイデアや情報を何でも出すことです。目に見える形で残す必要があります。テキストで箇条書きで書いてもいいですし、文章で思いついたことを長々と書いてもいい（フリーライティング）ですし、付箋を書いていっても構いません。マインドマップなど脳内のイメージを反映するような視覚性に寄った手法もあります。この時点では解釈や判断は行わず、とにかく量を重

視します。

収束とは、発散したものをうまくまとめることです。発散されたものを律儀に使って上手くまとめあげる要約と、発散されたものもヒントでしかないと捉える **蒸留** があります。要約は論理性やストーリーを重視し、蒸留は内容の面白さや本質の捕捉を重視します。蒸留の概念はわかりづらいかもしれませんが、発散されたものはすべてヒントでしかなく、使うも使わないも自由ですし、勝手に解釈を加えて改変するのも自由です。

発散と収束は一般的に行き来します。発散したものを部分的に収束して、思いついたものはまた書き足して、どこかのタイミングで一度おおがかりな収束をして、まだ落ち着かないからもう一度発散して――のようなことも当たり前に行ないます。ブレインストーミングや KJ 法など発想法と呼ばれる手法もいくつか存在しますが、本質は発散と収束にすぎません。自分なりに発散と収束ができるのなら、方法は問いません。

さて、タスクソースの話に戻りますが、先ほどの連想リストが静的な連想事項であるなら、**発散と収束は「動的な連想事項」である** と言えます。タスクを生み出すかもしれないヒントを、発散と収束を通して生み出していくのです。発散にせよ、収束（特に蒸留）にせよ、連想を多用します。そうして連想したことを発散として **実際に書いて**、それを見てさらに連想を引き出して、また書いて、整理して、ときには配置を変えたり削除したり束ねたりして（収束の結果もちゃんと反映して）――と書きながら進めていきます。

最終的には要約にせよ、蒸留にせよ、何らかのまとめが出ます（というよりたいていきりがないのでどこかで打ち切ります）。必要ならタスクも言語化します。動的な連想を繰り返してきたからこそ、これでよい、これ以上はたぶん無い、これ以上は仕方がないと自信を持って言える状態になります。

筆者個人はこの営みを非常によく使います。普段の仕事でも「さっき定例会議が終わったけど今週は何しようか」と発散と収束を少しやってみて、その結果「うん、この 3 つをやろう」とか「温めたいアイデアが一つあるからこれを詰めていくためのタスク化もしておこう」とか「打ち合わせ頻度増えるから今のうちに予定押さえておくか」とタスクを追加したり、といった行動が起こります。悩んでるときはよく適当に書き散らして（発散して）、軽くまとめたみたりもして（収束）、その結果「ああ、結局給料少なくて悩んでるんだな」「でもそのために過剰に努力する必要がないんだよな」「じゃあ現状維持しかなくない？」「強いて言えば出費を抑えるくらいか」「ミニマリズムの本を買ってみるか」などとやはり行動に繋がったりします。繋がらないこともありますが、ひとまず出し切ったとの解放感や達成感があるので気持ちは楽になります。

と、見るからに手間暇がかかる営みではありますが、自分を出し切るがゆえに納得感が得られやすいのがメリットです。

発散と収束のコツ

発散と収束は **自分ひとりで集中して行える環境を確保できるかどうか**、そしてそのような **孤独な営みにそもそも耐えられるか** がすべてです。発想法のやり方や使うツールなどテクニカ

ルなコツもあります（※1）が、そんなものは勉強するなり試行錯誤するなりすればどうとでもなりますし、原始的には付箋と紙とペンでもできます。

発散と収束は連想に全面的に頼るものであり、連想とは非常に個人的な活動です。なぜなら連想されてくる事柄には個人的な内容を含むことがあるからです。人目があると、個人的な内容を見せるかどうかの判断に認知の大部分を使ってしまうため集中できません。誰にも邪魔されず、100% 自分のやりたいようにできるのが理想です。人のアイデアやコメントや情報はもちろん使いますが、使うときはひとりでなくてはなりません。

ともすると**複数人で行いがちですが、実は典型的な悪手です** 。よほど実力が拮抗しており、信頼関係もあって、かつ脳の性能が優れている（脳内である程度発散と収束ができている）少人数のチームであれば構いませんが、そうでなければただの発散と収束ごっこになってしまいます。自分の力など 20% も出せていないでしょうし、発想とは名ばかりで、実際は場の空気感や権力者の言い分に従って断片的に恐る恐る意見を出すだけの会になってしまうことがよくあります。ともすると、そうなっていることにすら気づけません。複数人で行うのは、ひとりひとりが発散と収束を行った後に、その結果を持ち寄るときに限ります。

わかりやすいのが作家でしょう。マンガにせよ、小説にせよ、あるいはビジネス書でも何でもいいですが、本というレベルのコンテンツは基本的にひとりのライターが書いて、書いた後に編集者などにレビューしてもらうはずです。書く段階から複数人で一緒に書いていこうか、とはしませんし、それだとおそらく上手くいかないことは比較的容易にイメージできるのではないでしょうか。発想と収束も同じようなものです。誤解を恐れず言えば、クリエイティブな営みです。

そういうわけで、発散と収束はひとりで行うのが必須ですが、これは言い換えると孤独に耐えねばならないとも言えます。たとえば 1 時間の間、誰にも見せず、誰とも喋らずに、ただただひとりで目の前の発散物と収束物と向き合うことができるでしょうか。ここは思っている以上に特性に左右されます。できる人はできますし、できない人はやろうともしません。よく「できる」という人がいますが、実は「長時間ひとりで作業することはできる」というだけで、発散と収束はできなかったりします。発散と収束は、1 日数時間くらいしか続けられないほど疲れる（※2）ものでもあり、単に時間さえかければできてしまう「作業」とは全く異なるあり方です。思っている以上に向き不向きがあります。

一つの目安は「頭の中でだけではとても処理しきれない複雑で大量な事柄を」「第三者にも伝わるよう」表現しきれるかどうか、でしょう。10 万文字以上の本でも 30 分以上のプレゼンや講義でも構いません（※3）が、頭という限界を越えられるかどうかがポイントです。表現「しきれる」と書いたのも意図的であり、それなりの分量を秩序を持たせて扱うことも想定しています。付け焼き刃やその場のノリでは到底できないことです。越えるためには発散と収束が必要であり、その発散と収束にはひとりで集中する営みが必要です。表現しきれる人は、できていると言えるでしょう（※4）。

- ※
  - 1 このテクニックやツールの部分も実は重厚で、これだけで本を何冊も書けてしまうくらいですので本書では割愛します。

- 2 疲れる理由は二つあると思います。一つは考えをひねり出すという行為そのものが非常に消耗しやすいことで、アイデア出しにより体験しやすいです。なぜ消耗しやすいかの理屈はわかりませんが、筆者としては「正解がない」かつ「大量の判断を行っている」が両方重なるからだと思っています。もう一つは、自分が向き合いたくない記憶とも向き合う必要があることです。残念なことに、そういうネガティブな記憶に限って色々と連想しやすかったりします。この醜さと向き合う苦しみに耐える（鈍感で耐えるまでもないのがベストですが才能と言えるでしょう）必要もあるのです。
- 3 発散と収束で行えることは本や講演に限りません。むしろもっと汎用的で、戦略や戦術、もっと言えば選択肢を死角なく考え尽くせるようになる、とも言えます。このあたりの資質はストレングス・ファインダーにおける「戦略性」だと筆者は考えます。加えて、孤独に耐えるには「内省」資質も要りそうです。筆者としては、発散と収束の適性はストレングスの資質で説明できる気がするんですよね。調べてみたいところです。
- 4 頭の限界を越えることなく本を書けたり喋ったりできる人もいます。そのような化け物レベルの性能を持つ人がプロと呼ばれる存在になるのだと筆者は思います。実際それほどの性能があるからこそ、食っていけるほど素早く動ける――特にそれなりに短い締切だったり複数の仕事を並行するマルチタスクだったりなど厳しい制約下でも結果を出していくことができるのです。逆を言えば、化け物でなくても、発散と収束を使えば（時間はかかるがそれなりには）できるようになりますし、筆者による本書はまさにその一例です。

俯瞰のつくりこみ

方向性の 3 つ目は、俯瞰のつくりこみです。

俯瞰（**Overview**）とは、全体または特定の範囲を眺めることです。通常は眺められる状態になっていないので、意識的につくりこむ必要があります。たとえば仕事で、ある案件を担当していて、長い目で見てもうちょっと上手くやりたいなーと思っていて、登場人物ベースで考えてみたい場合、登場人物を俯瞰できると良いですよね。しかしそのままでは俯瞰はできないので、登場人物の名前をリストアップしてみたり、顔社員つきで並べてみたりするのです。このように俯瞰時に使うものを **俯瞰物** と呼びます。この場合、俯瞰物として登場人物名のリストや顔写真の列挙をつくっています。

俯瞰を行うことで色々連想を引き出せます。頭の中で俯瞰できれば良いですが、たいていは難しいので、わざわざ俯瞰物をつくるわけです。

俯瞰のコツ

まず俯瞰そのもののコツはありません。俯瞰物を見て、自分なりに連想したことを捉えるだけです。連想をさらに発展させたい場合は、発散と収束を行うのが良いでしょう。

肝心なのはそれよりも俯瞰の前、俯瞰物をつくる段階で、俯瞰物をつくるコツは **自分が使いやすいようにつくる** ことです。特に通常は何らかの俯瞰物が用意されていることが多いです。た

いていのプロジェクトには概要資料があったり、そこに体制図があったりしますし、タスク管理ツールでも様々なビューが用意されています。ともすると、それらをそのまま使いがちですが、アンチパターンです。連想は、少しでも自分にとって馴染みやすく違和感のない場所や見え方の方が捗ります。要は、自分にとっての理想的な俯瞰物をつくれるのは自分だけなので、自分でつくりましょうというわけです。

もちろん、自分にとっての理想なる正解は、最初は自分でもわからないので、しばらくは試行錯誤が続きます。といっても、まずは雑に俯瞰物をつくっていって、それから微修正していけばいいだけです。手段としてもアナログ・デジタルは問いません。自分のやりやすい方で結構です。一般論を言えば、空間的に配置できた方が融通が利くので、空間配置に長けた手段が良いでしょう。アナログだとやりやすいですが、デジタルに慣れているならデジタルでも構いません。あるいは筆者もそうですが、逆に空間だと馴染まない場合は、テキストによる列挙や箇条書きベースで俯瞰物をつくるのが良いかもしれません――とにかく、最適なやり方は本当に人それぞれです。

俯瞰物をつくる営みは面倒くさいですが、ここを越えないとスムーズな連想は引き出せません。この面倒くささに慣れてしまった方が早いと筆者は思います。もちろん、こんなことせずとも、既存の俯瞰物だけで連想が上手くいくならそれで結構です。ただ、それだと思うようにいかない場合もあるので、そんなときに自分でつくれるようになっておくと便利なのです。

発想事項の運用

発想事項の運用とは、「言語的につくられたインスピーション元」をいかにして見つけるか、またどのように付き合うか、との話になりますが、ここは非常に個人的かつ再現性もないため解説は割愛します。

締切との向き合い方

仕事にはしばしば締切が発生します。タスクソースの現実的な運用として、締切とどう向き合っていくかを深堀りしていきます。

タスクソースは締切と相性は悪い

まずタスクソース自体は締切――特にタイトな締切が存在する場面とは相性が悪いです。

タスクソースは、自分の裁量でタスクをつくるための源泉です。冒頭で解釈の公理についておさらいしましたが、率直に言えば、気分でタスク化しているにすぎません。当然ながら締切が存在する場合、気分でタスク化しているだけだと間に合わない可能性があります。アスリートタイプの人なら「間に合うようにタスク化する」こともできますが、万人向けではありません。

かといって最初からガッチガチにタスクを洗い出して管理するのも、私たち人間は怠け者なので難しいでしょうし、仮に仕事などでそうせざるをえない場合でも（単純な作業に帰着される世界でもなければ）正確な見積もりなどできませんから十中八九外れます。それを認めたくなくて残業でカバーしますよね。ひどい場合はサービス残業の形で正当な報酬がもらえないすらあります。外れるのに疲弊も加わっているので最悪です。プロジェクトだの案件だのと呼ばれる仕事は過酷だったり、最低でもワークライフバランスが一時的には崩れがちですが、それは計画という「そもそもそのとおりに踏襲することが現実的に難しい」ものををバカ真面目に行っていることと、計画ベースの管理に従い続けることによる疲弊とのダブルコンボが重なっているからです。

ソースアプローチとプランアプローチ

ここまでを整理します。仕事のやり方は二通りあるということです。

- ソースアプローチ
  - タスクソースを運用すること
  - タスクソースから適宜タスク化を行う
  - 主観に基づいて行えるが、締切と相性が悪い
  - いうなれば「探索的」
- プランアプローチ
  - 計画やスケジュールといったレベルで想定をつくり、そのとおりに動くこと
  - 基本的に仕事はこの価値観で行うことが疑われていない
  - 理に適っているように見えるが、どうせ計画は上手くいかない + カバーで残業するなど疲弊が増えやすいの二重苦で過酷になりやすい
  - いうなれば「計画的」

ソースとプランの臨界点

ここまでを踏まえると、両者の使い分けができれば良さそうに思えてきます。

状況にもよりますが、最も汎用的なのは **最初はソースアプローチを行ない、どこかでプランアプローチに切り替える** ことです。プランアプローチばかりだと息切れしますし、ソースアプローチばかりだと締切に間に合わないので、両取りします。まずはソースアプローチにて楽に進めていき、ある程度見えてきたらプランをつくってガッと走り切るイメージです。別の言い方をすると、最初のうちは探索的に動いて、その後で計画的に動きましょうとなります。

現代の仕事はプランアプローチの考え方に支配されていますが、これは管理上都合が良いからです。計画をつくると数字化ができるため金銭（予算など）との対応も取りやすいですし、数字いじりしかできない無能な管理職達にも数字いじりの仕事を渡すことができます。数字いじりのあり方を複雑にして、何ならルールとしてしまえば、従業員を数字で管理してこき使うこともできますし、逆に「仕事のための仕事」を創出して余計なお金を取ったり中抜きしたりもできます。

方法として優れているからではありません（この数字や管理こそが肝となる仕事や状況ももちろんあります）。

ソースアプローチを使うと、このプラン一辺倒なやり方から脱せます。まずはプランをつくらず、タスクソースからタスク化する、という形でマイペースにこなしていきます。そうすれば、マイペースではありますが、こなしてはいるので色々見えてきます。マイペースな分、疲弊も小さいので、より本質が見えたり、率直な議論ができたりすることも多いです。そうして色々見えてきたら、そこではじめてプランアプローチに切り替えて、あとは頑張って走り抜けます――このやり方なら、プランアプローチだけで疲弊することなく仕事をこなせます。

このとき重要なのが、どこでソースからプランに切り替えるかというタイミングです。この切り替えのベストタイミングを **SP臨界点（ソースとプランの臨界点）** と呼ばせてください。SP 臨界点の見極めが重要だということです。見極めのコツですが、何とも抽象的な言い方になりますが「見えてきたかどうか」です。ソースアプローチであれこれこなしていくうちに、情報や学びも増えてきて、どこかで「大体わかってきた」「あとはいけそう」と感じるポイントがあります。ここが SP 臨界点です。この状態になったら、ようやくプランをつくれます。少なくとも SP 臨界点の前、何もわからない状態からプランをつくるよりは、はるかに現実的かつ無理のないものになります。

ちなみに SP 臨界点が最初から見えている場合があります。後述もしますが単純な世界（たとえば作業に帰着できる仕事）であるケース、あるいはプロなど著しく実力があってすぐに臨界点が見えてしまっているケースなどです。この場合、あとは終わるまで作業できるかどうかが争点ですので、プランアプローチでいけます。ただ、そうは言っても、人間であり忘迷怠は生じるので、プランには余裕を持たせたい（持たせるための交渉や日頃の盤外戦も含む）ところです。このあたりはタスク管理の技術であり、プロでもできるとは限らないし、そもそもプロになれるほど生産性と行動力の高い人にタスク管理術を身につける余裕（そして適性）はないですから、基本的にはできません。だから慢性的に忙しい人が多いですし、秘書やマネージャーといった形で委譲する形態も言わずと知られています。

その他の戦略

ソースからプランに切り替える戦略を述べましたが、他にもあります。

- 1: ソースアプローチのみ使う
- 2: プランアプローチのみ使う
- 3: ソースアプローチ → プランアプローチ
- 4: プランアプローチ → ソースアプローチ

ここまで紹介してきた、SP 臨界点が要求される戦略が 3 です。残る 1, 2, 4 についてもかんたんに取り上げておきます。

1 については、ソースアプローチのみを使い続けるというものです。締切のあるタスクに対してはおすすめしません。この戦略は、締切がなく半永久的に続く（続けることができる）事柄に使い

ます。維持事項が典型例ですが、マイペースであっても続けることが大事なので、自分なりにタスク化すればそれでいいのです。ただし、ソースアプローチだけでは一向に進展がない場合は、プランアプローチを入れてテコ入れする必要があります。あるいは「これ以上成長しなくてもいい」と開き直るのもアリです（特に趣味の場合）。

2 については、プランアプローチのみ使い続けるもので、特にビジネスでは現状最も自然な戦略です。最初から計画やスケジュールを組まされた経験のある読者の人も少なくないでしょう。基本的に現代の VUCA な状況では悪手だと筆者は考えますが、管理する側がこれしかやり方を知らないケースも多く、基本的にはこれになりがちです。一方で、これが有効なシチュエーションはまだまだ数多くあります。建築や製造など単純な世界（現実の物理的制約のみを相手にすればよく、ソフトウェアなど電子的な世界よりははるかに単純でプランアプローチが通用しやすいです）はそうですし、規模の大きなプロジェクトや組織では、そもそも秩序を持たせるために数字による管理が幅を利かせるようになるため、プランアプローチ一択になりがちです。

4 については、あまりありませんが、プロジェクトの中止が該当します。もう行う必要はなくなったが、ここまでやってきたものがあるので、残りをどうしようかと考えて、その分をソースアプローチで行っていきます。中止命令が厳しい場合は、勝手に行うと怒られたり処分されたりする可能性もありますが、中止となってはいそうですかと全部捨てるのはもったいないので、自分なりに足掻くわけですね。個人的にはノウハウ化や振り返りなど、自分の成長につながる活動を（許される範囲で）しておくと良いかなと思います。

まとめ

- タスクソースとはタスクの源泉であり、必要に応じてタスクをつくりだす（タスク化する）もの
  - いつ、どんなタスクをどれだけつくるかはその人次第で、正解はない
- タスクソースとして以下が存在する
  - 目標事項
  - 維持事項
  - 連想事項
  - 発想事項
- 特にビジネスでありがちな「締切を持つ事柄」にも適用できる
  - プランアプローチはよく知られているが、タスクソースを用いたソースアプローチも可能
  - 先にソースアプローチを行ない、見えてきてからプランアプローチに移るのがベター

余談ですが、本章で一部出てきた「探索的なあり方」については、後の章で詳しく提案します（探索的タスク管理）。

=== Chapter-15 □第4部「個人タスク

管理 応用」===

個人タスク管理に関する高度なトピックスを扱います。タスク管理と聞いてすぐに思い浮かべるものではないかもしれませんが、これらを知って使いこなせるようになると、タスク管理の幅が広がります――自分にとって最適なバランスを模索しやすくなります。

=== Chapter-16 動線とリマインド ===

タスク管理でも最終的には行動が必要ですが、その行動を担うのは私たち人間です。一方で、特に現代人は忙しく、疲れやすいため、適切なタイミングで行動できる（行動するために思い出すことも含む）とは限りません。一般的にはカレンダーで予定をつくって、それを眺めて忘れないように祈りますが、それでも忘れてしまうことがあります。そうでなくとも目覚まし時計――アラームの概念が古典的によく使われているように、人間の性能そのものがポンコツなところもあります。私たちは自力で毎日決まった時間に起きることすらままならないほどポンコツなのです、がポンコツなままで許される人は稀でしょう。

よって必要に応じて行動できる、特に行動できるよう思い出せる（ようになる）ことは非常に重要です。これを担うのが動線とリマインドです。本章では動線とリマインドについて取り上げます。実は、単に行動できるようになりたいなら「環境や人の力に任せる」ことでも可能ですが、本章では取り上げません。本章ではあくまでも環境や人に頼らず、自分のみで、動線とリマインドに頼ることで行動できるようにすることを目指します。

動線

動線の概要

個人タスク管理の章でも少し取り上げました が、改めて書くと、**動線** とは毎日頻繁に通る場所のことです。トイレのドアやオフィスの机など物理的な場所には限定せず、PCのデスクトップ画面、スマホのホーム画面、仕事で毎日使っているチャットツールなども動線の一種と言えます。

何が動線であるかは人によって違います。また、同じ人であっても動線は変わることがあります。極端な話、引っ越しをしてしまうと物理的な動線は何から何まで変わりますよね。同様に新しいツールを採用したり、同じツールでも使い方を変えた場合にも動線が変わることがあります。

動線は忘迷怠を減らす

動線を使うと忘迷怠を減らせます。動線は頻繁に通る場所ですから、そこにタスクを置いておけば頻繁に目にするはずです。目にすれば行動できる可能性も高くなるはずです。たとえば前日寝る前にゴミ袋を玄関ドアに置いておけば、翌日出社で外に出るときに忘れずごみ捨てを行えると思います。

と、先ほどから「はず」とか「思います」などとぼかして書いてますが、そのとおり、絶対ではありません。**あくまでも目に入る機会を増やすだけです**。目に入ったからといって忘れないわけではないし、まして迷いや怠けをなくせるわけでもありませんし、それでなくせるなら苦労はしません。それでも、目に入ることさえなければほぼ 100% 忘れたままです（何もせず自然に思い出せるならそもそもタスク管理は不要）。動線上にタスクを置いて目に入る機会を増やす――行動できる確率を上げるのは、極めて現実的な営みなのです。

動線にタスクを置く

動線にタスクを置く、とはどういうことでしょうか。

まず例を挙げましょう。現在、平日の昼で、在宅勤務で自宅にいるとします。終業後に買い物に行きたいので「買い物」と書いた紙に書いたとします。このままだと終業後に思い出せるとは限らないので、この紙を動線に置きましょう。置き場所は多数考えられます。

- 1: ドアに貼る
  - 冷蔵庫、トイレ、玄関
- 2: 紙を丸めて靴に入れておく
- 3: 私有のカレンダーの、終業後の予定として「買い物」を入れておく
- 4: 普段から多用している X に「今日は買い物する」とポストしておく
- 5: 自分で自分に「買い物」とメールしておく
- 6: 会社PCの収納先の棚の中に、紙を入れておく

以下、順に見ていきましょう。同時に重要な概念も取り上げていきます。

ネグレクテッド

- 1: ドアに貼る
  - 冷蔵庫、トイレ、玄関

1: のドアに貼る案は、手軽ですが形骸化も容易です。冷蔵庫、トイレ、玄関はいずれも終業後より前に一度は通るでしょう。そのときに目にして「うん、終業後は買い物だよな」とそこで認識するのはいいですが、終業後にこれを思い出せるとは限りません。かといって、終業後にもう一度ドアを都合よく見るとも限りません。

加えて、仮に見たとしてもスルーしてしまう可能性がおそらく高いです。というのも、一度見て認識した項目を、あとでまた見ても認識しなくなるからです。「やっぱいいや」と確信的にスルーするのではなく、そのような判断すら介在せず、あえて言うなら無意識・無自覚レベルで読み飛ばしてしまいます。これを **ネグレクテッド（Neglected）** といい、個人タスク管理では非常によく遭遇する現象です。いつネグレクテッドになるかは人次第、状況次第ですが、筆者のようにストイックにタスク管理をまわせる人でも起こります。

動線による忘迷怠の軽減は、ネグレクテッドとの戦いでもあります。ネグレクテッドが起きないように、動線のどこに何を置くかを設計しなければならないのです。これを **動線のデザイン** といいます。動線のデザインは、あなた自身の仕事です。将来的に動線デザイナーのような役割も登場するかもしれませんが、現時点では先進的すぎて（あとマネタイズが難しくて）まだ出ないでしょう。自分で設計することからは避けられませんし、ここが嫌なら動線による忘迷怠軽減は諦めてください。動線のデザインについて、また後で取り上げます。

アクティブリマインド

- 2: 紙を丸めて靴に入れておく

この 2: は、特定の行動（この場合は靴を履く）時に発動するよう仕込んだとも言えます。実はリマインダーの一種です。リマインダーというと指定時間にアラームが鳴るよう仕込みますが、一般化すると「何かが起きたら知らせる」わけで、その何かとして「靴を履いたら」を使っているわけです。ただ、靴を履いたら、を仕組みとして直接検出することはできないので、靴に何かを入れておけば履いた時に気づけて事実上検出できるよね、としています。

このやり方のメリットは、まさにリマインダーの効果もそうですが **（向こうから）強く訴えてくる** ことです。目覚まし時計のアラームは一番わかりやすいですが、めざわりな音で聴覚的にうったえてきますよね。この 2: の案も、足裏や足指先の触覚にうったえるという意味では、それなりに強い訴え方と言えます。少なくとも文字を目に入れて自分の意志で読むよりはずっと強いです。強い分、行動しようという気にもなります。はっとするわけです。

このように仕組みの方から訴えてくるリマインド手法を **アクティブリマインド** といいます。デジタルツールを多用する人はポップアップメッセージや通知ダイアログなど視覚で訴えるものが馴染み深いでしょう。そうでなくとも目覚まし時計アラームなど聴覚的なものは誰もが知っているはずです。スマホだと音だけでなく振動（バイブレーション）で伝えることもできますが、これはポケットに入れているときは触覚的に伝えるとも言えます。靴に紙を入れておくのもこの触覚のカテゴリです（※1）。

- ※
  - 1 五感として残りは嗅覚（匂い）と味覚（味）がありますが、これらを用いたリマインドはまだ無いと思います。そのうち開発されるかもしれませんが、とっさに匂いを出したり、何かを口に含んだりは難しいと思うので、あるにしてもおそらく遅効的――少しずつ滲み出させて気付かせていくタイプのリマインドになるとは思います。

トラップリマインド

- 3: 私有のカレンダーの、終業後の予定として「買い物」を入れておく
- 4: 普段から多用している X に「今日は買い物する」とポストしておく
- 5: 自分で自分に「買い物」とメールしておく
- 6: 会社 PC の収納先の棚の中に、紙を入れておく

3: のやり方は、「カレンダー」という実行確度の高い――そのとおりに行動する可能性の高いツールに仕込んでいると言えます。これを **トラップリマインド** と言います。未来の自分にやらせるために、前もって仕込んでおくニュアンスです、と言うと小難しく聞こえますが、予定の設定は誰もが使っているかと思います。それを使って、予定としてタスクを入れておくわけです。

やり方はカレンダーに限りません。たとえば毎日何度も見る TODO リストを持っているとして、毎日必ず項目を全部つぶしているような生活習慣が確立されているとしたら、このリストに一行書き足すだけで十分でしょう。もちろん先送りの癖がついている場合はネグレクテッドが発生しやすいため注意が必要です。

4: のポストや 5: のセルフメールについては、人によってはトラップリマインドになりえます。X を頻繁に使っており、誤字脱字をチェックしたくて自分のポストを何度も見返す人なら 4: でも買い物を思い出せると思います。あるいはそうでなくとも、X をひとり用の鍵アカウントで使っていて、日記帳 TODO リストのように使っている場合でも、おそらく思い出せる可能性が高いでしょう。メールについても、終業後にプライベートのメールをチェックする習慣があるとしたら、そのタイミングで気付けるはずです。

6: の会社 PC 収納先も同様で、仮に毎日在宅勤務を終えるときに、会社 PC 一式を専用の棚に収納する習慣があるとしたら、その棚は「終業時にアクセスする動線」となっているわけですから、ここに紙を置いておくことで「終業時に目にする」をデザインできます。逆にこのような習慣がないか、定着していない場合は意味がありません。

いずれにせよ、トラップリマインドとは実行確度の高い部分に仕込むことを意味します。ここを見誤るとタスクを実行すべきタイミング（この場合は終業後）までに認識できなかったり、あるいはそれ以前に何度も認識してしまってネグレクテッドが起きたりします。

ちなみに筆者がやるとしたら、（3〜6のいずれでもありませんが）毎日何度も見ている TODO リストがあるので、ここに書き足すでしょう。あるいは 6: も使えそうです。筆者は在宅勤務終了時に、メリハリをつける意味でも会社 PC 一式を必ず別の部屋に収納しているので、ここに仕込むと確度は高いです。具体的には、PC 一式をバッグに入れて吊るしているので、フックに紙をぶっ刺しておく、とかですかね。

動線のデザイン

動線にタスクを置いて忘迷怠を軽減するには、動線のデザイン――どの動線に何を置いておく

かをよく考えて置くことが重要です。

まず不便なので一つ用語をつくります。動線にタスクを置くことで忘迷怠を減らすことを **動線リマインド** と呼ばせてください。動線にタスクを置くことで、あとで思い出せるようにする（リマインドさせる）ということです。さて、動線のデザインとは動線リマインドをどう仕込むか、とも言えます。

ネグレクテッドもロストも回避する

前提として動線リマインドの性質――実行タイミングというものが存在します。ここまでの例では「終業後に買い物したい」であり、「終業後」というタイミングがありました。

さて、リマインドの頻度ですが、このタイミングを自然に満たしたいです。多すぎると形骸化します（ **ネグレクテッド** ）。逆に少なすぎるとタイミングを逃してしまいます（**ロスト** と呼びます）。

つまりネグレクテッドが起きないよう少なめに、しかしロストしないように早く、とのバランスが欲しいのです。このバランスを担保できるように、動線のどこに何を設置するかを考える必要があります。

ハブとジョイントを使い分ける

次に押さえたいのが動線の種類です。ハブとジョイントの二つがあります。

- **ハブ(Hub)**
  - 「拠点」の意
  - ホームポジションと言い換えても良い
  - 一日のうち、ここを起点にして行動する、というような場所のこと
  - 何か行動してはここに戻ってきて、また行動しては戻ってきて、を繰り返す
  - 何らかのツール、ファイル、リストなど「普段から使っていて」「自分でも書き込める」場所がハブに相当する
    - ≒何らかの画面、または記入領域になる
- **ジョイント(Joint)**
  - 「関節」の意
  - よく通る通過点のこと
  - ハブ以外の動線はすべてジョイントと言える
  - 玄関のドアはここに該当する
  - 毎日複数回チェックするが、自分では書き込めない場所（情報収集全般やコミュニケーションツール全般）もここに該当

まずはハブにまとめ、オプションでジョイントを

動線リマインドは基本的にハブが望ましいです。ハブは言わば、自分にとって手慣れた作業場であり、ここにタスクをささっと書いておけば、あとで思い出せるはず――との安心感と安定感

があるからです。もちろんハブ次第では安定感が出ないこともあります。筆者はTODOリストでも安定的に忘迷怠なくこなせますが、それだと上手くいかずカレンダーに頼らないといけない人もいるでしょうし、カレンダーでもダメだという人もいるでしょう。

ハブがない人はハブをつくるところからです。これは（ツールを用いた）習慣をつくることにも等しいので、それなりに難しいです。カレンダーを日々使っている人はそこがハブになるでしょう。戦略の章で言えば、デイリーエリアも比較的採用しやすいと思います。他にも戦略はあるので参考にしてみてください。また次章では習慣を扱っており、こちらも参考になるかと思います。

ハブがあるからといって、タスクを置けば済むかというとそうとも限りません。すでに述べたとおり実行に適切なタイミングというものがあり、ここを尊重できないとネグレクテッド（リマインドしすぎて形骸化）かロスト（リマインドが遅すぎて行動できず）が起きます。こればっかりは、実際に置いてみて試してみるしかありません。目安は **よほどのビジーやイレギュラーがない限り、置けばほぼ100% 忘れることなく処理できる** です。これができない場合、あなたのハブは動線リマインドには向いていません。あるいは、あなた自身が動線リマインドに向いてない可能性もあります。

とはいえ最低限忘れることを防げればOKです。忘迷怠のうち、迷うことと怠けることはよくありますし、筆者もちゃんと認識した上で「今日はだるいし買い物しなくていいか」と怠けることもよくあります。それはそれでいいのです。**単に忘れてしまっているのと、忘れず思い出すことができた上でサボるのとでは天地ほどの違いがあります**。前者はただ忘れているだけですが、後者は意思決定です（意思決定も処理の一種であり、意思決定の結果として先送りすることも立派な処理です）。

次にジョイントですが、**ハブで上手く運用できない場合や、ハブで運用するのが面倒くさい場合にジョイントを使います**。筆者の場合、ごみ捨てを忘れないようにゴミ袋を玄関に置いておいたり、直近買いたい日用品を玄関の棚に付箋で貼り付けたりしていますが、これは玄関というジョイントにタスクを置いている例です。私は習慣として毎朝ごみ捨ての時間帯の前に必ず一度は散歩か運動をしますし、買い物時には棚の付箋をチェックするよう染み付いてもいるので、これで忘れず処理できます。普段使っているハブ（TOODリスト）に記載する必要はありません。

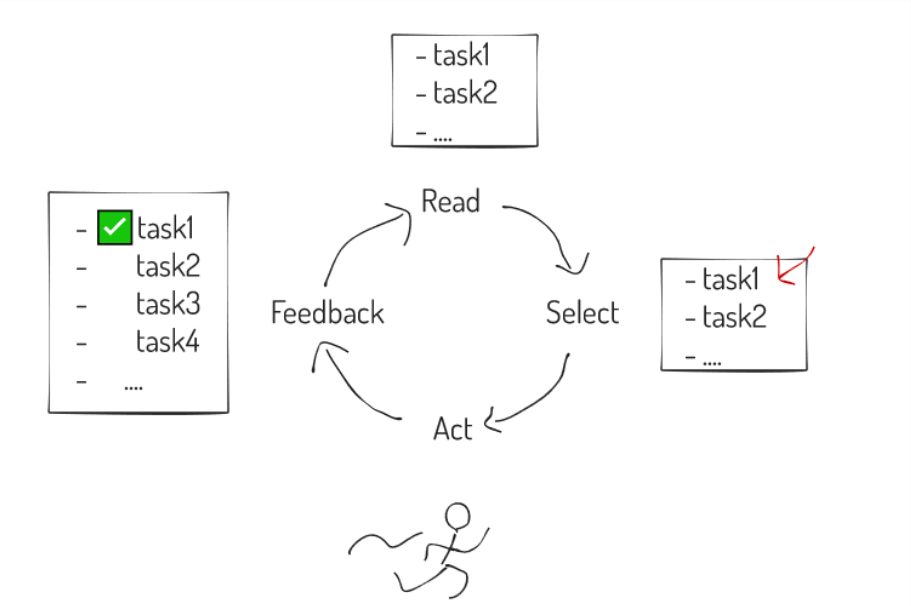
しかし一般論を言えば、ジョイントにタスクを置くのは面倒くさいし、定着する・しないも激しいので、できればハブに集約したいのです。そういう意味でハブが望ましいと書きました。個人差もありますので、ジョイントに置くだけで済むならそれでも構いません。

ちなみに、個人差は本当にあるようで、知人は「玄関前にゴミ袋を置いていたが、またいでスルーしてしまったから信用してない」のようなことを言っていました。そういうケースもあるんだなぁと思ったものです。逆に筆者は身体感覚や空間認識には敏感かつ機敏な方で、物理的に置いておけばまず気付きますし、気づけば処理しますが、一方で文字の読解はいいかげんなもので、ネグレクテッドは非常によく起きます。特にリモートワークになってから毎日出社時と退勤時に報告する必要があるのですが、出社報告をすっぽかしたことが何度もあります。ちゃんとTODOリストに書いているにもかかわらず、ネグレクテッドが発生して、なぜかすっぽかしてしまうのです（他の行動は問題ない）――と、このように本当に個人によって上手くいったりいかなかっ

たりするので、すでに述べたとおり、動線のデザインは自分で試行錯誤していくことが重要です。奥深いものです。

RSAF サイクル

動線リマインドとしてハブを使う際に、心がけると良いあり方が **RSAF サイクル** です。



rsaf

RSAF とは以下のサイクルを指します。

- 1: ツールを見る(Read)
- 2: タスクを選ぶ(Select)
- 3: 行動する(Act)
- 4: 行動後、ツールに戻ってきて状況を更新する(Feedback)

RSAF サイクルは、いわばツールをハブにしています。ツールとは何らかのタスク管理ツールです。手帳だったりスマホアプリだったり PC で使ってる Web アプリだったり、あるいは見せ方もリストだったりテーブルだったりボードだったり、と色々ありえますが、ともかく何らかのツールを使っているわけで、このツールをハブにするのです。

といっても、いきなりお使いのタスク管理ツールを RSAF 的に使うのは難しいでしょう。そもそもお使いのツールが定まってないかもしれません。段階としては、以下のようになります。

- 1: 選定の段階
  - タスク管理として常用しているツールがないのなら、まずは一つ決める
- 2: 親近の段階
  - 1 のツールを、既存の動線（ジョイントでもハブでもいい）に置く

    - 置けば触れる（使う）機会も増えます
    - よく知らない人と親しくなるために、会う機会を増やすイメージです
- 3: RSAF の段階
    - 1 のツールを RSAF 的に使う
    - つまり普段扱うタスクをなるべく全部ツールに入れて、RSAF 的に過ごします

前段として、選定と親近が必要というわけです。あちこち色んなツールに浮気していては定着しないので、これを使うんだと決めてください。そして実際に使う機会を増やすために、既存の動線に仕込んでください。やりやすいのは「ツールにタスクを追加する」とか「ツール内のタスク一覧を見る」といった指示を書いて、動線に置いておくことでしょう。もちろん、動線に仕込まず自分の意志でこまめに使える人はそれでも構いません。ここは人付き合いと同じです。

たとえるなら「親友をひとりつくる」ようなものです。親友をつくるためには、まずはコミュニケーションの機会を増やして仲良くならないといけません。機会をどう捻出するかは人次第ですよね。定期的に予定をつくるかもしれませんし、不定期で会って適当に喋ることを繰り返すだけでいける人もいるでしょう。もちろん、機会を増やしたからといって必ず親友になれる保証はなく、合う合わないはあります、合わない場合は距離を置くでしょう。ツールも同じで、このツールなら自分と合いそうだというものと出会える（あるいは自分が見出す）まで試行錯誤は続きます。

そうして一つのツールにある程度触れられるようになってから、ようやく RSAF サイクルを試せます。普段の生活で扱うタスクをなるべく全部ツールに入れた上で、常にそのツールを見て次の行動を決めるようにします。これはGTDでいうネクストアクション（の実現）そのものであり、別の言い方をすれば **ツールに従って行動する機械になれ** ということです。だからといってロボット戦略を取る必要はなく、何度も言うようにツールは色々あり、ツールの使い方も色々あります。自分の好みで構いません。また親友のたとえをしますが、親友の価値は親友が優秀かどうかではなく、親友と合うかどうかだったり、長らく過ごしてきたことによる信頼関係や愛着だったりしますよね。ツールも同じことです。自分に合うものを使ってください。

リマインド

リマインドの概要

リマインド（Remind） とは指定タイミングに指定要件を思い出させる仕組みです。事前にタイミングと要件を仕込んでおくと、そのタイミングになったときにその要件が何らかの形で知らされます。たとえ要件を忘れていても、知らされれば思い出せるので問題ないわけです。

リマインドを行う手段を **リマインダー（Reminder）** と呼びます。またリマインダーによって設定された個々のリマインド設定をリマインダーと呼ぶこともあります。リマインドを自分だけで実現するのは不可能ですから、リマインダーなる道具を使う必要があるのです。

リマインドのメリット

リマインドのメリットはただ一つで、**予定を確実かつ手間なく思い出せること** です。予定というと開始時間を守らないといけない事柄ですが、意識するだけで守れるとは限りません。かといって常にカレンダーを見ておくわけにもいきませんし、見たとしても思い出せるとは限りません（特に忙しいとき）。しかしリマインダーを設定していると、開始前に知らせてくれるので思い出せます。知らせてくれさえすれば思い出せるのです――と、まるで 100% のように書いていますが、もちろんそんなことはありません。それでも何も設定しないよりははるかに確率は上がります。

何より、一度設定しておけば、あとは忘れることができる（思い出せてくれるから忘れても問題ない）のが大きいです。予定を常に意識しておくと疲れますが、リマインダーを設定しておけば意識を捨ててセーブできます。

自分ひとりで完結できるのもポイントです。権力や経済力がある人なら、部下に頼んだり秘書やマネージャーを雇ったりすればいいだけですが中々そうもいきませんし、あらゆる用事を任せられるわけでもないでしょう。リマインダーであれば、自分ひとりで設定できます。

まとめると、以下のようになります。

- 何もしない → 忘れる
- 常に意識する → 忘れにくいが、疲れる
- 人に頼る → 力が必要だし、何でも頼めるとは限らない
- リマインド → 忘れても思い出せる、疲れない、設定は自分で行う必要がある

リマインドはバランスの良い方法なのです。

リマインドの限界

リマインドが行えるのは要件――事前設定した「思い出すためのヒント」を提示するところまでです。それを受けた上で実際に思い出したり、行動したりといったところは自分にかかっています。リマインダーはそこまではしてくれません。

ただしリマインダー側で工夫することはできます。**リマインドの強度を上げる** と表現しますが、より確実に思い出せるように、またすぐ行動に起こせるように、強めの設定を行うのです。

たとえば以下は、番号が大きいほど強度が挙がっています。13:50、つまり 10 分前にリマインドするものと考えてください。

- 1: 画面上に「14 時」のメッセージを出す
- 2: 画面上に「14 時から会議」のメッセージを出す
- 3: 画面上に「14 時から昇格面接」のメッセージを出す
- 4: 画面上に「14 時から昇格面接」のメッセージを出す + アラーム音も鳴らす

- 5: 4 のリマインダーを 13:50 のみならず 13:30 にもセットする

1〜3 はメッセージだけですが、3 の方が具体的で重要性を思い出しやすそうです。4 については音も出しているので、はっとして行動を切り替えやすいでしょう。5 については、リマインダーを 2 つに増やしています。かつ、13:50 という 10 分前でなく、30 分前にしていることで、おそらく行動にも余裕が生まれるでしょう。仮にそこを逃してたとしても、従来の 13:50 でもう一度思い出せるチャンスがあります。

特に 1 と 5 を比べると、5 の方がリマインダーとしてかなり強いことがわかるでしょう。その分、5 のリマインドは設定の手間も（おそらく 1 よりもだいぶ）面倒くさそうです。強度と手間はトレードオフです。

リマインダーは様々

リマインダーとして何が思い浮かぶでしょうか。数個くらいの人が多いと思いますが、実は様々なものが存在します。

アクティブリマインドの例

一番有名なのは目覚まし時計でしょう。事前に起床時刻を設定しておくと、その時間が来たときに音（アラーム）で知らせてくれます。目覚まし時計は以下の性質を持つリマインドということができます。

- 目覚まし時計
  - タイミング = 起床時刻
  - 要件 = 起きる
  - 知らせる手段 = 音（アラーム）

ただし、目覚まし時計に要件（言語として記述される必要があります）を設定する機能はありません。ツール自体が「目を覚ますための道具」なので、要件は言わずとも固定されており、「起きる」としています。何のために起きるかはわからず、ここは自力で思い出せなくてはなりません。

次にスマホなど携帯電話のアラーム機能を思い浮かべてください。目覚まし時計と違うのは、メッセージを表示させたり振動（バイブ）させたりが可能なことです。性質を見てみましょう。

- 携帯電話のアラーム機能
  - タイミング = 起床時刻
  - 要件 = 指定可能
  - 知らせる手段 = 音（アラーム）、振動（バイブ）

目覚まし時計よりも豊富であることがわかります。要件を思い出せない人なら事前にちゃんと書いておけばいいですし、音がうるさくて鬱陶しいと敏感な人ならバイブだけで済むかもしれませ

ん。また、設定も複数つくれるので、7 時に起きたいとしたら 6:30、6:35、6:45 など多段で設定することもできます。

次に、録音目覚ましという手段を考えます。これはアラーム音として自分が録音した音声を使う目覚ましです。iPhone の例になりますが、音声を自分で準備すれば純正でも可能ですし、アプリもあります。

- 録音目覚まし
  - タイミング = 起床時刻
  - 要件 = 指定可能
  - 知らせる手段 = 音（録音した音声）

知らせる手段が音である点は変わりませんが、使う音声が変わっています。何なら変えられます。アラーム音では起きれない人や、アラーム音だとノイジーすぎてすぐに切って無視してしまう人にも効果があるかもしれません。

すでに述べましたが、これらは仕組みの方から働きかけてくれるものであり、アクティブリマインドと呼びます。

トラップリマインドの例

アクティブリマインドほど確実性はないですが、リマインドとして機能する手段は他のもあります。

これもすでに挙げましたが、トラップリマインドです。動線上にタスク（を想起するもの）を置いておけば、自然と目に入るので思い出せる、というものです。たとえば平日出社する前にごみ捨てを忘れず行ないたければ、前日の夜にでもごみ袋を玄関に置いておきます。玄関は、出社する際に必ず通る動線のはずですから、出社前にちょうどごみ袋が目に入るはずです。

これも性質として記述してみましょう。

- 玄関ドアに物タスクを仕込む
  - タイミング = 玄関を通るとき
  - 要件 = 置いた物から思い出す必要がある
  - 知らせる手段 = 目に入る

ちなみにトラップリマインドは、仕掛け方を工夫すればアクティブリマインドになります。すでに「タスクを書いた紙を靴に入れておく」例を挙げましたが、視覚以外の五感に訴える手段（この場合は触覚）を使うと、より気づきやすくなるのです。強度が強いわけですね。これも性質を挙げておきます。

- 靴の中に「タスクについて書いた紙」を入れる
  - タイミング = 靴を履いたとき
  - 要件 = 紙を踏んだときに思い出すか、紙を広げて読んで思い出す必要がある
  - 知らせる手段 = 足裏または足指先の触感

ごみ袋を置いておくやり方と似ているようで、だいぶ違うことに注目してください。どちらでも通用する人はかんたんな方を使えばいいですが、片方しか通じない人は通じる方を使った方が良いですし、どちらも通じない人は別のリマインド方法を考えなくてはなりません。

ロケーションリマインド

ロケーションリマインド とは、特定範囲に入ったら or 出たらリマインドする、というものです。また iPhone の例になりますが、リマインダー機能に搭載されており、位置情報を追加することで行えます。

使い道は色々あります。会社を出たらリマインドする、自宅に近づいたらリマインドするなど普段の行動のどこかに仕込めば、定期的なタスクを忘れずこなせます。

他にも普段は足を運ばないデパートに近づいたらリマインドする、としておけばそのデパートで買いたかったものを買うかどうかをそのとき（そのデパートに近づいたとき）に判断できます。思い出せる自信がないなら「～～を買うかどうか決める」など要件もちゃんと書いておくと良いでしょう。このようなリマインドは何気に重要で、これを設定しておけば頭の片隅からも捨てることができます。デパート一つだと実感が湧きませんが、仮に買い物好きで色んな場所に行くようなライフスタイルなのだとしたら、ばかにならないと思います。たとえば 10 の場所分を全て頭の片隅で意識しておくのと、さっさとロケーションリマインダーをつくって忘れておくのとでは、だいぶ負担に違いが出ます（※1）。

- ※
  - 1 このように要件を頭から外に出して忘れてしまうことは重要です。頭に留めておくと疲れますし、集中や注意も削がれやすいからです。一つ一つ、一瞬一瞬は微々たるものでも、これが何時間、一日と続くと結構な負担になります。別のたとえをすると、貧乏で収入や出費をいちいち気にする人と、ある程度余裕があって些細な支出を一切気にしなくていい人とで、どちらが認知的に余裕があるかといえば後者です。言うなれば、**リマインダーは認知資源という貴重なリソースを節約する節約術** とも言えます

ヒューマンリマインド

本節冒頭では人に頼ることはリマインドではない、のような書き方をしましたが、あえてリマインドとして頼ることもできます。これを **ヒューマンリマインド** と呼びます。

以下に例を挙げます。下に行くほど使いやすいと思います。

- 秘書にスケジュール管理全般を任せる
- 部下に「会議忘れてそうなら声掛けてくれ」と頼んでおく
- 同居人と普段から予定を共有しておく（忘れていたら思い出させてくれることを期待）
- 友達にモーニングコールをお願いする

人に頼る分、仕込むのがかんたんなのがメリットです。特にツールの扱いに慣れてなかったり、い

ちいちツールを操作して仕込むのが苦手な人には向いています。一方で、容易に想像できるように、融通が利きづらい上に不確実性も高いのがデメリットです（一部の秘書など高品質な手段は除く）。また冒頭でも挙げましたが、要件の内容を相手に教える必要があるため、機密性の高いものやプライベートすぎる内容も扱えません。

余談ですが、個人的には、このヒューマンリマインドを誰でも活用できるような仕組みがあったら流行る気がします。推しを持つオタクの人向けに推しがコールを行うとか、友達間でグループをつくってリマインドし合うとか、見ず知らずの誰かのリマインドがたくさん漂う世界でボランティア的に誰かに（リマインドとして）声を届けるなど、色々できそうな気がします。リマインダーが非常に有用な手段である割に、あまり使われていないのは、面倒くささと味気無さによるものだと思っています。人の介入や人の声には、それらに抗える魅力があります。

雰囲気リマインド

本章冒頭にて「環境や人の力に任せる」件は扱わないと書きましたが、少しだけ取り上げます。

環境の力を借りるリマインド手法として **雰囲気リマインド（Atmosphere Remind）** があります。一番わかりやすいのは学校で、移動教室や体育など何かと慌ただしいですが、基本的に問題なく追従できると思います。これは教室という場があり、そこにクラスメイトや先生が居ることによって雰囲気が存在するからです。この雰囲気が「次は移動だ」との圧力をかけてきます。よほど外界を遮断していない限りは気付けますし、そもそも集団生活では「次に皆が参加するべき予定」の形で制約されていることが多いですから「ああ私も参加しないと……」とその気になりやすいです。ヒューマンリマインドと同様、不確実性は高そうですが、意外と強度があります。

デメリットとしては、リマインダーの融通が利かないことです。基本的には何らかの集団生活に身を置き、かつ同じ場所に物理的に集まる必要があります。最近ですとバーチャルオフィスがあるのでリモートでも可能ですし、Remotty などリアルタイムに顔写真を共有させることで皆の様子を可視化するものもあったり、と後者の集合手段についてはある程度融通を利かせられますが、「14 時の会議をリマインドしてほしい」といった個人的なリマインドには応えられません。あるいは雑談が活発な組織であるなら「私、14 時から会議なんすよ～、前もすっぽかしたんで怒られないようにしないとですねあはは」のようにヒューマンリマインドに持ち込む（誰かが覚えていてくれる＆自分が忘れていたときにツッコんでくれることを期待）こともできるでしょう。

余談：なぜ出社するのか

雰囲気リマインドの余談ですが、長くなったので新しい項にしました。

パンデミック以降リモートワークが実現可能だとわかったにもかかわらず、出社に回帰する動きが目立ちます。GAFAM もハイブリッドワーク（週に n 日は出社せよ）ですよね。なぜかというと、従業員のタスク管理能力が乏しいからです。タスク管理が十分行えるのならタスクにせよコミュ

ニケーションにせよ（予定を忘れることなくこなすための）リマインドにせよ問題なく行えるはずですが、これには強い自律性とスキルが求められます。自然に身につくものでもありませんし、学術的な体系があって勉強できるわけでもありませんし、そもそも個人差が大きすぎて体系化できるものでもありませんし、と身につけている可能性も低いです。筆者が本書を書いた理由の一つは、そのような現状を憂いてのことでもあります。

そもそも現代のビジネスは「多忙なシチュエーションに身を置いて瞬発的に行動していく」「密に連携していく」を是としており、その根幹は生産性競争を是とする搾取的価値観にあると私も考えます（他にも成功者や権力者のメンタルモデルが偏っている等もありますが割愛）。

この多忙縛りとも言うべきあり方にはタスク管理もリマインダーもさして役に立ちません。次に何をするかは雰囲気が教えてくれるし、何なら雰囲気に従わなければならないからです。つまり強度強めの雰囲気リマインドが使われているに等しいです。たとえば会議についても、その場で「ちょっと話そうか」と始めたり、あるいは雑談中に仕事の話が始まったりします。

まとめると「多忙縛りなやり方で成果を出せている」、かつ「タスク管理やリマインダーを使いこなすスキルと自律性はおそらくない」の双方が備わっています。後者を使えばリモートワークでも仕事はできますし、リモートなので出社よりも各自のペースを尊重できてQoLも上がりますが、前者ができて後者ができない状態でこれを目指すかというと、ノーではないでしょうか。

リマインダーをつくるには

リマインダーをつくる際の戦略は二つあります。

一つが **オンデマンド戦略** です。On Demand――要求があったらすぐに、とのニュアンスですが、「あ、これリマインダーにしておいた方がいいな」と気付いたら、その時点ですぐ行動して設定します。これはスタンスの章で取り上げたスプリンターそのもので、タスク管理ツールに入れるのすら面倒だし、すぐ終わるんだからその場でやってしまおうというわけです。逆を言えば、リマインダーの設定を「すぐ終われる」レベルで行えるようになっておく必要はあります。リマインダーを手元に持っておくことと、設定や運用などに手慣れておくことですね。たとえば筆者は「2週間後に面談があるから準備しておくように」と言われたとしても、（PCが手元にあるなら）その場ですぐに「面談当日のリマインダー」と「面談以前にどこか3日くらいチェックポイントとしてのリマインダー」を仕込むことができます。1分もかかりません。メモの章にてメモは素早く書くことが大事と書きましたが、同じことです。この手先の早さは妥協せず、とことん追求してください。早ければ早いほどスプリントしやすくなります。

もう一つが **定期戦略** です。筆者が使っているのは「カレンダーの今日の予定を見て、それぞれ10分前にリマインダーを入れろ」タスクを平日朝一に設定している、というものです。これにより毎日出社時に筆者はカレンダーを見て、予定がたとえば3つあったのならリマインダーも3つ登録します。会社のiPhoneのアラームを使っており、すでに何度も使っているので9:50、10:20、10:35、10:50など選択肢がたくさんあります。選んでオンにするだけです。用件はいちいち入れていません、アラームを聞けば何の会議か思い出せるし、思い出せなくてもカレンダー

を見ればわかる（そもそも大体リモートなのでカレンダーが会議開始の動線でもあります）からです。つまり一日一回、今日の予定をリマインド化するタスクを入れているわけですね。筆者はこれで足りていますが、別の頻度でも構いません。肝心なのは「リマインダーを設定するタスク」を定期的に仕込むことです。タスクだとわかりづらいなら予定でも構いません。まずは定期戦略に慣れる意味でも、頻度は毎日が良いでしょう。一日の最初にその日必要なリマインダーを全部セットする、から始めてみるのです。

この二つの戦略は併用すると便利です。定期戦略により今日の予定を取りこぼさなくなりますし、ちょっと離れた予定や仕掛けておきたい用件などはオンデマンドでさっさとやります。オンデマンドが苦手なら、週に一度くらい「リマインダーを設定する会」なるものを開催するのもいいでしょう。

どちらの戦略にせよ、重要なのは**リマインダーを仕込む機会を意識的に確保する** ことです。馬鹿らしくて、面倒くさいかもしれませんが、仕込みとはそういうものでしょう。このちょっとの手間により、頭の外に追い出せてしかも忘れない（というより直前で思い出させてくれる）ようになれるのですからお得です。

まとめ

- 動線やリマインドを工夫することで忘迷怠を減らせる
  - 特に予定など「開始タイミングを逃すと問題になる事柄」を忘れないために使う
- 動線にタスクを置いておくと、目に入るので思い出せるし、思い出せたら行動できる
  - タイミングが早すぎると認識から落ちる（ネグレクテッド）
  - 逆に遅すぎるとそもそも開始が間に合わない（ロスト）
  - ネグレクテッドもロストもしないバランスが重要だが、この設計や調整は意外と難しい（動線のデザイン）
- 動線にはハブとジョイントがあるが、可能ならハブを確立したい
- リマインダーはよく知られているが、やり方は意外と豊富
  - リマインダーは知らせてくれるところまでしかしない。行動に確実に繋げたいなら強度を強くするべきだが、仕込む手間とのトレードオフである
  - ヒューマンリマインドや雰囲気リマインドなど、不確実だがツールに頼らなくてもいいやり方もある
  - 必要なときにすぐ仕込むことと、定期的に仕込む機会をつくって備えることの両輪を使うと死角がない

=== Chapter-17 習慣 ===

日課、習慣、ルーティン、ルーチンタスク（定期タスク）――個人タスク管理において「継続的に

こなす営み」は避けては通れません。ここまで断片的には取り上げてきましたが、本章にて改めてまとめます。特に習慣なるものの解説にフォーカスします。

本章では「継続的にこなす営み」を **日課、習慣、ルーチンの三つに分けます**。このうち日課とルーチンはすでに解説しているので、かんたんな解説と他章へのリンクに留めます。本章では残る習慣について詳しく解説していきます。

日課

概要

日課（Daily Duty） とはデイリーミッションです。MUST や SHOULD のニュアンスが似合うような事柄を任務（ミッション）と呼ぶことがありますが、これを毎日課したものが日課です。英語では Duty としていますが、**正直** Mission でもいいと思います。

日課の管理方法

日課はその性質上、1日1回やるかどうかを管理して、全部やればオッケーです。

これは実は習慣トラッカーで管理できます。習慣トラッカーは、名前こそ習慣トラッカーとなっていますが、実は日課トラッカーと呼んだ方が正確です。

日課をどう選ぶか

ここで問題となるのは日課の選び方ですが、自分次第です。**何を日課にするかは自分で決めます**。

自分がやらなければならない（MUST）と感じたことや、やるべき（SHOULD）だと思ったことなどは何でも日課にすれば良いでしょう。もちろん何でもかんでも日課にしちゃうと、全部達成するのが難しくなるので、スモールスタートで少しずつ増やすのが無難です。

日課の実施タイミングは自由

日課はその日のうちに達成すれば良いものです。言い換えると**いつ達成しても構いません**。

無理に予定化して、そのとおりに行動するのが辛いと感じるくらいなら、リストアップだけしておいて好きなタイミングで消化する方が良いです。習慣トラッカーもまさにそうなっています（日課をリストアップして各々の達成可否を記入するだけ）。一方で、実施のタイミングが任意だと怠けや

すくなります。実施タイミングの強化と怠け具合はトレードオフです。

理想は日課ごとに実施タイミングを設計することです。たとえば犬の散歩は、おそらく犬から催促してくれるので何も管理しなくてもいいでしょう（催促されたら実施する、とのイベントドリブン）。花の水やりと花壇の手入れは、花から催促してくれるわけではないし、忘れてもいけないので、おそらく予定化なりリマインダーなりを入れて厳しくした方が良いでしょう。また仮にあなたの職業はクリエイターで、スキルの鍛錬、特に鈍らせないことが重要で、1日1時間くらいは練習が必要だとします。一方で仕事は忙しさと暇の波が激しく、生活自体が不規則になりがちだとします。スキル練習は日課にするでしょうが、いつ行うでしょうか。その日の状況も読めませんし、実施の管理はせず空いたときにやりますか。それともカレンダーを見て「スキル練習」との予定を毎日入れますか。それとも朝6時から1時間やる、のように時間まで決めてやってしまうでしょうか――もちろん正解はなく、人それぞれですが、このように日課の実施タイミングは個別に考えた方が（一見すると面倒ですが）生活としても自然であり、うまくいきます。

解説箇所

戦略の章にてTrackerとして解説しています（平易に説明するため「習慣」という言葉を使っています）。またタスクソースの章でも維持事項の文脈で、定期的なタスクをつくることの重要性を述べています。

ルーチンタスク

概要

ルーチン（ルーチンタスク, Routine Task）とは定期的に実行するタスクを指します。定期タスクと呼んでも構いません。定期的に実行する性質があるかどうかのみが争点であるため、日課や習慣もルーチンとして扱えることがあります。ルーティンワークとかルーティンパフォーマンスといった言葉もありますが、いったん忘れてください。本章ではルーティンではなくルーチンと書きます。またルーチンタスクと書くことにします。

ルーチンタスクとはタスクに頻度がくっついたものともいえます。「ごみ捨て」はただのタスクですが、「毎週火曜日と金曜日にごみ捨て」はルーチンタスクです。「ごみ捨て」タスクに「毎週火曜日と金曜日」という頻度をセットしているわけです。

大半がルーチンタスク

私達が抱える行動やタスクの大半はルーチンタスクです。日常生活にせよ、仕事にせよ、特定の頻度でこなしていることが数多くあるはずです。人次第ですが、一日におおよそ20〜60の

ルーチンタスクを抱えていると思います――というと多く聞こえるかもしれませんが、たとえば出社前の「朝ご飯を食べる」「歯磨きをする」「身だしなみを整える」「出社準備をする」、これだけでも 4 つです。もちろん「布団から出る」「トイレに行く」レベルで細分化すればいくらでも増えますが、さすがに極端ですね。そうではなく前者の粒度で洗い出した場合に 20〜60 くらいになるという意味です。

出社前の行動はわかりやすいですが、ルーチンタスクは他にもあります。冒頭のごみ捨ては火曜日と金曜日のみに発生しますし、同じごみ捨てでも可燃ごみと不燃ごみは曜日が違います。リモートと出社を使い分けるハイブリッドワークの人は、出社の日には（準備や片付けに関する）ルーチンタスクが増えるでしょう。

ルーチンタスクを管理するメリット

さて、そんなルーチンタスクですが、別の言い方をすると私達は **それなりにたくさん存在するルーチンタスクを、何も管理せず頭だけで判断して日々消化している** とも言えます。では、そんなやり方で最適な消化ができているでしょうか。「あー、あれやるの忘れてた」「あれもこれもしなきゃいけないんだけど、あー混乱するどうしよう」などと忘迷怠が発生したりはしないでしょうか。十中八九発生します。あるいは発生していることにさえ気付いていません。

ルーチンタスクをタスク管理すると忘迷怠を減らせます。どのルーチンタスクを、いつ、どの順番で実行するかを全部制御できるので、無駄なくこなせますし、やり忘れもなくせるわけです。もちろんそんな単純ではなく、サボったり先送りしたりもよく発生しますし、そもそも自分が抱えるルーチンタスク全部を言語化してツールに投入できるとも限りません（し通常そこまで厳密な管理はできません）。それでも、何も管理せず頭だけで抱えるよりははるかに楽できますし、忘迷怠も減らせます。

ルーチンタスクを管理するには

専用のタスク管理ツールを使います。

特に頻度を扱う能力を持つツールが必要です。ただの TODO リストやアナログなやり方では不可能ですし、カレンダーアプリであっても（何十というルーチンタスクを扱うには手間が大きく）厳しいでしょう。そもそも頻度からして「毎日実行したい」「2日に1回くらい」「毎週火曜日と金曜日」「週に一度くらい」「月末」「月初」など色々存在するわけで、これらを扱えなくてはなりません。やはり専用のツールが必要です。

さて、そのツールですが、Todoist や タスクシュート が挙げられます。ルータム や Routinery などルーチンタスク管理に特化したアプリも出てきています。これらに限らず頻度を扱えるツールは数多いですが、たいていは力不足です。ルーチンタスク管理では **とにかく大量のタスクを何度も操作する必要があるので、軽快に操作できる軽いツールが必要不可欠です**。回数で言うと、一日に何十何百回と操作するのはざらです。

既存ツールでは気に入らない場合は自分でつくることも視野に入れます。それくらいにデリケートなツールです。筆者も自分でつくっていたり、デジタルではなくアナログでも実現するための理論を整理した結果、電子書籍を書くに至ったりなど色々試行錯誤しています。ちなみにアナログだと相当面倒くさいですし、書籍の反応もまさにそのとおりで芳しくありません。やはり専用のデジタルツールを使うのが良いでしょう。

解説箇所

本書では戦略の章、Robot にて解説しています。

またロボットというほど厳しい管理でなくとも、ある程度雑に管理することもできます。たとえば習慣トラッカーでルーチンタスクを並べるだけで（何も使わず頭の中だけで頑張るよりは）違います。同様に戦略の章にて取り上げているので、ルーチンタスクと向き合いたい方は参考にしつつ、自分なりに模索してみてください。

習慣

概要

習慣（Habit） とは、定期的に例外なく行う事柄です。まず定期性があり、1日1回だったり1日2回だったり3日に1回だったり頻度は様々ですし、厳密に特定の間隔を刻むとも限りませんが、定期的に行ないます。次にこれを一度の例外も許さず、一度も漏らすことなく確実に行ないます。逆を言うと、例外なく確実に行えるほど染み付いているとも言えます。習慣とは **定期的に行う、定着した事柄** ということもできます。

習慣はスキルのようなものです。スキルとはただの知識や経験ではなく、難なく安定してこなせるほど訓練して定着させたものだと思います。習慣も同様で、訓練という言葉が似合う世界です。定着させるためにはそれなりに積まねばなりません。一瞬で身につくものではないし、楽して身につけようと考えることがそもそも間違いです。つまり習慣には努力を要します。

習慣というと新しく手に入れるイメージがあるかもしれませんが、逆に「悪い習慣を断ち切りたい」ケースもあります。習慣には良いものと悪いものがあり、良いものは手に入れるし、悪いものは手放さねばなりません。それぞれ異なる戦略が求められますが、いずれにせよスキルのようなもので、努力は必要です。

習慣でないもの

日課は習慣ではありません。すでに述べたとおり、日課はただのデイリーミッションであり、取り組

むタイミングは自由だからです。習慣は日課として管理するまでもなく、自動的に行うものです。逆を言えば、管理するまでもなく毎日必ず問題なく消化できるほどに至った日課は習慣と呼んでもいいでしょう。

ルーチンタスクも習慣ではありません。ルーチンタスクは「頻度を持ったタスク」であり、タスク管理ツール上で上手く扱うための概念です。たとえば毎日 30 分の散歩を行ない場合は、「散歩する」タスクを「頻度 = 毎日」で設定して、かつ朝に目に入るようタスクリストの上の方に配置する、といったことをします。そうすれば毎日タスクリストを運用しているだけで、おそらく朝に「散歩する」が目に入るから（ちゃんと行動すれば）散歩をこなせるというわけです。が、これは結局自分で指示をつくって自分で従っているにすぎません。定着とは言い難いです。一方、習慣とは、管理するまでもなく毎日（よほどのことがない限りは）必ず散歩をしますし、できます。逆をいうと、ルーチンタスク管理をしていて、もう管理せずとも勝手にこなせるくらいになったとするなら、それは習慣になったと言えるでしょう。

そしてもう一つ、**思考習慣も習慣ではありません**。習慣と聞くと、考え方や取り組み方といった形の習慣を思い浮かべる人もいるかもしれませんが、本書ではこれら思考習慣とはみなしません。かわりにモットーという形で扱っています。モットーは状況に応じて適切な行動を行う（応用と呼ぶのでした）必要があり、これはスキル（技能）というよりアビリティ（能力）です。才能や特性に依存します。

習慣は 3 種類

習慣は三種類あります。

- 良い習慣（Good Habit）
  - 取り入れたい・続けたい習慣
  - 新しく採用したり、すでに定着していても微修正してもっと良くしたりする
- 悪い習慣（Bad Habit）
  - やめたい・薄めたい習慣
  - 思い切ってやめたり、少しずつ頻度や機会を減らしたりする
- 微妙な習慣（Odd Habit）
  - 良い習慣か悪い習慣か判別がつかない習慣
  - そのまま放置したり、良い or 悪いと判断して対処（良いならちゃんと採用するし悪いならちゃんと捨てる）したりする

習慣というと良い習慣を増やしたいとか、逆に悪い習慣をやめたいなどと考えがちですが、両方存在するため両方を相手にするのが良いです。また、どちらとも取れない微妙な習慣もあり、これらも良いか悪いかに倒す（あるいは倒せないからと戦略的に放置する）必要があります。

習慣は無限に抱えられるものではないですし、同時に多数を扱えるほど人間は器用でもないため、取捨選択が必要です。そのとっかかりとして良い、悪い、微妙の 3 つの軸で考えるととっつきやすいのです。

習慣の意義は行動の最大化

習慣のメリットとは何でしょう。なぜ習慣が欲しいのでしょうか。

それは強くなりたいからです（※1）。

強くなるためには行動が必要です。行動すれば知識や経験、生理的なレベルでは刺激や負荷が得られて、脳が学習・対処を行うからです。一方で、誤った知識や過剰な負荷はダメージにもなり、下手すれば私達を心身的に壊します。要は行動を重ねれば重ねるほど強くなります（ダメージを負う行動はなくせばなくすほどダメージも減ります）。

となると、あとはいかにして行動を増やすかですが、ここで習慣が登場します。習慣とは定期的に例外なく行うものであり、**行動の最大化** とも言えます。たとえばある能力を得るために、1日1時間の行動が300回必要だとすると、理想は300日間1日も欠かさず行うことです。意志の力で完結するのは難しいでしょうが、習慣であればかんたんです。習慣ゆえに勝手にやりますし、やれない方が落ち着かないくらいで、やれない日があっても捻出しようとさえするからです。

好きこそものの上手なれと言います。好きでやっている人は自分の行動を努力と思ってないので、好きじゃなくて努力している人では勝てないわけです。これと同じことが習慣についても言えます。習慣で行動している人も、自分の行動を努力と思っていません。

- ※
  - 1 強くなる以外の目的もあるでしょう。たとえば快適に過ごすために日頃からこまめに掃除する習慣をつくりたい、などです。これはそのとおりで納得できる理由です。筆者としては表現の違いだと考えます。こまめに掃除する習慣は、言わば掃除をこまめに行う力を得るという意味で「強くなる」とも言えます。表現の違いでしかないので、強くなりたいでも快適でありたいでも、お好みの表現を使えば良いと思います。

習慣の限界

二つあります。

一つは、無限に抱えられるものではないということです。どれだけ抱えられるかは人次第であり、抱える習慣次第でもあります。体力と時間は有限なのでわかりやすいですが、集中力や注意力は見過ごされがちです。

たとえば一日一冊、新しい本を読む習慣が定着しているとしても、読書には集中や注意を要しますから、これらが枯渇していない時間帯かつ頭のエンジンがかかり始める時間帯（通常は起床直後～数時間以内）に行うのが良いでしょう。頭の性能が高い人や、日中さして疲れることをしない人は就寝に近い側の時間でもこなせると思いますが、おそらく仕事などで忙しくなるとこなしづらくなるはずです。自身のキャパシティの限界を超えて、読書習慣が入らなくなったわけです。あるいは無理やり入れることもできるでしょうが、身にならないでしょう。目が滑って思う

ように読めないし、休んで再開してもやっぱり滑る、との経験をした人は少なくないと思います。特に昨今はスマホなど認知資源を奪う手段にありふれていますから、就寝まで半日以上あるのにもう頭が働かない、なんてこともざらにあります。

もう一つは、習慣はただ行動を繰り返すだけだという点です。行動自体の効果や、行動を重ねることの価値には一切触れません。別の言い方をすると、努力は正しく行わないと意味がないですが、習慣も同様で、ちゃんと強くなるためには正しい（行動を行える）習慣を組む必要があります。正しい行動を重ねないと一定以上強くなることはできませんし、下手な癖などがあると逆に弱化につながります。特に一度ついた癖を取り除くのが大変なように、悪い習慣をやめるのもかんたんではないため苦労します。

ハビットエンジニアリング

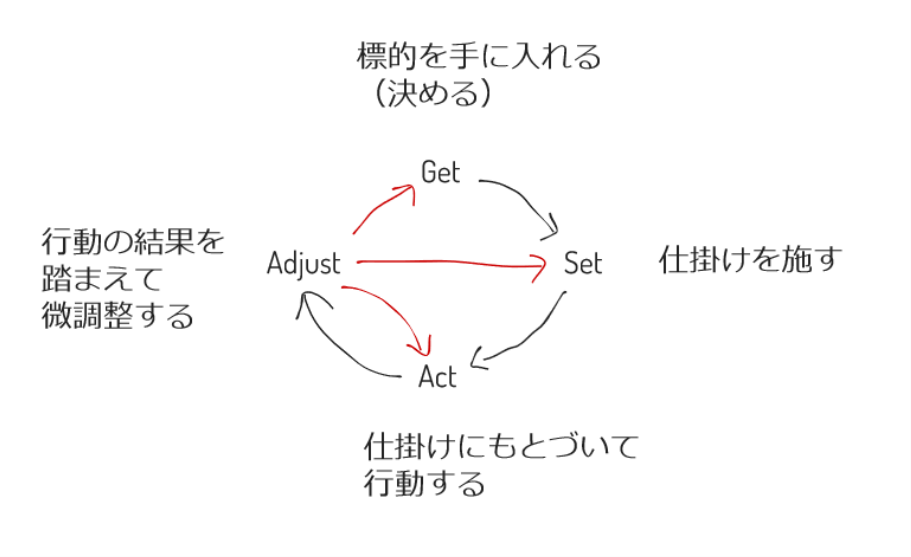
良い習慣を新しく増やしたり、悪い習慣を減らしたり、あるいは微妙な習慣の対処を決めたりすることを **ハビットエンジニアリング（Habit Engineering）** と呼ばせてください。本節ではハビットエンジニアリングと題して、実際に習慣を扱っていくための考え方やテクニックを取り上げていきます。

GSAAD ループ

習慣はいきなり採用したり取り除いたりできるほど甘いものではありません。対象を設定して、日々少しずつ行動して、その結果もチェックして微修正して――といった地道な営みを続けてようやく勝ち取れるものです。

この典型的な流れを **GSAAD ループ** と呼びます。

- 1: Get
  - ターゲットを手に入れる（標的となる習慣を定める）
- 2: Set
  - ターゲットに干渉するための仕掛けを施す
- 3: Act
  - 仕掛けをトリガーに、行動する
- 4: ADjust(Adjust)
  - 行動の結果を踏まえて微調整する
    - 結果とは行動した、しなかった、途中でやめた、想定外のタイミングで行動した等
    - 微調整とはターゲットや仕掛けや行動への働きかけ



GSAAD

一般的なサイクルとは違い、最後の Adjust の行き先が複数あることに注意してください。

1: Get、ターゲットを定める

まずは扱いたい習慣を定めます。良い習慣を増やしたい場合、散歩したいとか読書したいとかボイトレしたいとかいった形で何かしら言語化できるはずです。悪い習慣を減らしたい場合も、寝る前の飲酒をやめたいとか通勤中にネットサーフィンしてるのをやめたいなど何らかの対象があるはずです。良いか悪いかわからない微妙な習慣についても、同様に対象があると思います（微妙な習慣は「悪い習慣候補」と呼ぶこともできます）。重要なのは**「具体的な行動」のレベルで定める** ことです。痩せたいとか賢くなりたいとか本を読めるようになりたいといった願望ではなく、行動にフォーカスしてください。

行動の目線で決めると味気がなく、やる気も湧かなかったりしますが構いません。というより、そんな状態でも「習慣にしたい」と思える行動を選ばなければなりません。もしそういう行動がないのなら、どうせハビットエンジニアリングは続かないので現時点では諦めてください。習慣を扱うためには、具体的な行動をターゲットとして据えることが前提です。これさえできないようでは話になりません。ハビットエンジニアリング以前に、具体的な行動を知るところから始めましょう、となります。願望をブレイクダウンしてどんな習慣（行動）を身につければいいかを理解するとか、単純にアイデア出しや内省のつもりで粘り強く振り返ってみて洗い出してみるくらいの気持ちで向き合ってみることをおすすめします。

2: Set、仕掛けを施す

良い習慣にせよ、悪い習慣にせよ、何らかの行動が伴うわけですが、意志の力だけで毎回必ず行動できるはずもありません。ではどうするかというと、特定の条件が揃ったら自動的に行動してしまうような仕掛けをつくります――というと小難しく聞こえるかもしれませんが、いくつか例を

挙げます。

- 動線とリマインダー
  - 前章で取り上げました
  - リマインダーを設定すれば、たとえ忘れていても気付かせてくれます
  - 普段何度も通る動線にタスクを置いておけば、何度も通るのでそのうちに目に入ります
- ハビットスタッキング
  - 新しい行動を「既存の習慣」にくっつけて運用することです
  - 既存の習慣は欠かさず行うはずなので、そこにくっつけておけばついでに新しい行動もやりやすくなります
  - たとえば毎朝歯磨きをしているが、鼻毛の処理は定着してなくて習慣にしたい場合、歯ブラシのそばに鼻毛カッターを置いておきます
    - より確実にやりたいなら、歯磨き粉を小さな袋や箱に入れて封じて、その上に鼻毛カッターを置いておきます
    - ただ、これは「歯磨き粉を封じる」行動が必要で、この行動を忘れて破綻するかもしれません（習慣のための習慣が要求されている状態）
- IF-THEN プランニング
  - もし XXX が起きたら YYY をする、という風に条件と行動のルールを決めます
  - これは前の章で取り上げたモットーに等しく、向き不向きが分かれます
  - XXX の部分を動線やリマインダーなどで補強できるといいでしょう
- タスクリストやインボックスへの追記
  - 普段使っているタスクリストやインボックスに「新しく行ないたい行動」や「今後やめたい行動（をやめるための何らかの新しい行動）」を追記します
  - 普段使っているだけに、目にするはずです
  - もちろん目にしたから実際に行動に移せるとは限りませんが、目にすら入らず完全に忘れるよりはマシです

他にも色々な工夫があると思いますが、本質的には二つだけです。忘れないように気付かせることと、気付いた後に実際に行動するよう強度を高めることの二つですね。後者、強度の高めた手段の最たる例がヒューマンリマインドや雰囲気リマインドといった「外部から働きかけてもらう」です。逆にリストに書かれたことを忠実に守れるアスリートタイプの人は、タスクリストに一行書いておくだけで行動できたりします。

詳細は前章を参照してもらうとして、肝心なのは自分が実際に行動に起こせる程度の強度を設定することです。そのような仕掛けを考えて仕込むことです。筆者はアスリートタイプなので普段使ってるタスクリストに書くだけでたいてい事足りますが、たいていの人はそうじゃないと思います。また自力ではどうにもできなくて、全面的に環境や他人に頼らざるを得ない人も少なくありません。ここは本当に人それぞれですし、同じ人であってもターゲットにした習慣次第で変わります。

3: Act、仕掛けをトリガーにして行動する

Set にて仕掛けを施すと、ひとまず行動を行えるようになります。

このとき問題となるのが行動の中身です。何を使ってどれだけ行えばいいのでしょうか。たとえば散歩したい場合、以下が考えられますよね。

- 何分歩く？
- どんな服装や靴で歩く？
- 持ち物は？
- どこを歩く？
- 雨が降っていたらどうする？ etc

これら行動の中身、もっと言えば設定値のことを **行動パラメーター（Action Parameter）** と言います。この場合は散歩時間、服装や靴、持ち物、散歩場所や範囲、雨の有無などがパラメーターです。

実を言うと、この行動パラメーターは非常に重要で、仕掛けた行動が習慣として定着するかどうかはパラメーターが適切かどうかで決まります。散歩を例にするとわかりやすいですが、いきなり 3 時間散歩しようとしても続きませんし、雨の多い梅雨の時期に始めたとしてもおそらく雨が鬱陶しくて挫折すると思います。まずは 10 分だけやってみるとか、雨が降っているときはやらないとか、ショッピングモールなど広い屋内に行く機会が多いなら散歩行動をそこにくっつける（雨というパラメーター自体をなくす）といった工夫をすると良いでしょう。

唐突ですが着手ベースという言葉があります。着手さえできたら進捗があった、とみなす考え方です。作業興奮という言葉もあるように、始めた後はしばらく続くものですが、逆に最初腰を上げて始めるところが重たいわけです。そのハードルを超えるために「着手しただけでもえらいんだよ」と自分を甘えさせるのです。この考え方は Act においても有効なのです。

習慣とは行動の定着であり、行動という経験を積み重ねることで固まっていくものですが、**重要なのは行動の質ではなく行動の回数です**。100点よりも60点を取れ、といった類の格言は数多くありますが、それと同じで、習慣化についてもまずは雑でかっこ悪くてもいいので行動回数を増やせばいいのです。そして増やすために、どんなパラメーターなら腰を上げやすいかを模索するのです。

パラメーターの模索ですが、自分なりにやりやすい塩梅を探すことと、一般的なセオリーに従うことの両方を意識してください。自分が行動するかどうかが最重要なので、自分なりに探す方がモチベーションは維持しやすいです。一方、自分ではわからない場合は、セオリーに従ってみてそこから微調整していくと良いでしょう。セオリーについては、現在なら ChatGPT など生成 AI に聞くのが早いかもしれません。

ちなみに「パラメーターを軽くすればいいか」というと、そう単純でもありません。たとえば散歩を習慣化したい場合、「これから散歩します、と音読すれば OK」とみなすことはかなり軽いと思いますが、おそらく効果がありません。音読はできるようになるかもしれませんが、「で？」となって先が続きません。それよりも「とりあえず玄関で靴を履いてみる」の方が続きやすいと思います。靴

を履く行為でも厳しいようなら、それよりも軽い行為から始めましょう。

4: Adjust、微調整する

Act にて行動したか、あるいは行動したかったか、いずれにせよ何らかの結果が出ているはずですので、これを参考に微調整を加えます。

かんたんに場合分けを書きます。

- 行動できた場合
  - より快適な行動を続けられるよう行動パラメーターを調整しましょう（Act）
- 行動できなかった場合
  - 仕掛けを疑いましょう（Set）
  - 行動パラメーターを疑いましょう（Act）
- 行動しなかった場合
  - ターゲットを疑いましょう（Get）
  - 行動パラメーターを疑いましょう（Act）

できなかった場合としなかった場合を区別していますが、意図的です。

できなかったとは、単に忘れていたとか疲れていて無理だったなど、何らかの事情があったというニュアンスです。行動できるような調整をすれば良いです。

しなかったとは、できないわけではないけどなぜか腰が上がらなかった、といったニュアンスです。主因は二つあって、行動パラメーターがきつすぎてハードルが高いか、そもそもターゲットが不適切か、です。Get のステップでは具体的な行動を定めなさい、としましたが、定めるためにはそれなりのモチベーションが必要不可欠です。私たちの心身は正直なもので、その気がないのにテキトーに定めたり、みんなやってるからとい軽率に始めようとしたり、といった程度では行動に移せません。それが「しなかった」として現れます。習慣は今後ずっと、極端に言えば未来永劫続けるものですから、その程度のモチベーションでは話にならないわけです。解としては「モチベーションが湧くまで諦めなさい」あるいは「別の、モチベーションが湧く行動をターゲットにしなさい」になります。

習慣の定着

定着（ハビチュエーション, Habituation） とは、良い習慣を一つ獲得すること、または悪い習慣を一つ断ち切ることを指します。どこまで行けば定着とみなすかについては、主観で構いません。目安は一ヶ月とか二ヶ月とか、一年でも構いませんが、一日も欠かすことなく何らかの管理に頼ることなくこなせることですが、体感的に「そろそろ定着してきたな」というのがわかります。特に GSAAD ループで仕掛けた仕組みを無視し始めたり、一応従ってはいるが「別にもう仕組みに頼らなくてもできるよなー」と感じたりするのが兆候です。

定着を確認できたら、GSAADの仕組みや微調整は一切なくしてしまって構いません。これを**撤収**と呼びます。撤収した後でも定着した習慣（として行っている行動）のクオリティの維持や向上が必要なことがありますが、これはその行動内で行えば良いです。たとえば哲学者がいるとして、普段思索を広げるために散歩の習慣を確立しているとします。散歩の質が思考の質に直結するため、散歩の改良は絶えず行った方が良いですが、これはGSAADのActやAdjustのステップで行動パラメーターを調整するという意味ではなく、散歩という習慣の中で適当に行えば良いのです。

撤収時に習慣（として行っている行動）を忘れてしまうことがあります。このときどうするかもその人次第です。人間ですのでエラーは起きます。**定着した習慣も稀に崩れることがあります**。が、一時的なものなので気にしなくても構いません。何回か続くようなら、そのときに初めて点検すればいいのです。崩れる原因ですが、せっかちでなければ「実はまだ定着してなかった」とのオチは少なくて、最も多いのは生活の変化です。習慣の限界でも挙げましたが、生活が変わると「今行っている習慣が自分のキャパシティに入らなくなった」が起きがちなのです。別の言い方をすると、**生活の過ごし方が慌ただしく変化する人は定着しづらい**とも言えます、習慣化には不利です。そういう意味で、習慣を確立したければ、生活をできるだけ固定するのが実は近道だったりします。いわゆる生活リズムですね。

悪い習慣を減らす

GSAADループは良い習慣を増やす際は理解しやすいと思いますが、悪い習慣を減らすとなると理解しづらいです。すでに「今後やめたい行動（をやめるための何らかの新しい行動）」とまどろっこしい表現も使っています。

そのとおりで、悪い習慣を減らすことは、良い習慣を増やすことよりも理解しづらくなっています。良い習慣を増やすのは何かを「する」と表現できますが、悪い習慣を減らすのは何かをやめる、もっと言えば「しない」と表現することになるからです。否定形の指示は理解しにくいとか、悪魔の証明（無いことの証明）は難しいとか言いますが、「しない」を行動に落とすのは難しいのです。

さて、悪い習慣ですが、減らすアプローチは3つあります。

- 1: 妨害する
- 2: 根本を排除する（盤外戦）
- 3: 撹乱する

1: の妨害するとは、悪い習慣（として行っている行動）を妨害するための何らかの行動や仕組みを追加することです。たとえば依存症というほどではないが寝る前の飲酒をやめたい場合、冷蔵庫内のお酒がゼロ本であれば止まるかもしれません。寝る前に冷蔵庫内のお酒を空にする、との行動をどこかに追加しておけば良いのです。あるいは風味を損なわせるためにガムを噛んだり、歯磨きのタイミングを寝る直前に変えたりなどもできるかもしれません。

これは単純な例ですし、これで成功するとは限りませんが、妨害の発想はこれと変わりません。とにかく悪い習慣を妨害する別の行動や仕組みを差し込むのです。特に直接直前に差し込めない場合は「妨害を行う仕組みを設置するための行動」を仕込みましょう。この妨害行動は習慣化する必要はなく、ルーチンタスクや日課で構いません。

次に 2: の排除ですが、妨害だけではどうにもならない、あるいは追いつかないことがあります。個人の工夫では限度があるということです。そういう場合は、悪い習慣のもとに直接干渉するしかありません。寝る前の飲酒をやめたいが、病気や依存になっている場合、通院するのがベストでしょう。あるいは入眠を妨げるストレス源を特定して、その排除や回避が必要かもしれません。ストレスフルな人付き合いをやめるとか、遠回しに言っても効かない人にはっきりと言うとか、そもそもスマホしすぎで慢性的な睡眠不足なので睡眠時間を増やすよう生活を変えよう（変えるための「良い習慣」を増やしていこう）など原因は様々です。

干渉というと、こちらから働きかけるイメージが強いかもしれませんが、距離を置くといったやり方も可能です。むしろこちらから働きかけるとトラブルになることが多いので、通常は後者のスルーを試した後、それでも離れられないなら腹をくくってこちらから、の二段階を踏むのが無難でしょう。いずれにせよエネルギーも気力も使うことですが、個人の工夫でどうにもならないのなら、あとは腹をくくるしかないのです。くくれないと偶然収まるか、こちらが壊れるまでずっと続きます。ちなみに、このような根本への干渉は盤外戦と呼び、本書でも何度か言及しています。GSAAD ループに限らず、タスク管理全般において重要な視点です。

最後に 3: の撹乱（かくらん）ですが、これは自身の生活環境を劇的に変えてしまうことです。たとえば引っ越しや転職はわかりやすいと思います。結局、私達の習慣は生活環境に左右されていますし、「人間が変わる方法は3つしかない」との格言もあります。曰く、時間配分と住む場所と付き合い人です。いずれも環境を変えるレベルのインパクトがないと変わらないことですよね。だったらいっそのこと環境を変えてみましょう、というのが撹乱です。撹乱の名前のとおり、かき乱すニュアンスです。

微妙な習慣を倒す

習慣には 3 種類あり、良いものと悪いものの他に微妙なものもありました。

微妙な習慣とは、良い習慣なのか悪い習慣なのか判別がつかないことです。というより習慣には良い側面と悪い側面の両方があっても珍しくありません。たとえば筋トレという習慣は、筋力向上や心身への刺激の面では良いですが、免疫力低下や体温変化により体調を崩しやすいという意味では悪いです。

微妙な習慣の扱い方は単純で、良い習慣なのか悪い習慣なのか、どちらかであると決めることです。筋トレの例で言えば、「筋力が欲しいからこれでいいか」と続けるなり、「体調崩すのが不便な割に、別に筋肉あってもあまり嬉しさはないんだよなー正直」と気付いてやめるなり決めてしまうのです。決めてしまわないと、続けるかやめるか迷ってしまって認知資源を食います。このノイズは馬鹿になりません。良い習慣だと決めた場合は、そのまま続けます。悪い習慣だと決

めた場合は、前述のとおり、悪い習慣を減らす取り組みをしてなくします。そういう意味で、**微妙な習慣とは「悪い習慣候補」と呼ぶこともできます。**

決めるのに悩むかもしれませんが、どうせ答えは無いため意味はありません。さっさと決めた方が良いです。決めた後、しばらく生活すると影響が出てくるので、そのときまた調整すればいいのです。良い習慣だと決めたけどやっぱり悪い影響が強いなとわかったら悪い習慣とみなしてなくせばいいですし、逆に悪い習慣だと決めてなくしたのはいいが恩恵の一部がなくなって不便になった等があるなら良い習慣とみなして再度組み込めばいいのです。ちなみに筆者は、筋トレについては良い習慣になったり悪い習慣になったりします。花粉がひどくなる春は悪い習慣に倒してます。

習慣にはただでさえキャパシティの限界があり、たくさん抱えきれるわけではありませんから、微妙な習慣はてきぱきとどちらかに倒して処理してしまいたいところです。この微妙な習慣をさっさと扱うフットワークは、ハビットエンジニアリングの成否を分ける鍵だとさえ筆者は考えます。ここができないと、物を溜め込んでしまうように良い習慣も溜め込んでしまって苦しくなったり、逆に悪い習慣を溜め込んでしまって慢性的な弱化に陥ってしまったりします。

習慣を無理やり手に入れる「捨て身」

ハビットエンジニアリングでは GSAAD ループなど、自力で仕込んで自力で行動することを念頭に置いています。一方で、特性上そのような自律的な行動が苦手な人もいます。そういう人でも習慣をつくったり消したりするにはどうすればいいでしょうか。

解はありません。強いて言えば、Set の項で述べたように環境や人など外部の力に頼った仕掛けを工夫するか、悪い習慣を減らすアプローチの一つである撹乱――引っ越しのような環境の変更を行って無理やりかき乱すか、くらいです(※1)。

ちなみにプロが存在する世界では、このやり方が使われがちです。職人における弟子入りやスポーツのクラブ活動、寮生活や合宿などでは生活を厳しくコントロールされ、かつこれが年単位など長期に渡って続くので嫌でも（その世界に関する）習慣が身につきます。別の言い方をすれば、習慣とは本来それくらいの荒療治が必要なものです。とはいえ現実的に常に荒療治するわけにはいきませんから、ハビットエンジニアリングの形で地道に行うのです。この地道なやり方ができないのなら、荒療治しかありません。それを捨て身と表現しています。

- ※
    - 1 筆者としては前者の、外部の力に頼ったやり方をもっと開拓できればいいんだろうなと思っています。現状は仕事やパートナーなど強い信頼や利害関係に基づいた組織に入って一緒に過ごすか、カフェで勉強するなど自主活動を行う場所を変えるくらいしか方法が無いのですが、テクノロジーの力でもうちょっと工夫できたらなぁと思うのです。擬似的な仕事仲間やパートナーをマッチングさせてお互いに監視・干渉しながら過ごせる SNS であるとか、学校生活のような拘束の強い過ごし方を誰でも柔軟に使えるようにするサービスなど、やり様はある気がします――が、まだとっかかりが得られませ

ん。難しいのは、単に居場所をつくればいいのではなく、自律的にハビットエンジニアリングできない人の背中を推してくれるような居場所や仕掛けをつくるには、また彼らをやる気にさせて誘うにはどうしたら、ということです。

まとめ

- 継続的にこなす営みは日課、ルーチンタスク、習慣に分けることができる
  - 日課はデイリーミッション、習慣トラッカーにて
  - ルーチンタスクは専用のタスク管理ツールについて
- 習慣とは
  - 定期的に例外なく行う事柄
  - スキルのようなもので、訓練の言葉が似合う
  - 意義は行動を最大化できること
- ハビットエンジニアリング
  - 習慣は良い習慣、悪い習慣、微妙な習慣がある
  - GSAAD ループ
    - ターゲットとなる習慣（行動）を定めて、行動を行うための仕掛けをつくって、行動して、微調整する
    - 仕掛けにせよ微調整にせよ「チューニング」にかかっている。意志でできたら苦労などしない
    - 定着させるには質よりも量。行動の回数を増やすことを心がける。例: 着手は進捗
  - 自力でループを回せない人は環境や人の力に頼るか、撹乱（環境丸ごと変える）くらいしか手段がない
    - スポーツや職人など、厳しい鍛錬を要する世界では使われがち

=== Chapter-18 コンテキスト ===

タスク管理において避けられない多義語の一つがコンテキスト（Context）です。本章ではこのコンテキストについて詳しく見ていきます。

コンテキストの概要

コンテキスト

コンテキスト（Context）とは直訳すると「文脈」であり、ある場面や対象に紐づいた周辺状況や付随情報を指します。非常に多義語であり、ジャンルによっては別途定義が存在することがあります。

タスク管理では、れっきとした定義は存在しませんが多義語ではあり、いくつかのニュアンスで使われています。広義には上記の定義――タスクに紐づいた周辺状況や付随情報、くらいの理解で問題ありません。各ニュアンスの具体的な話はもう少し後でします。今やっても小難しくて意味不明になるからです。しばし概念的なお話にお付き合いください。

一つだけ確実に言えることがあるとするなら、タスクを見て「はいそうですか」と従えるほど単純ではない、ということです。納得感がないと動きたくないし、今やればいいのか後でいいのか判断もしたいですよね。納得感や判断のヒントを得るためには、文脈を知る必要があります。

「目の前のタスクをこなすだけの機械」を脱したいのであれば、タスクをただ消化するだけではなく、タスクの文脈を理解した上で向き合い方を練る必要があります。

文脈を知ることの重要性

文脈を無視するとどうなるか

突然ですが、文脈を無視したことで起こる現象をいくつか挙げてみます。

- 1: Q&A や質疑応答において、直接的な回答が返ってこない
  - 逆に質問を返されたり、聞いてもないうんちくを語られたりする
- 2: 誰かに軽率にアドバイスをしても煙たがれる
- 3: 相談に乗ってアドバイスをしても素直に従ってくれない
- 4: 正論が通じるとは限らないし、通じないことが多い

こういった現象は、いずれも文脈を無視しているがゆえに起こることです。

> 1: Q&A や質疑応答において、直接的な回答が返ってこない

1: は、回答者が質問者の文脈を知るために、直接的ではない回答をしています。あるいは質問者が何かおかしいことをしている（おかしい文脈を持っている）と思っているので、それを是正しようとしています。初心者が上級者に質問するといった場面でよく起こります。

> 2: 誰かに軽率にアドバイスをしても煙たがれる
>
> 3: 相談に乗ってアドバイスをしても素直に従ってくれない

2: と 3: については、アドバイスされた側にも色んな事情や制約や感情があるわけで、アドバイス単体ではあまり役に立たないからそうなります（煙たがったり従わなかったりします）。これはアドバイスする側が、される側の文脈を無視しているか、あるいはアドバイスの内容が的を得ておら

ず事実上無視に等しくなっているとも言えるわけです。

これを一般化したもののの一つが 4: で、

> 4: 正論が通じるとは限らないし、通じないことが多い

いわゆる「正論は通じない」です。正論は、たしかに正しいことやもっともなことを言ってはいるのですが、個人個人の文脈に配慮していないのでほとんど使い物にならないのです。どころか、言われる側からすると「そんな当たり前のことはわかってんだよ」「こっちにも色々あるんだ」「お前が私の何を知っているのだ」と苛つくのではないでしょうか。使い物にならないどころか、逆効果ですらあります。

タスク管理、特にタスクをこなす場合も同様です。文脈と向き合わず、ただただタスクを消化することは、アドバイスをそのまま鵜呑みにして行動するようなものです。もちろん文脈を見るまでもなく、やらないといけない・やるべき・やった方がいいタスクについてはやればいい（し、仕事など文脈関係なしにとりあえずやるしかない場面もあります）のですが、たいていのタスクには選択や判断の余地がありますし、あるべきです。もし中長期的に余地がない場合、それはただの搾取や過労でしょう。悪い生き方とまでは言いませんが、向き不向きが激しい世界です。

文脈を知らないと、選択や判断のしようがなくなります。逆を言えば、文脈を知ればやりようが生まれます。知り切れるとは限りませんし、知ったからどうにかなるとも限りませんが、それでも知ろうとさえしないのと知ろうとするのでは全然違うのです。

文脈を知りに行く営みは身近にある

たとえばコンサルやカウンセリングは、どちらも顧客の文脈を知ろうとします。コンサルでは顧客企業の文脈を主に情報と数字から知ろうとします。カウンセリングは患者との信頼関係を築くことで患者から情報を出してもらおうと試みます。新規事業におけるヒアリング、インタビュー、あるいは共創といったものも同様です。顧客やその他ステークホルダーにはそれぞれ文脈があり、それを知るのと知らないのとではアイデアや事業の組み立て方が変わります。そもそもマネージャーにせよチームメンバーにせよ、日々仕事をする上でメンバー各々の文脈を知りつつ配慮していく、といったことは誰しも行っているのではないでしょうか。

文脈というと小難しく聞こえますが、このように実は非常に身近なものでもあります。文脈がわかりづらいなら事情とか背景などと言い換えてもいいでしょう。

切られないために文脈を知る

このとおり文脈を知るとは身近な営みですが、ここにはある共通点があります。

文脈を知るための試行錯誤がやりづらい、ということです。プログラミングやゲームを行う方なら、試行錯誤はいくらでも好き勝手に行うとの思考回路を持っていると思いますが、一般的に仕事では通用しません。仕事には人が絡んでおり、その「人」という存在に対して試行錯誤が

許されない からです。

人には感情があるし、特に仕事だと忙しくて機会さえ少ないですよね。会議一つ整えるのに調整に苦労する人も少なくないでしょう。感情があるくせに機会は少ないというおおよそ最悪の条件です。こんなものが相手では試行錯誤などまともにできません。最初の一回で（少なくともある程度は）勝ち取れないと、以降は相手にしてもらえなくなります。この理不尽に勝つためには、事前準備して成功率を上げるしかありません。準備の種類は色々あり、礼儀の演出のようなものありますが、本章で扱うコンテキスト――文脈もその一つです。文脈という情報を知り、これらを尊重した動きができれば勝てる確率を上げられる（というより露骨に負ける確率を下げられる）のです。

社会にせよ、個人的な人間関係にせよ、人生の大半においてこの理不尽なゲームが登場しがちです。上手く生きている人は例外なく文脈を知った上で立ち回っています。文脈を無視して成果で殴るケースもありますが、才能の域であり稀なのであてにはできません。文脈を知ろうとしないのは極めて不利なのです。

そもそも集団が行動や意思決定の面で何かと遅くなりがちなのも、文脈を知る猶予が欲しいとの心理が働くからだと筆者は思っています。何でもかんでも正直に正論や本音をぶつけて議論しあう、試す、そうして素早く動いていく、みたいな関係は幻想であり、発達障害者の集団や心理的安全性が極めて高い小集団など特殊なケースでしか成立しません。文脈を知る余裕（**文脈理解猶予** と呼ばせてください）は必要です。いわゆるハイコンテキスト、ローコンテキストといった文化特性は、この文脈理解猶予をどう過ごすかの違いです。ハイコンテキストは「察しろ」で、ローコンテキストは「ちゃんと言え」です。いずれにせよ、察し合うための余裕やちゃんと言い合うための余裕が必要なわけです。

長くなったのでまとめます。

- 人に対しては試行錯誤ができない
- 失敗しないためには一発で決めねばらない
    - 決めるためには文脈という情報が必要、つまり文脈を知ることが必要
- 文脈を知るための猶予は恒常的に確保される（筆者の仮説）
    - だからこそ集団になると何かと遅くなる
    - 一方で特殊な条件下であれば鈍らずに済む
        - 本文では書いてませんが、他にも可能です。たとえば軍隊のような厳しいあり方でも可能です

コンテキストには動的と静的がある

文脈を知ることの重要性を述べたので、次はタスク管理との絡みも述べます。特に種類があることを押さえておきたいです。

タスク管理におけるコンテキストは大まかに二種類に分かれます。動的なものと静的なもので

す。

動的なコンテキスト

動的なコンテキスト（Dynamic Context）とは、状況のことです。

たとえば資料作成をして、1 時間くらい続けた後に休憩をはさんだとしても、あなたの頭の中には資料作成に関する諸々が残っているはずです。ここまでやったとか、この辺は仕方ないので手を抜いたとか、次はここから書こうとか、あそことあそこがまだ甘いから詰めたいなー――など、仕事を行ったことで発生・変化した文脈が色々あるはずです。これらは忘れずにいたいですよね。休憩後にすぐ再開できればいいですが、忙しいなどで再開が数日後になったりすると最悪です。文脈の全部あるいは大部分が飛んでしまうので、思い出すところから始めなければなりません。無理に再開すると、文脈が無視された整合性のない資料になってしまいます。

また、複数人で仕事を行っている場合、特に「現場」の概念があるような慌ただしい仕事の場合、誰かが何か行動するたびに状況が変わるため場の状況は刻一刻と変化します。現場にフルコミットして、絶えずついていかなければ遅れてしまいますし、教育のような丁寧なフォローもいちいちできないでしょう。必然的に現場に居続けるしか選択肢がなく、過酷になりがちです。

このような類の文脈は関わる度に、あるいは絶えず動的に変化するため、動的なコンテキストと言えます。

静的なコンテキスト

静的なコンテキスト（Static Context）とは、動的なコンテキストほど慌ただしくは変化しない付随情報を指します。属性と言っても良いでしょう。

人の気持ち、信念、生活上の制約、性的指向やその他の特性といったものは（その人に紐づいた）静的なコンテキストです。「気持ちは動的に変わるものじゃないか」と思われるかもしれませんが、ここでは根っこはあまり変わらず、その見せ方が変わるだけだ、と解釈しますので静的の範疇になります。ちなみに説得、攻撃、洗脳などで無理やり変えることもできますが、それでも動的というほどではありません。

タスクにも静的なコンテキストが色々紐づいています。資料作成というタスク一つ取っても、なぜ必要なのかとか、どういう経緯で生じたのかとか、資料を目にする人物は誰かとか、作成を行えるタイミングはいつか、場所はどこかなど色々なコンテキストがあります。

具体例を一つ挙げます。

- 「当部門の事業はジリ貧であり、新規事業を開拓していく必要がある」
- 「幹部からの意向でもあり、予算もそれなりに下りている」
- 「当部門ではすべての企画は幹部も目を通す」
- 「コンプラも整っているので就業中しか着手できない」

- 「リモートなので自宅でもできるが、コンプラによりカフェ等での作業はNG」

また、作業者自身のコンテキストについても具体例を一つ挙げてみましょう。

- 「私は新しいアイデアを考えたり人と話したりするのが好き」
- 「私は既存事業で作業員になったりマネジメントをしたりすることに面白さを感じないし、向いてもいない、だから新しい検討で食べていく路線を探りたい」
- 「子供も生まれるし、家も買ったのでお金はもっと欲しい、昇進しなければならない」
- 「でも当部門にいる限り幹部の審査は避けられず、幹部が通すかどうかは内容よりも信頼の面が強い。時間も知識もない幹部のための説明や配慮に手間暇かけるのは違うと考えている」
- 「転職が理想だし、ゆるくは探しているが見つからない。会社の待遇自体は悪くない。総合的に考えて、このまま要領を覚えて上手くやった方が良さそう」

いかがでしょうか。資料作成タスクと向き合う際、ここまでコンテキストが挙がっているのと、何も挙がっていないのとではずいぶんと違うのではないでしょうか。この例では自分としても部門としてもマッチしているので「ちゃんと頑張ろう」とモチベーションを上げやすいでしょう。逆に部門にその気がなかったり、自分にその気がなかったりするなら、真面目に取り組みすぎても辛いだけですので消化試合として手を抜くか、今回は手を抜かないが次回以降はもうやらないと決めるか等が良さそうです。コンテキストがわかれば、こういった判断が行えるのです。

タスク管理における 5 つのコンテキスト

コンテキスト自体の概要、特に必要性と、タスク管理との絡みとして動的・静的があることを書きました。さらにタスク管理に踏み込んでいきます。

具体的にはタスク管理におけるコンテキスト、というものを見ていきます。

5つのコンテキスト

最初に表でまとめます。

No	名前	分類	解説	例
1	タスクコンテキスト	静的	そのタスクが持つ文脈	なぜ必要か、なぜ今行うのか、誰が内を言っているか、何が契機か、何に繋がるのか
2	セルフコンテキスト	静的	その人に関する文脈	その人が抱えている制約、立場や役割、生活、信条、意図や企み
3	ツールコンテキスト	静的	そのタスク管理ツールが持つ文脈	ツールのコンセプト、デザインや雰囲気とその理由、ツール開発者の思想、用語Xをツール上ではどう捉えているか
4	トリガーコンテキスト	静的	そのタスクを実行できる状況	人、場所、時間帯、道具
5	ロードコンテキスト	動的	そのタスクを行うときに必要な「頭の中に入れておくこと」	コンテキストスイッチングで切り替えるものがこれにあたる

1〜3 のタスク、セルフ、ツールはわかりやすいと思います。それぞれタスク、人、ツールに関する文脈を指しています。

4 のトリガーについては、コンテナの章で述べたコンテキストタグのことです。たとえば「本を読む」タスクに「電車」タグをつけておくと、「電車に乗ったときに行えるタスク」を探したいときに電車タグで検索してヒットさせることができます。これは「本を読む」タスクに「電車内なら実行できますよ」という状況的性質をつけているに等しいです。**トリガー（Trigger）** とは契機という意味で、タスク管理においては「始めるきっかけ」です。つまりトリガーコンテキストとは、タスクの「それはいつ始めるのか」「いつなら始められるのか」を指しています。

5 のロードについては、いわゆるコンテキストスイッチングの話です。異なる仕事に切り替えたり、休んでいた状態から仕事を始めたりする際は、その仕事の文脈を頭に入れる必要がありますが、これはコンテキストスイッチングと呼ばれます。頭を丸ごと切り替えるようなもので、とても疲

れる行為です。特に集中を要する高度な仕事では脳内で相当な情報を処理していますが、割り込みなどで邪魔されるとこれが消えちゃいます。取り戻すのには骨が折れます。このコンテキストスイッチングは、「そのタスクを行うのに必要な諸情報」を頭の中に読み込む、特に読み込みし直すということもでき、その諸情報を**ロードコンテキスト（Load Context）**と呼んでいます。

現場のコンテキストが無い理由

動的なコンテクストの項にて、現場の文脈なるものがあるとの話をしました。一方、上記 5 つのコンテキストではこれを扱っていません。どういうことでしょうか。もちろん意図的です。

現場のコンテキストは、タスク管理におけるコンテキストとしては扱いません。

なぜかというと、現場のコンテキストは慌ただしすぎてタスク管理も何もないからです。現場という刻一刻と変化する状況にひたすら食いついて臨機応変に何とかする――これ以上でもこれ以下でもありません。タスク管理の公理でも述べましたが、タスク管理とは言語化とツールの利用が必ず伴う営みです（※1）。それすら許されないほど慌ただしい状況はそもそも想定していませんし、タスク管理は役に立ちません。

ただし全く役に立たないかというとそうではなくて、自分自身が一歩引いて、タスク管理を行う余裕を確保した上で管理を始めればいいのです。個人でやるなら個人タスク管理ですし、チームやプロジェクトで行う・行わせるなら（本書ではほとんど扱っていませんが）プロジェクトタスク管理になります。いずれにせよ、タスク管理を始めると、現場のコンテキストなるもの結局上記の 5 つのコンテキストに分解されます。あの人がああ言ってる、この人はこう言っているらしい等はセルフコンテキストですし、これとこれをやれと言われたけど本当にやっていいのか、ちょっと状況整理してみよう等はタスクコンテキストです。

- ※
  - 1 公理上は「ツールを使う」と「タスクかどうかは人が決める」しか言っていませんが、ツールを使うためには言語化した情報を入れないといけないので、言語化も事実上必要になります。物タスクなど言語を使わないタスク管理もあるかもしれませんが、筆者は現状知りませんし、言語化しないとタスクの内容がはっきりしなくて管理できないので（インボックス程度の軽い利用を除けば）今後発展することもないと思います――が、もちろん未来なんて誰にもわかりません。どこかの天才が言語化しないタスク管理なるものを開拓する可能性も案外あるかもしれません。

コンテキストのトレードオフ

容易に想像がつきますが、コンテキストを知るのにもコストがかかります。またすでに述べたように文脈は人が持っていることが多く、人から情報を直接的ないしは間接的に集めることもよくありますが、人が相手ゆえに、やり方や程度を間違えると関係が悪化します。悪化すれば以後

引き出せなくなってしまいます。嫌われるとか、ハブられるとか、心を開いてくれないとか警戒されるなど、言わずとも誰でも知っていることです。そういう意味でリスクがあるとも言えます。

コンテキストを知る行為は時間や手間とのトレードオフだけと思いがちですが、実はリスキーでもあるのです。やらなすぎるとコンテキストがわかりませんし、やりすぎてもコストがかかりすぎて行動するコストが残ってなかったり（※1）、リスクを踏んでしまって先に進めなくなったりします。最適なバランスを常に模索せねばなりません。

とはいえ、一般的には「知らなすぎる」「見向きもしない」ことの方が多いですから、まずは意識的に知りに行くくらいが良いでしょう。

一方で、大きな組織や政治の強い世界になってくると、事態もすぐには動きませんし、タイミングというものもありますし、そもそも信頼関係をつくるところから始まるケースも多いので、辛抱強く少しずつ知りに行く慎重さが求められます。筆者はこれが苦手で、よくリスクを負ってしまいます。こう言うと下っ端に限った話に聞こえますが、上層の人が部下や現場からコンテキストを集める場合も同様で、トップダウンでいたずらに推進したところで表面上取り繕われておしまいです。このような慎重さはネガティブ・ケイパビリティと言えるとも思います。

- ※
  - 1 当然ながらコンテキストを知るだけでは意味がなく、知った上でどう動くかを考える・判断する・実際に行動することも大事です。これはそうするための時間やお金が必要とも言えます。新規事業におけるリサーチと同様、コンテキストを知る営みもきりがないものですから、バランスを考えて行わないとすぐに尽きてしまいます。あるいはもっと一般論を言うと、コンテキストを知ることばかりに注視して行動が伴わない人は、ただの「情報通」や「細かいことばかり言う人」で終始してしまいます。ちなみに、コンテキストを知るよりもさっさと行動した方が早い（あるいは行動した方が多くのコンテキストが得られる）シチュエーションもそれなりにあり、よく「行動あるのみ」的な信条を持つ人がいますが、この人達はそのようなシチュエーションで成功体験を重ねています。参考にはなりますが、万能ではありません。やはり、どの程度コンテキストを知りに行くかは、その都度探る他ないのです。

5 つのコンテキスト詳細

5 つのコンテキストについて概略を見てきました。続いて各々の詳細に入ります。

タスクコンテキスト

タスクコンテキストとは、そのタスクが持つ文脈を指します。

文脈の切り口は無数にありますが、5W1H でアプローチしてみましょう。Where はやりづらいの

で割愛します。

- Who
  - そのタスクは誰が必要だと言っているか
  - そのタスクは誰のためになるのか
  - そのタスクをやったこと・やらなかったことで誰に影響するか
- What
  - そのタスクを取り巻く事実は何か、意見は何か（特に事実は何か）
- When
  - そのタスクはいつ行えばいいのか、またそれはなぜか
  - そのタスクを今行う理由は何か、事情は何か
  - そのタスクが最初に定義されたのはいつか
- Why
  - そのタスクはなぜ必要なのか
- How
  - そのタスクはどうやって行うのか、特に手段面で制約があったりしないか

こういったことがわかるのとわからないのとでは全然違います。文脈を尊重すれば賢く最低限の手間で立ち回れる上に、自分としても納得感があるため QoL も落ちません。逆に文脈がわからず淡々とこなすだけではどちらも削がれがちです。**セルフコンテキストを知ると実行のタイミングと効率を最適化でき、あわよくば納得感も得られて精神衛生が良くなります。**

この切り口ですが、他にも無数にあるので、自分なりの切り口をストックしておくと良いでしょう。参考にできそうなヒントをいくつか挙げておきます。

- 5W1H
- 異文化理解力に基づいた「説得」の観点
  - アジア圏は「包括的思考」であり、周囲の誰が何を言っているかを知りに行く傾向
  - 欧州は「原理優先」であり、理論や概念から始める傾向
  - 米国は「応用優先」であり、事実や発言・意見から始める傾向
- 事実と意見の区別
- 人、環境、ツールの三観点
- ツールーロール
  - Tool（ツール）、Rule（ルール）、Role（役割や立場）

セルフコンテキスト

セルフコンテキストとは、その人が持つ文脈を指します。

前の章で調動脈について述べましたが、ここでいう調子と動機はセルフコンテキストの一種です（ちなみに文脈はタスクの文脈でありタスクコンテキストに相当）。他にもその人の事情、信条や信念、先天的または後天的に備わった特性や、ストレングスファインダーやビッグファイブのよう

な診断内容も含まれます――と、書くと割と何でも含んじゃってますが、そのとおりで割と何でも含みますが、一言で言えば**その人がなぜそんな行動をするかという因子** 全般です。

たとえば筆者は「自分の手で進める」ことや「毎日同じ生活リズムにストイックに従う」が好きです。これは私の特性から来るものだったり、キャリアアンカーという診断からも推し量ることができました。これらのコンテキストを持つ私にとって、仕事を毎日定時に収めたり残業を基本的に受け付けないのは当たり前ですし、それはチームや上司にも強く希望し、必要なら議論や衝突も辞さないくらいに自明なことではあるのですが、このコンテキストを知らない他者から見たら意味不明でしょう。

セルフコンテキストを知れば、その人の行動原理や行動特性がわかって自分の精神衛生を保てます。なんでそんなことするんだろう、という疑問や憤りに理由が付くので、（納得はできないかもしれませんが）理解はできるようになるからです。

セルフコンテキストは本当に厄介なもので、ここが阻害されると、人はかんたんに信頼関係を壊したり衝突を辞さなかったりします。こう言うと「いや社会人なんだからもうちょっと協調性を……」と思われるかもしれませんが、そう思える人は単にセルフコンテキストが広くて融通が利きやすいだけです。そういう人でも、たとえば子供の急病で早退が必要になったときに「いや仕事なんだから抜けちゃ困るよ」と言われたらイラッとするし信頼もできなくなるのではないでしょうか。これは育児や家族を大切にする、とのコンテキストがあるからです。

このセルフコンテキストですが、本当に人それぞれであり、何一つとして万人に当てはまる絶対的なものはありません。子供よりも仕事を選ぶ人もいますし、仕事よりも推しのキャラクターの誕生日を選ぶオタクもいれば、仕事よりもその職場にいる特定の人が目的になっている（のでその人がいなくなるとあっさり退職する）等もあります。教員やボランティアなど自分の時間やお金よりも大切な何かで動く人もいれば、他人や社会のことを何とも思わず自分の生活と安寧を重視するが表面上はワークライフバランスの域を演じている人もいますし、同僚や顧客を性的対象や恋愛対象とみなして消費・獲得することに重きを置く人もいます。本当に多様ですし、「事実は小説よりも奇なり」もよく似合うと思います。

もちろん、人のセルフコンテキストの、特に奥深くは通常は隠されるので、かんたんに知ることはできませんが、**セルフコンテキストはある程度推定できます**。家族を大事にする、オタクとして推しを大事にする、自分を殺してでも会社の出世競争にしがみつく、あたりは有名でしょう。別に正確に当てられなくてもいいのです。ろくでなしな言い方になるかもしれませんが、行動原理や特性がわかる程度にラベリングできればそれでオッケーなのです。この言い方が嫌なら、セルフコンテキストの仮説検証と言い換えても構いません。

ツールコンテキスト

ツールコンテキストとは、タスク管理ツールが持つ文脈を指します。もっと言えば**設計思想や世界観と同義です**。

タスク管理はツールを用いて行うものですが、そのツールがそもそも曲者で、そもそも色んなツールがありますし、ツール各々も癖が強いです。単純な用途に割り切るなら別ですが、そもそもタスク管理は本質的に煩雑かつ個人的な営みなので、その辺のツールをテキトーに使うだけで上手くいくほど甘くはありません。タスク管理が定着しない人の半分くらいは、テキトーに使うことから脱せていないからだとさえ思います。それくらいにありふれたアンチパターンです。

では、どうするのかというと、ツールのコンテキストを理解して、尊重しながら使うのです。たとえば、

- そのツールでは「タスク」をどういう概念だと捉えているのか
- どんな戦略やスタンスを想定しているか
- どんな用語を定義しているか
- カスタマイズ性はどの程度か
    - カスタマイズ性が高い ≒ 各自の好みに各自で調整してください
    - カスタマイズ性が低い ≒ 四の五の言わずにツールが想定するとおりに使ってください
- デザインや居心地にはどの程度気を遣っているか

などが考えられます。他にも無数にありますが、ポイントは「なぜこんなつくりになっているのだろう」「なぜこんな概念をわざわざ導入しているのだろう」と疑問を持って、その答えを **自分なりに** 探すことです。自分なりに、と書いたのは、答えが直接示されているとは限らないからです。ドキュメントが充実している場合は書いてあることもありますが、書いてない方が多いでしょう。多かれ少なかれ自分なりに解釈する必要性は出てきます。

おすすめはツールの開発者（作者個人だったりチームだったり）の動向を追いかけることです。ツールには開発者の思想が出ます。ここで思想とは開発者自身のセルフコンテキストというよりは、その開発者が開発する物に込める思い、という意味です。開発者がブログや SNS などで発信していることもあれば、他の開発物という形で間接的に共通した性質を推し量ることもできます。

一方で、組織規模が大きくなってくると、ブランドによる細かい雰囲気や癖を除けば本質的な差はなくなってきます。たとえば Google と Microsoft は、一見するとどちらもタスク管理系サービスのあり方がだいぶ違いますが、片方でできることは他方でもできますし差はありません。どちらを使ってもいいです。しかし、好みの問題がおそらく出てくると思います。加えて「普段 Google を使っているから Google カレンダーだと連携しやすくていい」とか「会社が全社的に Microsoft だから選択肢がない」「もう Outlook ばっか使ってて馴染んでしまった」とかいった事情も絡んでくることが多いです。

この好みや事情が厄介で、惑わされやすいです。チームで仕事する場合や、すでにどちらかをヘビーユースしている場合は致し方ないとして、惑わされないでください。本章は個人タスク管理の話をしているので個人が使う想定で書きますが、タスク管理を上手くやれるかどうかは **ツールがどうつくられているかを踏まえた上で、そのつくりと自分が合うか、あるいは合わせに行けそうかどうか次第です**。この合うかどうかを判断するために、ツールコンテキストを知る必要があるのです。もちろん使い続けてみて肌で学ぶ、でもいいですが、合うとわかるまで続ける

のは苦しいと思います。たとえば３ヶ月間、とりあえず使い続けてみるといったことができるでしょうか。新しい習慣を単にひとつつくるよりもはるかに難しいことです（それでもアスリートタイプの人なら比較的楽でしょうが）。かといって全く試さないのも違います。試すこと自体は必要です。ただ、それと並行してツールコンテキストを調べた方が近道だという話です。「あー、このツールはこういう感じなんだー」とわかればしめたもので、合うなら続ければいいし合わなければやめて次を探せます。そういう意味では人付き合いと同じかもしれません。

と、ここで一つ注目してほしいのですが、ここまでの説明では好みや事情といった要素には考慮していません。重要ではないし、何なら有害ですらあるからです。先ほどテキトーに使うのはアンチパターンだと書きましたが、同様に、**好き嫌いや事情で選ぶのも実はアンチパターンです**。もう一度書きますが、タスク管理は本質的に煩雑かつ個人的な営みで、これに対抗するためには、それなりに煩雑な概念や仕組みが必要になります。タスク管理は難しい営みです。かんたんなようで、かんたんじゃない。舐めてはいけません（※1）。何事にもそれなりに基礎があるように、タスク管理もそうです。なのに、タスク管理は現状体系化されていませんし、そのくせ本質的に個人的でもあるので、それら基礎をどう使うかも自分で探して行けねばならないのです。好き嫌いで感覚的に選ぶとか、事情があるからとりあえずこれを使うとかいったことはテキトーに使っていることと大差ありません。そうではなく、ちゃんとツールの設計思想や世界観と向き合い、自分がそれに合わせられるかを見極めます。**合うのなら嫌いでも使った方がいいですし、合わないのなら好きでも使わない方がいいです**。ツールは実際に自分にとって役に立つかどうかがすべてであり、ここと向き合わず表面的に使うのはおままごとでしかありません。と、かなり厳しいことを書きましたが、そうです。難しいし面倒くさい世界ですよね。だからこそ今まで体系化されていないのですし、筆者もこのような本を書いています。

- ※
  - 1「タスク管理はかんたんだよ」とか「そんな難しくないよ、シンプルでいいよ」と考える人もいるかもしれませんが、参考にできるとは限りません。そういう人はたいてい右記のいずれかです――1:慌ただしい状況に身を置いて振り回されているだけ。2:頭の性能が優れていてツールを使わずともこなせる力がある。3:パートナーや部下や経済力など何らかのパワーでタスク管理を委譲している。4:単に抱えるタスクが少ない過ごし方ができている。逆を言えば、これらに当てはまるのであれば、そのとおりタスク管理はかんたんで済むか、要らないか、（特に1:の場合は）役に立たないでしょう。

トリガーコンテキスト

トリガーコンテキストとは、そのタスクを実行できる状況を指します。

個人タスク管理を真面目に行うと何十何百何千というタスクを扱うことになりますが、当然ながらいきなり全部はこなせないので「とりあえず保存しておいて」「一つずつ潰していく」営みが必要です。このときによく使われるのがフィルタリングで、指定条件を満たしたタスクのみを抽出します。このフィルタリングの条件として重宝するのがトリガーコンテキストです。いくつか例を挙げます。

- 通勤中に行えるタスク
- 自分は家にいるが、パートナーも子供もいないときに行いたいタスク
- 会社の昼休憩中に行えるタスク

こういったタスクをそのときに抽出したい場合、タスクにあらかじめ「通勤中」「家族がいないとき」「昼休憩」といったタグをつけておくのです。そうすると、通勤中に通勤中タグで検索することで「通勤中に行えるタスク」を抽出できます。もちろんツール（携帯性も考えるとスマホ一択でしょう、あるいは頑張ればアナログな手帳でも可能です ※1）にこれらタスクとタグを日頃から入れておくこととツール自体を持ち歩くことが前提です。

と、このような「今の状況で行えるタスクを引っ張ってくる」営みは、別にツールを使わず頭だけでもできるのですが、忘迷怠が発生しがちです。何十ならまだしも、何百何千なんて頭だけでは扱えませんよね。だからこそツールを使うのです。逆を言えば、別にそこまでタスクを抱えてないという人には不要です。

- ※
  - 1 たとえばあな吉手帳術では付箋パッドで付箋をグルーピングできますが、これはトリガーコンテキスト名のエリアをつくってそこに付箋を入れるとも言え、本質的にはトリガーコンテキストの付与と同義です。これに限らず、アナログではタグの概念を実現できないので、名前をつけたエリアを設けてそこに書いたり置いたりすることで代替できます。

オーバーヒート

トリガーコンテキストは、自分一人では到底処理しきれない多数のタスクを抱えている人にこそ重宝します。このような人、特に状態を **オーバーヒート** と呼びます。機械が稼働しすぎて異常に熱くなっているイメージです。このオーバーヒートにはさらに二種類あって、やりたいことが多い **オーバーウォント** と、やるべきことが多い **オーバーマスト** に分かれます。いずれにせよ過熱して壊れそうになっている点は同じです。

オーバーヒートの対処方法はいくつかあります。無難なのは他の人に任せたりお金で解決したりといった「委譲する」アプローチか、自分がこなせるキャパシティ以上を受け付けないとする「捨てる」アプローチのいずれかですが、どちらも難しいことです。できれば苦労しないですよね。そもそも個人的なタスクは人に任せづらいですし、メモの章でも述べましたが特に日本では宗教観の影響で捨てるのが苦手になりがちです。そういうわけで「いや、自分自身でやるしかない」「やりたい」と至るのは自然なことです。でもキャパシティがオーバーしてて全部は入らないし、頭だけでタスク全部覚えて使い分けるのも無理だ……。

この状況を打開できるのがトリガーコンテキストです。タスクというものは、いつ行えるかが限られている場合が多いですから、それを全部ツールに記録しておいて、その状況が来たときにすぐ引き出せるようにしておけばいいのです。これをタスク管理的に表現すると **タスクに「トリガーコンテキスト」を表現したタグをつけておく** になります。つけておけば、いつでも検索・抽出できます。

鍛錬あるのみ

トリガーコンテキストの運用は難しいです。

というのも、日々慌ただしい状況の中で「タスクの登録」「タグの追加」「トリガーコンテキストで検索」といった行為を何度も行わねばならないからです。そもそもツールを手放さず日常的に使っている状態が必須ですし、その上、追加や検索も素早く行えなくてはなりません。これは練習や鍛錬という言葉が似合うくらい、地道に身に着けていかねば至れない境地です。もちろん、この境地に至らなくても行うことは可能ですが、手間暇がかかりすぎて形骸化待ったなしでしょう。追加や検索にいちいち何十秒もかけてられないですよね。10 秒とか、できれば数秒くらいでできるようになりたいのです。

もちろん少しずつ始めることもできます。たとえば「通勤中」タグだけひとまず運用してみよう、くらいならかんたんです。無理なく一つずつタグを増やしていけば自然でしょう。

最適なトリガーコンテキストは人それぞれ

さらに厄介なのが、このトリガーコンテキストを使えばいい、という正解がないことです。たとえば先ほど挙げた「通勤中」は、通勤をしていない人には通じませんし、「家族がいないとき」も独身の人には意味ないです。「昼休憩」も裁量高めで働いている人や夜型の人やそもそも働いていない人には使えないものでしょう。

自分を取り巻くトリガーコンテキストとして何があるかを知らねばなりません。かつ、それを素早く操作に反映できるよう言語化も要ります。「通勤中」と「電車内」と「移動中」は全部違いますし、ツールによっては並び順や色もあるので「1: 通勤中」のように数字を入れて並び順をコントロールしたり、文字色や背景色をどうするかも設定の余地がありますよね。こういうのを全部決めなくてはなりません。というか、いきなり決めるのはまず無理なので、試しながら調整していくことになります。鍛錬あるのみと書きましたが、本当に地道です。

コンテキスト四天王

では自分のトリガーコンテキストを自分でいきなり言語化してタグに落とし込めるかというと、これも難しいです。幸いなことに、先人が色々な事例を教えてくれています。筆者はこれをさらにまとめて、よく使われるトリガーコンテキストを **コンテキスト四天王** と呼んでいます。以下の四つです。

- 人 Person
  - そのタスクを行うのに必要な人
  - そのタスクを行う際に、いると邪魔な人
- 場所 Place
  - そのタスクを行える場所
- 時間帯 Section

    - そのタスクを行える時間帯
- 道具 Tool
    - そのタスクを行うのに必要な道具

まずは人、場所、時間帯の三つを意識すると良いでしょう。ここまでの例で言えば、「通勤中」は場所または時間帯、「家族がいないとき」は人、「昼休憩」は時間帯です。私たちは通常、人や場所や時間帯によってできることが限られていますので、まずはこの三つについてトリガーコンテキストを考えるだけでも十分です。

道具については、実はあまり出番がありません。実質的に「道具がある場所にいるかどうか」や「道具を使える時間帯であるかどうか」に帰着されることが多いからです。「パソコンが必須」タグを使うよりも「自宅」とか「出勤中」の方がわかりやすく使いやすいと思います。

ロードコンテキスト

ロードコンテキストとは、頭の中に入れておく文脈を指します。

私たちは何かを行うとき、その過程や状況に関する諸々（ロードコンテキスト）を頭の中に入れます。特に一度の行動で終わらない仕事については、またあとで「続きから」行うことになりますが、その続きが何かという情報は通常、頭の中にあります。「ああ、ここまでやってたよな」とか「とりあえずこれとこれをこうするつもりでいるんだけど」など色々考えているはずです。逆にその辺を覚えていなかったり、思い出せなかったりすると腰が上がりません。あるいは前回やったことを無視して、非効率的なことをしてしまうかもしれません。思い出せるまで粘るのも手ですが、それはそれで大変ですよね。

コンテキストスイッチング（Context Switching）という言葉を聞いたことがある人は多いでしょう。これは頭の中に入れたロードコンテキストを切り替える（スイッチングする）という意味です。ある管理職の人がいたとして、幹部への報告、部下を集めた定例会議、部下各々との 1on1 ミーティング、家庭でのパートナーとのやり取り、子供とのやり取りなどを抱えているとして、これだけでも 6 個のロードコンテキストがあります。部下が 5 人いたら、一人一人違うはずですから 1on1 ミーティングだけでも 5 人分のロードコンテキストがありますよね。仕事が変わる度に切り替えねばなりません。部下 A さんとのミーティングなのに B さん用の文脈が頭にあっても使い物にならないし、失礼になってしまう恐れもあります。すでに述べたとおり、人が相手だと文脈を理解せずコミュニケーションすると見限られてしまうリスクもあります――と、これは人を相手にする生き方の例ですが、人を相手にしない場面でも例外ではなく、たとえば趣味にせよ仕事にせよクリエイターは手掛けている作品に応じてロードコンテキストが変わりますし、ゲームにせよスポーツにせよ対戦の世界に身を置く人についても、ステージやチームごとに変わるはずです。

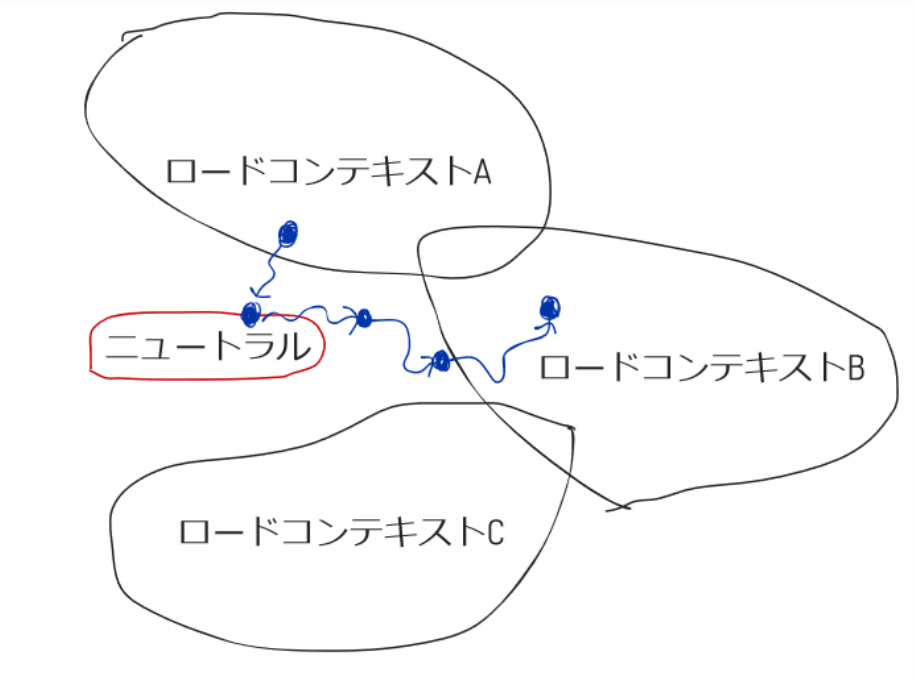
言うまでもなくコンテキストスイッチングは重要です。しかし前提として**コンテキストスイッチングはとても疲れます**。重要なのに疲れるというシビアな営みです。上手くやれるに越したことはないでしょう。単純な話、疲れにくくなればよりたくさんの仕事や行動をこなせるようになりますし、最低限の切り替えができるようになるだけでも（文脈を考慮できなかったことによる）失望や見

限りを食らうリスクを減らせます。何より**切り替えの腰が上がるようになると、行動力が上がります**。体力やバイタリティという話ではなく、私たちが何かと怠けがちなのは、コンテキストスイッチングが疲れるからだという面もあるのです。スイッチングを楽々行えるようになれば、その分の怠惰は減らせます。実際はモチベーションの問題もあってそう単純ではないのですが、少なくとも「スイッチングの疲労」という邪魔者を一つ薄めることができるのです。

では、どうすればもっと上手くやれるでしょうか。

コンテキストスイッチング・モデル

まずはコンテキストスイッチングがどう行われているかのイメージを理解したいです。これはあくまでイメージであって、脳科学的にこのようなメカニズムが働いている等を論じているわけではないことに注意してください。直感的に理解しやすくするために筆者が考えたモデルです。



load_context

上の図は、ロードコンテキスト A から B にスイッチングする際のイメージです。ポイントは二点で、**領域** をフラフラ歩いていることと、**ニュートラルという回復地帯** を経由していることです。

ここから以下のような本質を導けます。原則という形で名前もつけておきましょう。

- 地道の原則
  - 歩いているので、移動自体にそれなりの時間がかかる
- 回復の原則
  - いったん回復地帯を経由するべきである
- 面の原則

    - 領域であるため広がりや重なりがある、少なくともゼロイチのような単純な状態ではない
- 等価交換の原則
    - ロードコンテキスト領域の奥底に行けばどっぷり浸かれるが、行くのにも戻って来るのにもその分時間がかかる
    - （等価交換とはトレードオフの意）

コンテキストスイッチングをどう行うかは、このモデルに則った形で設計すると馴染みやすいのです。モデルを踏まえると、おおよその戦略も見えてきます。いくつか見ていきましょう。

休憩の時間をしっかり取る

まずは基本中の基本ですが、ニュートラル（回復地帯）を経由するための **休憩の時間をしっかり取りましょう。**

休憩の強度としては、筆者の体感ですが、睡眠 >> 昼寝 > 瞑想やぼーっとすること >> ネットサーフィンや談話 > 家事や散歩や手慣れた作業、くらいだと思います。睡眠はニュートラル領域の奥底に行くイメージだと考えてください。後ろの方のネットサーフィンや散歩は、ニュートラルには入っていないが、かなり近づいているようなイメージです。近づくだけでも回復は得られますし、別の領域に行く際はむしろやりやすいです。

一つ注意したいのですが、だからといってネットサーフィンや散歩ばかりしてればいいかというと、そうでもないのです。ネットサーフィンでは注意資源（認知資源）、散歩では物理的な体力も使います。これらが減りすぎても動けなくなるので、やりすぎると本末転倒です。あくまでもコンテキストスイッチングの観点においてはニュートラルに位置するものだ、という話にすぎません。そういう意味では、休憩手段を分散させるのは手です。ネットサーフィンしか使えないよりも、ネットサーフィンと散歩と家事を全部使えた方が、休憩の余地は広がります。前者だと注意資源を消耗してしまって、早々にどこかでバテてしまうでしょう。

これら休憩は意識的に取りたいです。筆者としては、1時間仕事Xをする → 1時間ネットサーフィン → 1時間仕事Yをする → 1時間散歩、のように同じ時間だけ休憩を差し込んでも全然問題ないと思います。一見すると休憩にだいぶ時間を割いてそうですが、コンテキストスイッチングは疲れますし、スイッチングが不十分のまま仕事してもかえって雑になったり無駄に詰まったりして進行が遅くなりがちです。徹夜するよりもしっかり睡眠を取った方が捗る、くらいならわかりやすいが、これはコンテキストスイッチング全般でもそうだと考えてください。おそらく実際に試して、生産性など成果を測定してみるのが良いでしょう。休憩をこまめに、大胆に入れた方が出ると思います。

休憩の手段は多岐に渡りますが、本章では解説しません。筆者としては「自分に合った手段と時間を取り入れる」と「一つの手段に頼りすぎると（ネットサーフィンで注意資源を使い切るように）別の資源が死んでしまうので色んな手段を使う」の二点を押さえて、あとは自分で模索すればいいかなと思っています。その他詳細は、この手の話題を扱った参考文献に譲ります:

19、22

スプリント vs マラソン

コンテキストスイッチングのあり方として、スプリントが合う人とマラソンが合う人がいると感じます。

スプリントとはオンオフをしっかり区別することです。やるときはしっかりやる、ロードコンテキスト領域の奥底にまで一気に突っ込み、しばらく滞在してあくせくした後、帰りも一気に駆け抜けるイメージ。一方でマラソンとは、マイペースにゆっくりと常に移動し続けることです。

スプリントの場合、たとえば仕事Aを3時間して、その後2〜3時間くらい休憩してから、仕事Bにまた3時間くらいどっぷりと浸かる――そんな過ごし方になります。途中の仔細は見向きもしません。仕事Aの最中に用事を思い出したから済ませるかー、みたいな脱線はせず、ただただ気が済むか落ち着くまで仕事Aにぶっこんでいます。

マラソンの場合は、だらだらとながら作業しながらAをこなしつつ、Bをこなしつつ、のような中途半端な過ごし方になります。とはいえ今はAに注力している、今は休憩寄り、今はBに寄っている、などウェイトの傾向は出ます。ただスプリントのようにAにだけぶっこむ！というほど特化はしていません。ながらやマルチタスクといった言葉がよく似合います。また常に歩いているようなものなので、足を止めるような大きめの休憩は取りません。たとえば昼休憩中や散歩中でもスマホを片手に何かを読んだり聴いたりするでしょう。

どちらのスタイルが合っているかは人それぞれですので、ご自身のスタイルを意識すると良いでしょう。スプリントでしたら割り込みや誘惑は積極的に断つべきですし、マラソンであれば逆に下手に縛らず開放的で賑やかな環境の方が過ごしやすいです。一つの目安は散歩で、30分から1時間くらいの散歩を毎日行うとして、散歩中に他のことを行いたいかどうか、です。あるいは行ってしまうかどうか、でも構いません。数日くらいだと演技も可能なので、数ヶ月や一年など長期的に続ける場合を考えてみてください。行う場合はマラソンタイプだと思います。

ちなみに現代ではマラソンが主流です。スマホやSNSの影響は言わずもがなですし、仕事も長時間拘束されがちで、休日や休憩時間も未だに偏っていて、で待ち時間に対してもシビアになりがちですよね。なので性格次第というより、生活次第だとは思っています。実際、筆者自身も昔は明らかにマラソン派でしたが、現在はスプリント派です。そのためにミニマリズムなど現代の潮流に抗う行動もかなりしています（たとえばスマホを持っていません）。もう一つ、仕事の現場でもマラソン派が主流となっていて、慌ただしい状況下でマルチタスクを求められがちです。スプリント派は、よほど実力があればどうとでもなりますが、そうでもなければ中々厳しい社会だと感じます――と、マラソンが主流であることを書いたのは、**主流に惑わされず自分のタイプを判断してほしいから**です。

バッファを取る

バッファとはイレギュラーに備えて取っておく余裕時間ですが、モデルで示したとおり、コンテキストスイッチングは移動のようなものであり、それなりに時間がかかります。**移動時間は見越してお**

いた方が何かと良いです。

特に色んな仕事に切り替えないといけない人は、10 分でも 5 分でもいいので、余分に取ることを意識してみてほしいです。前述でいうマラソン派の人は、間を置かずにすぐ切り替えがちですが、これは疲れます。移動でもそうですよね、たとえば乗り換えでも次の電車の発車時刻まであと数秒しかなくて駆け込み気味に乗車するのと、5 分くらい待ち時間があってゆっくり待ってから乗車するのとでは違います。このイメージです。管理職の方も、会議の予定が連続で詰まっている場合が多いかと思いますが、数分くらい休憩があるだけでもだいぶ違うでしょう。

コンテキストスイッチング一つ一つの負荷は大して大きくありませんが、これも重なると効きます。意識的にバッファを取っておくことで、この重なりの暴力もある程度食い止められます。

土地勘を養う

土地勘とはたとえですが、各ロードコンテキストの位置関係と歩きづらさはぜひとも知っておきたいところです。

位置関係とは、文脈的に似ていることの度合いです。たとえばブログ記事 A を書くことと、同じブログの別の記事 B を書くことはタスクとして近いです。文脈が似ているので、コンテキストスイッチングも比較的しやすいです。もちろん記事内容によってはスイッチングが大変なケースもありますが、少なくともブログを書くという環境面では同じです。これは内容観点では遠いが、環境観点では近いと言えます。距離的に近い（似ている）観点があることこそが重要であり、このような観点が多ければ多いほどスイッチングはしやすくなります。位置関係の近い観点が一つ存在することを ニア（near） と呼ばせてください。この場合、環境観点がニアです。ニアを最低一つ、できれば複数探したいのです。

さて、どんな観点があるかですが、ここは各々しっくり来るものを探してください。参考までに、筆者がよく使う観点を挙げておきます。

- 内容感（仮にカテゴリーに分けるとしたらどのカテゴリーに属するか）
- 締切感（寝かせる余裕があるかどうかなど）
- 資料感（誰かに伝えるための手間が要るのか、それとも自分だけがわかればいいのか）
- 審査感（誰かにチェックしてもらう必要性の度合い）
- 契機感（トリガーコンテキスト感。同じ状況で実行できそうか）

小難しく書いてますが、要は内容感は「同じ食レポ記事なら記事 A から記事 B にもスイッチングしやすいよね」、締切感は「今日締切のものはまとめて行った方がやりやすいよね」、資料感は「人に伝える必要がなく手元で検討すればいいものはまとめて行った方が捗りそうだよね」といったことを言っています。締切感が一番わかりやすいでしょうか。今日が締切のタスクが複数あったとしても、あまりスイッチングの負担に煩わされずに対処できると思います（疲れはしますが）。これは締切という観点において、コンテキストの位置関係つまりは距離が近いからです。近いからスイッチングの負担が小さくて済んでいるのです。「締切感」はよく知られた観点の一つと言えるでしょう。ニアを探すのもかんたんです。言葉にするまでもなく、今日が 2024/07/03

なら、各タスクの締切が 2024/07/03（今日中）や 2024/07/04（明日中）であることを確認すればいいだけだからです。

このように観点をたくさん知り、自分が抱えるタスク達の位置関係を把握してニアを探せば、スイッチングを行いやすくなります。頭だけで把握できない場合は、書いたりプロットしたりもしても良いでしょう。Miroなど無限キャンバスが役に立ちます。

もう一つ、歩きづらさについては、端的に言えばどれだけ消耗しやすいかです。一番わかりやすいのがスポーツにおける本番でしょう。陸上でも野球でも何でもいいですが、本番を一つ行うためには前後のウォームアップやクールダウンも含めて、かなりの手間がかかります。一日複数回本番を行うことは通常ないはずです。頭脳労働についても、アイデア出しなど正解含む制約の無い状態で発散や収束を行う営みは非常に疲れるもので、一日数時間とできるものではありません。これらは非常に歩きづらい山岳地帯を登り降りしているようなもので、移動自体にかなりの負担がかかります。一方、身体の各部位を鍛えたり技術的な練習をしたりすることや、スキルを発揮して何らかの作業を行うことなどは（習熟具合にもよりますが慣れてしまえば）さほど疲れません。熱心な者やプロは一日中、下手すれば四六時中活動していられますが、これは歩きやすい平地を移動しているようなものです。

何のタスクがどれだけ歩きづらいかは、そのタスクの性質と、タスクの実行者の能力で決まります。たとえば現在 40 万文字、第 3 巻相当まで書けている小説の続きを書くタスクがあるとして、これは歩きづらいでしょうか。答えは人によります。まず小説を書かない大多数の一般人にとっては、まともに歩けないほど険しいでしょう。書ける人については、頭の中で組み立てることができてそれを出力・微調整するだけで済む才能の持ち主（個人的には小説家に必要な才能と思います）であれば歩きやすいはずです。逆に、筆者もそうですが、頭の中では完結できず、書いたものと脳内を絶えず行き来しながら頑張って整合性を合わせつつ、より面白い展開も粘り強く探しに行くようなタイプですと歩きづらいです。実際、筆者はいくつかの Web 小説を書いてきましたが、歩きづらすぎて一日数時間くらいしか書けません。プロの小説家や Web 小説でも毎日更新できるような人にとっては平地を歩くようなものですが、筆者にとって山岳地帯を登るようなものです。

この歩きづらさの把握は非常に重要です。歩きづらいかどうかで適切な向き合い方が変わるからです。まず **歩きづらいタスクは一日の中で実行できるタイミングが限られます**。筆者にとっての小説がまさにそうで、起床後のフレッシュな頭で数時間もつかどうか、という程度です。仕事後の疲れた頭では書けません。なので筆者が小説を書きたいのなら、なるべく起床後早めに書く必要があるのです。会社員で仕事があることも考えると、早起きをして数時間の時間をつくる必要もありますね。そのためには生活を早寝早起きにシフトせねばなりません――と、ここまでやらないといけなくなります。逆を言えば、歩きづらさを把握しているからこそ、自信を持ってこうして行動に移せているわけですし、ここまでしない限りは書けません。散歩時の軽装で本格的な登山ができないのと同じです。登山に必要な準備は、やるしかないのです。次に **歩きやすいタスクは一日中どこでもできます**。始めれば進みますので、始められるかどうかが争点となります。カフェで仕事をしたり、とりあえずたくさん仕事を詰め込んで忙しくしたりする人が多いのもそのためですね。自律的に始めるのが難しいので、環境や集団の力に頼るわけです。あま

りに当たり前すぎて自覚がないこともよくあります。

このように歩きづらさを把握できれば、どう取り組んでいくかも考えやすくなります。ゲームでたとえるとマッピングのようなものです。マップを自分でつくるのは大変ですが、一度つくってしまえば、以後はそのつくったマップを見て最適な行動が取れるようになります。

タスク管理のスタンスとの関係

スタンスの章でもコンテキストを重視したスタンスをいくつか取り上げています。

コンテキストン は、タスクコンテキストを重視するスタンスです。ちなみにセルフコンテキストは重視しません（してもよい）。セルフコンテキストは、どちらかと言えばマネージャーが部下と上手く付き合ったり、社内政治で勝つためにキーマンを理解して対策を打ったりといった場面で重視されます。しかし現代は多様性の時代でもあり、他の人のセルフコンテキストを尊重できない人は今後淘汰されていく（少なくとも万能ではなく活躍の機会が限定される）と思います。

ミキサーは、ロードコンテキストを重視する――つまりはコンテキストスイッチングを抑えるスタンスです。位置関係の近いタスクを把握して、まとめて処理しようとします。

まとめ

- コンテキストとは文脈のこと
- コンテキストを知ることの重要性
  - 1 踏まえないと通じないし、下手すれば失敗するから
  - 2 何度も挑戦できるほどチャンスは多くないから（特に対人関係）
  - 1+2、つまり少ないチャンスで確実にものにするために、コンテキストを知っておいた方が良いから
- コンテキストには動的と静的がある
- タスク管理においては、コンテキストは 5 つある
  - 5 つ
    - タスクに紐づくタスクコンテキスト
    - 人に紐づくセルフコンテキスト
    - 道具に紐づくツールコンテキスト
    - そのタスクを実行できる状況、を意味するトリガーコンテキスト
    - コンテキストスイッチングにて切り替えるロードコンテキスト
  - タスクとセルフとツールについては、
    - 知りに行くのにもコストがかかるためトレードオフである
    - どんな切り口で知りに行くかは人それぞれ、自分なりの手札を揃えておくと便利
  - トリガーとロードについては、

- 最適なあり方は人それぞれなので自分で模索していきたい
  - 筆者の例も示しているが、これは筆者の例であり、おそらく参考にならない
- コンテキスト四天王とコンテキストスイッチングモデルは便利

=== Chapter-19 割り込みと脱線 ===

タスク管理が上手くいかない要因の一つとして、タスクをスムーズに遂行できてないことが挙げられます。なぜスムーズにできないかにも色々ありますが、ワープ（割り込みと脱線）が絡んでいることがよくあります。本章ではこのワープと向き合っていきます。

ワープの概要

割り込み

外部から干渉されて逸れることを **割り込み** と呼びます。

外部と書きましたが、**割り込みは「人」からもたらされます**。たとえば声を掛けられたり、接近されて視線や空気などで圧を出されたりします。割り込まれると、人として無視できない（少なくともしづらい）状況になり、向き合わざるを得なくなります。

脱線

自分から寄り道に逸れることを **脱線** と呼びます。

脱線は「情報」という誘惑に負けることで発生します。情報とは場の雰囲気だったり、周囲にある道具や設備だったり、スマホやPCで見ているコンテンツだったり、あるいは単に頭に浮かんだ何かだったりと様々ですが、それらに負けてそっちに行ってしまうと脱線となります。

割り込みと脱線の境界は曖昧です。たとえばチャットで DM が飛んできたと通知された場合、これが割り込みなのか脱線なのかは人によって違います。対面で直接声を掛けられた、くらいに捉える人にとっては割り込みでしょうし、逆に「あとで処理すればいい」と捉える人にとっては情報にすぎません（この情報に負けて返事したりすると「脱線した」となります）。もう一つ、極端なことを言えば、人から話しかけられても、それを無視できる精神性を持つ人であれば、割り込みの概念はほとんどないことになります。それでも大きな声や肩揺すりなど身体的に干渉されればさすがに気づくでしょう。一つの目安としては、**ひとまずの判断や返事でさえも無視できるなら脱線、無視できないなら割り込み** です。

ワープ

割り込みと脱線を総称して **ワープ（Warp）** と呼びます。あるタスクに集中しているのに、別の何かに飛ばされてしまうというニュアンスです。外から干渉されて飛ばされるのが割り込みで、自分から飛んでしまうのが脱線です。

ワープは良い時と悪い時がある

ワープは必ずしも悪いとは限りません。たとえば何らかの緊急対応だったり、熟練者からの助言や指摘だったりは割り込まれた方が上手くいくでしょうし、そうでなくとも漠然とやる気が出てないときに同僚が割り込んでくれて、そこで話をしたり別の仕事をしたりするうちに気分が晴れて助かった、のようなこともあります。脱線についても、漠然と暇を潰したい場合は自然に任せてあちこち脱線した方が退屈しないですし、アイデア出しや議論の発散段階の文脈ではむしろ歓迎するものでしょう。

とはいえ、通常は何らかのタスクをこなしていて、邪魔が入った分だけ損をする場面が多いと思います。このような場面を **攻略** と呼びます。攻略中にワープをされると困るのです。

一つ注意したいのは、何が攻略になるかは人それぞれだという点です。特に攻略に頼るかどうかがタイプ（行動特性――メンタルモデルと呼べるレベルかもしれません）として分かれており、たとえ同じタスクをこなす場合でも A さんは攻略としてこなすが、B さんは攻略のつもりではいない、といったことがありえます。A さんタイプを攻略派、B さんタイプを防衛派と呼ぶことにしましょう。B さんのように防衛派が多数派な職場だと、おそらく割り込みはよく起きます。攻略派の A さんとしてはやりづらいでしょう。逆に A さんのように攻略派が多数派な職場だと、割り込みはあまりないと思いますが、防衛派の B さんは寂しさを感じるでしょう、また割り込まれることで色々活性化してバリバリ働くことを前提としていますから（防衛派が多いとその機会が少なくて）普段の生産性も出ないかもしれません。リモートワークが始まってそうなってしまった人も多いのではないでしょうか。

なぜワープは良くないか

攻略中は邪魔が入った分だけ損をすると書きましたが、具体的にどんな弊害があるでしょうか。二つあります。

一つ目はパフォーマンスが阻害されます。1 時間フルで費やせるのと、割り込みが何回かあって 35 分しか費やせなかったのとでは、前者の方が単純計算でパフォーマンスが良いですよね。というより前者が標準のパフォーマンスであり、後者の方が標準からマイナスのパフォーマンスだと言えるわけです。

二つ目は、コンテキストスイッチングが発生するからです。前章にてロードコンテキストについて取

り上げましたが、タスクを行う際は「そのタスクをどこまでどのように行ってきたか」という類の文脈（ロードコンテキスト）を頭に入れる必要があります。ワープが起きると、ここがぐちゃぐちゃになってしまうのです。頭の性能が高い人であれば上手く保持しながら対処できる、いわゆるマルチタスクも可能ですが、乱されていることに変わりはありません。ここは思っている以上に負荷がかかってまして、今行っているタスク → ワープ先のタスク → 今行っているタスク、と複数回の切り替えが必要になります。これは思いの外、疲れます。ひどいと切り替えが終わる前に別のワープが発生してしまうことさえあります。まさにてんてこまいです。

マイナスパフォーマンスとコンテキストスイッチングの阻害――どちらが深刻かは状況次第ですが、私たちは人間であり、人間である以上認知資源や認知能力にも限界がありますので、通常は後者の方が深刻です。ワープが頻発してコンテキストスイッチングがたくさん発生していると、あっという間に疲れてしまいます。各タスクの効率や質も落ちますし、余裕もないので改善したり新しいことを考えたりといったこともできず、忙殺されるだけの日々になります。長時間働いてる割には生産性が低かったり、令和の時代なのにやり方や考え方がアップデートされていなかったりするのも、筆者はこの点にあると思っています。ワープを舐めすぎなのです。人間は認知的な体力をさほど持たない脆弱な生き物にすぎません。だからこそ、消耗を抑えるためにワープと向き合う必要があるのです。

もちろん、これは生産性やアップデートが絶対だと言っているわけではありません。その必要がないなら別にワープを放置してても構いません。しかし、ビジネスではそうもいかないでしょうし、私生活であっても同じことを繰り返すと飽きてしまいます。時代もただでさえ VUCA――先が読めない上にすぐ変化しますので、前進や変化に充てる、もっと言えば充てるための余裕の捻出は出来た方が良いと思います。その捻出の筆頭が、ワープの対処と防止です。

SWAP マトリックス

ワープをどう処理するかはマトリックスで整理できます。

	Strong **強い後回し**	Weak **弱い後回し**
Active **割り込み**	1	2
Passive **脱線**	3	4

これを **SWAP マトリックス** と呼びます。

後回しという言葉が出ていますが、後回しとは、タスクAの最中にタスクBにワープする状況のときに、どちらか片方を後回しにしなければならないということです。A を後回しにする場合を **弱い後回し** と呼びます。自分が元々やっていた A を後回しにしているわけで、いわば負けているため弱いと表現しています。逆に B を後回しにする場合を **強い後回し** と呼びます。割り込みにせよ、脱線にせよ、後からやってきた B に負けずに A を死守する（やってきた B を後回しにしている）ため強いと表現しています。

つまりSWAPマトリックスは、ワープを構成する「割り込み」と「脱線」それぞれについて、強い場合と弱い場合の二通りに分岐しています。計4通りのパターンがあると言っています。

対処と防止

各パターンの解説に入る前に、ワープの対処と防止の違いについて軽く触れておきます。

対処とは、ワープが起きたときに対処を行うことを意味します。具体的には今やっているAを続けるか、ワープ先のBを始めるか、の二択です。前者が強い後回しで、後者が弱い後回しでしたね。つまり対処とは後回しに他なりません。AかBか、**必ずどちらか片方は後回しになります**。「両方並行してマルチタスクできるよ」も厳密には不可能ではありませんが、SWAPマトリックスでは扱いません。必ずどちらかを選び、どちらかを（少なくとも一時的には）捨てます。

防止とは、そもそもワープが起きないように、あるいは起きたときに自動的に即座に反応できるように盤外戦をする――つまりは備えておくことです。たとえば割り込みに対して強い後回しを行いたい（割り込んできたタスクBを後回しにしたい＝割り込まれたくない）場合、そもそも割り込まれないような何らかの働きかけや仕掛けをします。割り込んでくる人と距離を置く必要もあるかもしれません。逆に割り込みに対して弱い後回しを行いたい（割り込んできたタスクBこそ優先したい）場合、遠慮せずいいから何かあったらすぐに報告してくれ、といった働きかけをして頻度を上げると思います。あるいはアラームが鳴るよう仕込んだり、しきい値を下げて鳴りやすくしたりなどもありえます。

以降からは各パターンの解説に入ります。対処と防止の両方を取り上げます。

1 Active Strong

割り込みに対して強い後回しを行うのが **Active Strong** です。タスクAの最中にタスクBが割り込んできても、「いやAを続けますんで」とBを後回しにするわけですね。

対処と防止についてですが、Active Strongにおける対処とは強い後回しをできるだけ行えるようにすることです。また防止とはそもそも割り込みが起きないようにすることです。

Active Strongは、対処よりも防止を重視するべきものです。というのも、対処として行えることは限られているからです。かんたんなのは相手を不機嫌にしてでも断ったり待たせたりすることですが、言うまでもなく対人関係に差し支えます。そもそも割り込みはたいてい相手の方がパワーが強いか、少なくとも対等なので断りづらいです。心理的精神的にもそうですし、仮に行動に移したとしても相手が許してくれません。そういうわけで、対処したいのであれば、相手が納得する言い分を即席で提示することになります。言い分をあらかじめ準備しておけば済むというほど単純ではなく、周囲の空気や相手の非言語情報、その他文脈も踏まえながら、相手が納得する演出をしないといけません。コミュニケーション術とか交渉術とか人誑しのテクニックといった話になります。本書では扱いませんし、筆者もてんで苦手です。

防止については、主に以下が存在します。

- アタック（Attack）
  - 割り込み元に直接働きかける
  - □ 成功すれば割り込みを激減できる
  - □ 対人的な議論や主張が必要なことが多く、荒れやすい
- エスケープ（Escape）
  - 割り込まれないよう環境（場所や所属など）を変える
  - □ やらないよりはマシ
  - □ 一時しのぎにしかならない
    - 一時的な対処だけではもたない場合は、永続的な対処（異動・転職・引っ越し・逃亡など）が必要
- ガード（Guard）
  - 割り込んでくる人の相手をする専任者・窓口を設ける
  - □ 割り込みに煩わされなくなる
  - □ アタックやエスケープといった極端な対処ではなく、比較的無難に導入しやすい
  - □ 番人の調達や設定が難しく、番人の負担も大きい
  - □ 重要な情報が番人によって加工されてしまうことがある（又聞きになる）

典型的にはまずエスケープにて距離を置き、それでもダメならアタックにて直接働きかける、になるでしょう。アタックはたいてい荒れますが、普段の会話や会議の中で少しずつ提案するなど温和に進めることもできます（控えめなアタック）。ただ、筆者の感覚ですと、割り込んでくる人の割り込みが自然となくなることはほぼなくて、たいてい何らかの性格や事情が絡んでいます（攻略派・防衛派はメンタルモデルのレベルかも、とはすでに述べました）ので、控えめなアタックでは一向に改善されない気がします。アタックするなら玉砕覚悟で強めに、が原則です。アタックで最も多い成功パターンは、難しい人認定されることです。こいつ面倒くさいから相手にするのやめよ、とみなされたら勝ちで、以後割り込みは激減します。もちろん立場や待遇など失うものも多いです。それが嫌なら、味方を増やした上でアタックを仕掛けることで大義的に勝ち取る手もありますが、そもそも味方を集められるなら苦労はしません。このあたりも対処と同様、本書の範囲を越えますので割愛します。

エスケープもアタックも難しい場合は、ガードが使えるかもしれません。特にビジネスだと役割や窓口を使い分けることはよくあるので意外と通じやすい（通じる言い分や体裁を整えやすい）です。問題はガードを担ってくれる人材を募ったり体制をつくったりするだけのリソース、たとえば時間とお金があるかです。割り込んでくる側がひどい場合は、チームの誰かが壁役になることも検討したいところです。特に壁役は専任させるのが望ましいでしょう。壁役というと、通常の仕事や用事はそのままで、さらに壁役も背負わせるとのパターンが多いですが、それでは当人は大変です。なんでせめて今の仕事は免除、あるいは少なくとも軽減して、その代わり壁役に専念してください、とするのです。その方が壁役当人も取り組みやすく、中途半端に対応してぐだぐだになる展開も回避しやすいと思います。ですがこれも結局、割り込みへの防止にそこまで投資できるかという時間とお金の話に帰着されます。そういう意味で、**ガードとは投資** です。ガードするためにそこまで費やせるかが問われます。個人にせよ、チームにせよ、判断する者の手腕が

もろに問われるものです。

2 Active Weak

割り込みに対して弱い後回しを行うのが **Active Weak** です。タスクAの最中にタスクBが割り込んできたら「わかりました」とBを優先します――つまり元々やっていたAをいったん後回しにします。

対処と防止ですが、Active Weakにおける対処とは弱い後回しをできるだけ行えるようにすることです。また防止とは割り込みをできるだけ増やしたり、素早くしたり、来た時に判断無しに即座に対応できるようにしたりすることです。少しわかりづらいですが、Active Weakは割り込んできたタスクBを重視したい場合に重宝します。Bを確実かつ素早くこなすのが肝心ですので、Bが高頻度に（できれば即座に）届くことと、届いたときにすぐ行動に起こせることができれば良いのです。防止というと割り込みを防ぐものと考えがちですが、Active Weakでは割り込んでくる方こそが重要なので、むしろ割り込みを起こしやすくする行動こそが（把握や判断が遅れることに対する）防止になるのです。

Active Weakの対処

対処については、集中の解除とコンテキストスイッチングコストの最小化の二点が必要です。

まず集中の解除とは、タスクBが割り込んできたときにその事に気付きたいという話です。タスクAに集中していると、多少割り込まれたとしても気付きませんよね。それでは困るので、Aへの集中を解かないといけません。やり方は原始的で、割り込んでくる人に「もし私が気付かなかったらこのやり方で気付かせてくれ」と周知させる――これだけです。たとえば「気付かなかったら肩を揺らしてくれ」とか「デスクに呼び鈴ボタンを置いてるからこれを押してくれ」など。また、そもそも割り込んでくる人が割り込みづらいシチュエーションも多い（部下が“忙しい上司”に割り込む場合等）ので、そういうときでも割り込める手段を用意しておくとなお良いです。チャットでDMを送ってくれとか、緊急なら会議中でも何でもいいから入ってきてくれて構わないと言っておくなどですね。

次にコンテキストスイッチングコスト（長いので以下スイッチングコスト）の最小化ですが、これは前章で取り上げたコンテキストスイッチングの話です。Active Weakを行うと、以下の流れになるわけですが、

- 1: タスクAをやっている
- 2: タスクBが割り込んできた → タスクBをやる
- 3: タスクBが終わったので、タスクAに戻る

このとき2の部分と3の部分で二回ほどコンテキストスイッチングが起きています。前者ではAからBに頭を切り替え、Bが終わった後、後者ではBからAに切り替えてますよね。前章でも述べましたが、この頭の切り替え、コンテキストスイッチングなる行為は非常に疲れます。疲れる

と切り替えに時間がかかるのでタスクBの遂行（あるいはBが終わった後のAの再開）のスピードや品質も落ちますし、ひどいと切り替えすら諦めて文脈を頭に入れないままテキトーにこなしたりします。このとおり、ろくなことにならないので拒絶反応も起きがちで、割り込まれたときの判断力も落ちがちです。弱い後回しを積極的にやりたいとしても、いちいち支障が生じるのです。

このような負の影響を抑えるには、スイッチングコストを減らすか、切り替えるための余裕を設けるしかありません。わかりやすく効果が高いのは後者の余裕です。仕事Aの最中に急きょ会議Bが割り込んできたとき、Bをいきなり開始するのと、15分くらい置いてから開始するのとでは後者の方が良いのです。割り込むほど重要なのだから今すぐ始めたい、としがちですが、ここで意識的にバッファを差し込めるかどうかでBの遂行、その質が大きく分かれます。木こりのジレンマという言葉があります。劣悪な斧を使っている木こりは、斧を研げばいいのに「作業で忙しくて研ぐ時間なんてない」と言って研がないのです。これと同じで、文脈を理解した方が捗るのに、暇が無いといって理解しない（スイッチングしない）のです。直感に反するかもしれませんが、よほど緊急でいちかばちかでも早くやらねばならないケースを除き、バッファは意識的に取ることをおすすめします。

もう一つ、スイッチングコストを減らす方法ですが、これはタスクA側のコンテキストを上手く保存しておくのが良いでしょう。仕事Aの最中に会議Bが割り込んできた場合、Aとして何をしていたかとかどこまで考えていたのかといった情報を外に出しておくのです。メモ書きでもいいですし、画面のスクリーンショットでもいいでしょう。とにかく何かしら思い出せるようなヒントを残しておきます。こうしておけば、Bが終わってAに戻ってきたときもすぐに思い出せて再開できます。スイッチングコストも（何も残さず自分の頭だけで頑張って思い出すのと比べると）ずいぶんと抑えられます。

Active Weakの防止

次に防止についてですが、わかりづらいのでもう一度書いておくと、防止とは割り込んでくるタスクBを確実かつ素早く対処できるようにするという話です。Bの確実かつ迅速な遂行が妨害されるのを防止する、というわけですね。

防止として行えることは大まかに二つあります。

一つ目が割り込みを発生しやすくすることです。発生しやすくと一口に言ってもやり方は様々で、すでに見たように「何かあればいいからすぐ声を掛けろ」と周知させることはもちろん、通知の仕組みがあるのなら通知の強度を上げることもできます。割り込んでくる側を信用できないなら、自分から監視しに行ったり、監視する人材や仕組みに頼るのもアリでしょう。一方で、何でもかんでも割り込まれてもすべてには対応できないため、どこかで優先順位付けが必要になります。最適なバランスは少しずつ微調整しながら探るしかありません。ここを端折ると形骸化します。わかりやすい例が階層組織における緊急時の報告で、たいてい現場の下っ端かどこかが確信犯的に報告をサボったり、報告ルートの中間層が多すぎて届くのに時間がかかったりしますよね。これはどちらも微調整をしていないからです。理想論になりますが、どんなタスクをどの

ように割り込ませるか、にはその人その組織その状況で最適なバランスがあり、この模索に終わりはありません。逆説的に、常時バランスを調整するつもりで望むのが良いのです。

もう一つは、割り込まれたときにすぐ動けるように普段から身軽にしておくことです。唐突ですが、ヤクザのトップが普段暇しているのは、いざというときにすぐ動けるようにするためだそうです。左ききのエレンというマンガでも『おっさんつーもんはさ、若者がドタバタしてる時のタメにヒマしてんのよ』との台詞が出てきます。そうでなくとも、会社員の人なら普段割と暇そうにしている上層部の人たちが、緊急時になるとすぐに真剣に動き出すとの光景を見たことがあるのではないでしょうか。身軽な人が優秀で役に立つとは限りませんが、身軽な方が動きやすいのは事実です。Active Weak が望ましい場面であるなら、できるだけ身軽になれるよう整えにいくことは重要です。これが意外と難しかったりします。私たちは退屈に耐えられるようにはなっていませんし、全うであれば罪悪感もありますし、実際色んな仕事で役に立てる力もあるでしょうし、ということで仕事を抱えがちだからです。もちろん完全に暇して遊んでばかりいても、割り込まれたところでコンテキストスイッチングできないので役に立ちません。すぐに切り替えできるよう、ある程度状況のキャッチアップは必要です。かといって、キャッチアップさせるために会議を開催させたり思いつきで呼び止めて喋らせたりするのも（やられる側からしてみると）鬱陶しいですよね――と、中々に難しいのです。

3 Passive Strong

脱線に対して強い後回しを行うのが **Passive Strong** です。タスク A の最中に誘惑 B が目に入っても「いや B には逸れないぞ」と踏ん張って B を後回しにします。B の誘惑に勝つことを意味します。

Passive Strong にも対処と防止があります。対処とは誘惑 B に負けずに、元やっていたタスク A を続けることです。防止とは、そもそも誘惑が発生しないようにすることです。Active Strong と同様、重要なのは防止の方です。対処としてできることは限られており、対処に頼るのは分が悪いからです。

対処についてですが、要は誘惑に勝つということですので、できることはあまりありません。意志を強く持ってください、くらいしか言えません。それができればそもそも苦労はしません（アスリートタイプの人なら比較的やりやすいでしょうが）し、特性次第では意志が全く役に立たないこともあります。

となると事実上 **防止をどれだけ行えるかにかかっています**。防止の戦略としては、意志を強くするか、誘惑が起きないようにするかの二つがあります。

意志を強くするとは、動機や感情といったものを日頃から高めておくことです。やる気が起きやすい仕事だけを普段から行う、自分を支えてくれる人のことを高頻度で思い浮かべたり見返したりして何度も心を奮い立たせる、またそもそもネガティブに陥らないよう体調を日頃から整える（調動脈でいう調、調子ですね）などがあります。当たり前のことを泥臭く行わないといけません。ポイントは二つあって、なんだかんだ人間であり感情や非言語情報が強いのでなるべくこれ

らが多い状況に身を置くことと、やる気が起きる仕事というよりも「起きない仕事を避ける」という発想をすることです。**プラスを得るよりもマイナスを避けます**。スタンスの章にて取り上げた Minimize Negatives です。同ページでは時間や精神をセーブするために、との書き方をしていますが、実は普段から活動的に生きる場合も重要なのです。自分にとってネガティブなものにはどうしても意志が強く働かない（※1）ので、なるべく遠ざけた方が良いのです。そういう取捨選択を繰り返していくことで判断の目も養われていき、誘惑にも勝ちやすくなります。逆を言えば、ここをサボっていると、一向に鍛えられないため、（意志に頼って抗うという前提では）誘惑に振り回される日々となります。意志なんかあてにならないという人は、たいていは単にこの営みをサボっているだけです。もちろん、それ以前に特性上意志に頼ることの分が悪い人たちもいます。アスリートに対するアプライアはすでに述べましたし、ADHD などの発達障害も知られていますし、精神疾患など病気が絡んでいることもあれば、単に慢性的に多忙で意志を発揮する余裕がないケースもあります。

次に誘惑が起きないようにするとは、環境を変えることです。自宅だと誘惑が多いのでカフェで勉強するとか、学校や会社だと場の雰囲気や周囲の目があるから打ち込みやすいなど、言葉にするまでもなく知られていることですが、これを意識的に取り入れます。この点はリマインダーの章でも少し取り上げました（雰囲気リマインド）。ポイントは誘惑に手を伸ばすまでの物理的コストと精神的コストの両方を減らすことです。たとえばお酒が大好きで、思考力が低下するし規則的にダメだとわかっていても日中リモートワーク中に飲酒の誘惑に勝てない不真面目な社員がいるとして、おそらく出社してしまえば完璧に防げると思います。オフィスなら物理的にお酒を入手する手間も大きいですし、入手したところで同僚の目があるから飲めるはずもないという精神的な障壁もあります。一方、自由度が非常に高いベンチャーや自営業の飲食店などでは、あまり効果がないどころか、むしろ仲間や常連と一緒に盛り上がってしまって逆効果かもしれません。さて、「誘惑が起きづらい環境」ですが、人それぞれですので、自分で探さないといけません。最初のうちは、いきなり狙ってありつくのが難しいので、テキトーに色んな環境に飛び込んでみて感触を掴むのが良いでしょう。そのためには色んな環境に飛び込むバイタリティが必要で、日頃から健康的な心身をつくる努力をしておいた方が良い――と、いつもの泥臭い結論がなんだかんだ王道だったりします。ちなみに、同じ人であっても、加齢やライフステージの変化によりここは変わります。環境を見つけて一生涯キープするというよりは、必要に応じて探せるようなフットワークを身に着けておく方が融通が利きます。

環境の変え方はまだあります。物理的な場所を変えるだけではなく、普段使っている環境を工夫することもできます。たとえば普段使っているスマホやアプリの通知を切る――これも環境を変えていると言えます。通知を切れば、通知という誘惑はなくなりますよね。とてもかんたんなことですが、実際難しいのは **依存しているから** です。現代人の情報への依存は改めて言うまでもなく当たり前の事象ですが、だからこそ向き合う他はありません。依存を断ち切るために使えるのが、環境を丸ごと変えてしまうことです。デジタルデトックスなる営みがまさにそれで、数日間物理的にスマホに触れないような生活をしたりします。そこまでやると、スマホに依存していた脳が少し緩和されて「比較的抗いやすい状態」になります（なることがあります）。この状態で現実に戻れば、通知を切る行為もあっさり行えるかもしれません。依存している限りは切るという行動が取れないので、取れるようになるまで荒療治が必要というわけですね。そのために、依存先に頼らない環境に身を置くのです。デジタルデトックスは極端な例ですし、もっとひどいと依存

症となり専門家による治療が必要ですが、そこまでひどくないなら、軽い工夫も含めてやり用は色々あります。動線や習慣の考え方は役に立つでしょうし、役に立つと言えばタスク管理自体がそうですが、要は **色んなやり方や考え方があることを知り、自分の特性も知った上で、上手く自分の行動を誘導できるような仕掛けをつくる** のです。個人タスク管理とは、タスクを上手くこなすよう自分なりに仕掛けをつくる営みだとも言えます。当然ながら環境を変えるための仕掛けもつくれます。そのヒントは本書の随所にあると思いますので、自分なりに模索していただければと思います。

- ※
    - 1 ネガティブなものには意志が働かないと書きましたが、ここでいうネガティブとは起伏に乏しい、勢いが小さい、反応しないといったニュアンスです。殺意や憎悪など、いわゆる「負の強い感情」は、もちろん意志を支えてくれます。筆者としては、これらもプラスだと思っています。ただ方向性に違いがあり、一部の方向性によっては倫理的に推奨されない（あと消耗や疲弊が大きい）というだけの話です。

4 Passive Weak

脱線に対して弱い後回しを行うのが **Passive Weak** です。タスク A の最中に誘惑 B が目に入ったら、A は置いといて B にホイホイついていきます。自ら誘惑にかかりに行くわけですね。

Passive Weak における対処とは、誘惑 B にすぐに移ることです。防止とは誘惑がなるべく発生するようにすることです――というとわかりづらいですが、「これだ！」という視点が得られるまで粘り続けるアイデア出しや暇つぶしの文脈が該当します。

まず対処ですが、対処として行えることはほとんどありません。というのも、単に何もしなければ誘惑には負けるからです。そういうわけで、Passive Weak についても防止の方が重要になります。

防止の戦略はおおまかに二つあります。誘惑に勝ててしまう状況を薄めることと、誘惑が生じる機会を増やすことの二つです。

誘惑に勝てる状況とは、タスク A に集中しすぎているとか、タスク A の重要性が高すぎるせいで A 以外には見向きもしなくなるといった状態を指します。こういう状態ですと、多少誘惑が来ても見向きもしないので Passive Weak はできません。集中しすぎないようにしましょうとか、重要すぎるタスクは抱えないようにしましょう、と助言することになります。前者、集中しすぎについては意志でコントロールできるものではないので、集中を妨害する仕組みをつくるなり、集中しづらい場所に足を運んだりします。単純なやり方ですと、機械的に1時間ごとにリマインダーを仕掛けるだけでも違いますし、あえて PC を使わずスマホで仕事をするとかデスクや椅子や自室を捨てるといった縛りを加えるなど不便を増やすことでも対応できます。自分に負荷をかけるイメージですね。続いて後者、重要すぎるタスクを抱えないようにする方ですが、これは人生哲学レベルの話になり、難しいです。たとえばペットや子供といった養育に手を出さなければ、生物の養育という「重要になりがちなタスク」も起きなくなりますが、だからといって養育しない生き方を

選ぶのは難しいでしょう。仕事も同様で、期待や責任の重たい仕事を抱えないようにするために、たとえば無能や変人を器用に演じられるかというと、やはり難しいのではないでしょうか。

二つ目、誘惑が生じる機会を増やすという点については、受動と能動があります。単に誘惑が多い場所に行ったり、ネットサーフィンやゲームがまさにそうですが誘惑――というより刺激ですね、刺激の多い手段に溺れるのが受動です。手間無く暇を潰すために使えますが、ご存知のとおり行き過ぎると依存度が高まって中毒になります。手軽ですがハマりやすいわけですね。能動については、探索と言い換えても良いもので、「これだ！」と思える何かと出会うために自分から情報や刺激を取りに行くことを指します。芸術的な文脈だと体験に身を投じてインスピレーションを待ったり、創造的な文脈だとリラックスしてのんびりすることでひらめきに賭けたりしますが、通常は大量のインプットに身を委ねるか、文章や発言など思考をアウトプットしながらその続きや連想を手繰り寄せていく必要があります。受動よりも高度な営みであり、仕事にせよ趣味にせよ、明確な目的や強い動機が無いと始められませんし、続きません。それなりの集中力とスキルも必要としますし、何より正解が無い中で粘り続ける胆力（場合によっては自分で正解を決める意思決定も）も要求されますし、当然ながら疲れます。このとおりハードルもクセも強いですが、使いこなせると強いです。

SWAP マトリックスのまとめ

ワープをどう処理するかは SWAP マトリックスで分類できるのでした。

そもそも処理として対処と防止がありました。対処とは、起きたときに対応することです。防止とは、そもそも起きないように（あるいは恒常的に起きるように）備えることです。

SWAP マトリックスの 4 パターンは以下のとおりです。

- 1: Active Strong、割り込みに対して強い後回しを行う
- 2: Active Weak、割り込みに対して弱い後回しを行う
- 3: Passive Strong、脱線に対して強い後回しを行う
- 4: Passive Weak、脱線に対して弱い後回しを行う

ここまでを踏まえると、4 パターンそれぞれに対して対処と防止があるとわかります。内容も含めて、改めてまとめると以下のようになります。

- 1: Active Strong、割り込みに対して強い後回しを行う
  - 対処: 強引に断るか、相手が納得するように器用に断る。タスク管理というより対人コミュニケーションの話になる
  - 防止: 相手に働きかけるアタック、相手から距離を置くエスケープ、相手と対応する役を置くガード
- 2: Active Weak、割り込みに対して弱い後回しを行う
  - 対処: 集中を解除させる行動をしてもらう（してもらうには防止策が必要）、コンテキストスイッチングコストを最小化する

    - 防止: 割り込んでもらいやすくするための仕組みや仕込みをする、割り込まれたときにすぐ動けるよう身軽になっておく
- 3: Passive Strong、脱線に対して強い後回しを行う
    - 対処: できることはない。強いて言えば「意思を強く持って誘惑に勝て」くらいしか言えない
    - 防止: 意志を強くするか、誘惑が起きにくい環境に行く or 変えていくか
- 4: Passive Weak、脱線に対して弱い後回しを行う
    - 対処: 何もしなくていい。何もしなければ自然と誘惑に負ける
    - 防止: 誘惑に勝ててしまう状況を薄める（自分に妨害の負荷を課すかのと重要なタスクを抱えない生き方をするのと）か、誘惑が生じる機会を増やす（受動と能動がある）

各パターンを上手く使い分けてください。割り込まれたくないなら Active Strong を使います。逆に割り込んで欲しいなら Active Weak を使います。割り込みはないが脱線がある場合で、脱線せず今のタスクに集中したいなら Passive Strong を使います。色んな刺激やヒントが欲しいなど反応的・探索的なシチュエーションであるなら Passive Weak を使います。

Passive Strong の対処

SWAP マトリックスの 3 つ目は Passive Strong――誘惑があっても負けない、つまり脱線しないためには、について書きましたが、対処法はありませんでした。強いて言えば「意志を強く持て」ですが、意志でどうにかなるなら苦労はしません。一応他にも方法はあります。ただ実践が難しく、人を選びますので、高度なトピックスとして本項で解説します。

方法としては以下の 2 つです。

- Explicit Attention（注意の明示）
- 生活リズムの固定

Explicit Attention

Explicit Attention（注意の明示） とは、書きながら行動する過ごし方を指します。何らかの行動を行う際に、常にタスクや思考を書いて、書いたそれを見ながら進めていきます。また進めた内容は書き込みに反映していきます。

書くことで注意の対象を明示的につくる

通常、私たちは頭の中で考えながら行動したり、慣れているなら考えすらしないで反射的に動けたりします。しかし Explicit Attention では少し手間を追加して、「今現在の文脈を書きながら」「それを見ながら」「書いてるそれらも更新しながら」という制約を加えます。わかりやすいの

は買い物リストです。頭の中で覚えたり、その場で思い出しながら買うのではなく、手元のメモを見ながら買っていくわけです。また単に見るだけではなく、カゴに入れた項目は消します（更新）――と、このような営みを脱線対処の観点で一般化したのが Explicit Attention です。

その名のとおり、**書くことで注意の対象を明示的につくる** のが肝です。まず、書くことで言語化や取捨選択が行われるので、そのことに集中しやすくなります。かつ、書いたそれを読むことで、脱線しそうになっても思い出して踏ん張ることができます。買い物リストもそうですよね。リストをちらちら見たり、買ったものを一つずつ潰したりすることで「次はこれを買おう」とか「余計なものは買わなくていい」とか「考え事浮かんだけど今はこれらの買い物を済ませよう」とかいった軌道修正――脱線から逸れないようにキープすることができるわけです。

何を当たり前のことを、と思われるかもしれませんが、Explicit Attention の力はとてつもないものです。買い物リストの例だけを見ても、何にも頼らないよりは確実かつ安心して忘迷怠無くこなせることがわかるかと思います。もちろん 100% ではなありませんが、たとえば絶対失敗が許されない場面ですと誰でも頼ろうとするのではないでしょうか。Explicit Attention は、この力を汎用的に使えるようにするものです。こんな小難しい名前をつけるまでもありません。特に買い物リストなどチェックリストを持って一つずつチェックしていく、くらいであればやったことがある人も多いでしょう。原理的にはそれだけです。常にチェック項目をつくって、それを見ながらチェックつけていって、項目増えたらまた書き足して、などとチェックリストを相棒に行動するだけです。相棒が注意を引き付けてくれるから脱線しづらくなるという、ただそれだけのことです。

軌道修正

Explicit Attention は脱線に対抗できる手段です。書くことで注意の対象を可視化し、それを読むことで注意がそれから逸れないようにします。あるいは **逸れかけていても** 軌道修正できます。逸れた後はダメです、逸れた後は自分が疲弊するか外部から刺激が来ない限りは戻れません。あくまでも逸れる前までが勝負です。

別の言い方をすると、逸れない程度には高頻度に書いたり読んだりする必要があります。たとえば買い物リストは、1分以内にチラチラ見てないと逸れてしまうかもしれません。人によっては数分以上見なくても逸れずに済むかもしれませんが、おそらく通常は数分も空けてしまうともう逸れてしまうと思います――リストに無かった品物を買うかどうか悩んだり、リストがあるのにそれ以上読むのを放棄して勘で買い物を続けたり、常連や店員との会話に夢中になってしまってリストの存在を失念しまったりするのです。もしあなたが「1分以上目を離すと脱線しやすい」のであれば、1分未満の間隔で買い物リストを確認、メンテナンスしないといけません。このあたりの話は前章で言及したネグレクテッドとロストと似ていますが、Explicit Attention では、より細かい頻度が要求されます。

必要なのはスキル

さて、どういう行動のときに、どんなことを、どこに書くか――についてですが、人次第、状況次第としか言えません。デスクなど落ち着いて読み書きができる状況が好ましいことを除けば、各自

で探るしかありません。

少なくともスキルは必要です。言語化する分、言語化したそれを実際に書く分、書いたそれをすぐに読み返して理解する分、終わったものや古いものを消したり退避したりといった更新の分、など必要な操作はいくつかあり、これらをすべて息するように行えなくてはなりません。このあたりのスキルが鈍いと、読み書きの手間が大きすぎて形骸化します。

一つの目安は **正解の無い検討を一時間以上、脱線なく継続できるかどうか** でしょう。引っ越すとしたらどこがいいかとか、一億円もらったとしたら何をするかとか、ちょうど 30 日後に心臓麻痺で死ぬとしたら今からどうするかなど、検討内容は何でもいいですが、こういったことを一時間の間、脱線を起こさず続けられるでしょうか。さすがに 1 秒も脱線しないのは無理ですが、「一時間極力使ってちゃんと検討しました」と胸張って誓って言えるくらいです。なお体調、疲労、忙しさは問題無いと仮定してください。

通常は（他者や環境による強い干渉がなければ）まず不可能でしょう。難なくできるならそれは才能と呼べるものだと思います。これを可能にするために、Explicit Attention を使うのです。手段は何でも構いません。スマホでも、PCでも、ノートでも手帳でもいいです。頭の中にノートをつくれるならそれでも良いですが、脳内は脱線しやすいのでおそらく難しいと思います。だからこそ Explicit Attention では「書くこと」と書いています。前述したとおり、落ち着いて読み書きできる環境がベストですが、慣れれば電車内や散歩中などでもできます。

この目安を達成するために、何を使ってどんなことを読み書きしながら進めていけばいいか――この答えは人それぞれなので、あなた自身が探るしかありません。それが Explicit Attention です。特性もありますし、人によっては一生できないかもしれません。しかしながら、Explicit Attention が上手くいかない主因はスキルです。単に素早く読み書きできる要領がないだけです。

本当は Explicit Attention の具体的なやり方も体系的にまとめて説明できたら良かったのですが、本書ではそこまでは踏み込みません。一応、第 5 部では「書く営み」を重視したタスク管理を提案しており、参考にできるかもしれません。

自分を動かしてくれるバランスが難しい

実は一番むずかしいのは **自分を軌道修正してくれるような言い回しや雰囲気** だったりします。

たとえ読んだとしても、脱線しかけていた意識を戻せなければ意味がありません。書き込みが複雑すぎると読んでも頭に入らずスルーしますし、逆にかんたんすぎるとすぐ理解した（した気になった）はいいものの腑には落ちなくて……、とどちらに転んでも形骸化します。このバランスを自分で探り当てるのが難しいのです。

たとえば買い物リストで言うと、単に買うものを並べているだけで問題なく軌道修正できる人もいれば、それだけだとスルーしちゃうので自分を叱る言葉を別途書かないといけない人もいます

し、箇条書きだけだと味気なさすぎてスルーしちゃうけど可愛いメモ帳のデザインに書いたメモならちゃんと読めるなんて人もいます。特に最後の例が顕著ですが、書き込んでいる内容のみならず書き込み先のデザインや雰囲気が絡むこともよくあるのです。

人は思っている以上に言語情報の扱いが下手で、かつルーズです。通常は言語以外の要素を総動員して、Explicit Attention として使っている道具そのものから離れてしまわないような工夫に苦労することになります。雰囲気の他には、「みんな使っているから」とか「あの人から勧められたから」といった対人要素もよく使われます。使うからみっともない、なんてことはなくて、むしろ使えるものなら何でも使いたいです。買い物リストを使い続けられるかどうか、また使っていて実際に脱線を防げるかどうかがすべてですので、手段は選ばないくらいのつもりで追い求めましょう。直感もあてにしていいですし、機能性よりもおしゃれだから選んだ等もアリです。

もう一つ、もし可能であれば、言語情報自体をすらすら扱えるように底を上げておきたいところではあります。語彙力と言えばいいのか、国語力と言えばいいのか、筆者もまだ答えを出せていませんが、**自分が書いた** 言語情報と触れ合う基礎体力なるものがあって、ここをいかに高めておくかが存外重要と感じます。空いている階段と混雑しているエスカレーターがあって後者を選ぶ人は多いですが、これは身体能力というよりも、自分のためだけに体を動かす基礎体力なるものが少ないからです。なので身体能力が高いスポーツ部の学生でも基礎体力がないと階段を登ろうとしません。逆に基礎体力がある場合は、階段で疲れて息が切れるとわかっていても難なく階段を登ろうとします。この「基礎体力なるもの」が、読み書きについてもあると筆者は感じています。実際、文章が上手い人やタイピングが早い人が Explicit Attention が上手いとは限りません。一つ問題を挙げるとすれば、この基礎体力の鍛錬も精神的に難しいことでしょうか。他人に向けた文章でも作品でもない、単に自分が自分に宛てた醜い文章と何度も向き合うことになるわけですから。文章の拙さはもちろん、これだけしか言語化できないのかといった現状も残酷なまでに可視化されます。脱線を防ぐということは、逃げずに向き合い続けることでもあるのです。

生活リズムの固定

脱線に対処するもう一つの方法は、**生活リズムをできるだけ固定してしまう** ことです。

リズムリマインド

これは単純な話で、染み付いた生活リズムがあると、いつ何をするかが大体固定されるので、脱線しそうになったときに気持ち悪さや居心地の悪さを感じて抵抗しやすくなります。「16 時からご飯を作り始めなきゃいけないから」と脱線先のネットサーフィンをやめたり、就寝前の自由時間でつい漫画に熱中してしまい 22 時の就寝をオーバーしそうなときでも「そろそろ寝る時間か」と気づけたりします。身体がリズムを覚えていて、思い出させてくれるイメージです。

これを私は **リズムリマインド** と呼んでいます。音楽やゲームのリズムというより、生活リズムや体内時計といった生理的なリズムです。そんなバカな、と思われるかもしれませんが、実際に人に

はサーカディアンリズムやウルトラディアンリズムといったものが備わっていることが知られています。特にスポーツや養育などで厳しい生活をおくった経験のある人は、体がリズムをおぼえている、との感覚を知っているのではないでしょうか。

シークレット・シチュエーション下で誘惑に勝つ

だからといって、実際にリズムリマインドがすぐ手に入るかというとそうでもないですし、リマインドというからには確実性が高いかというとこれもそうでもなくて、要は個人差によりけりなのですが、それでも無いよりははるかにマシです。

筆者の肌感覚として、リズムリマインドの強度は「外部からの圧力」未満、「自分の意志」以上、といったところでしょうか。単に意志に頼るだけと比べるとかなり頼もしいですが、それでも子供やペットやチームやその他環境といった外部からの働きかけほどは強くはありません。

それでも、多少苦労してでも身につける価値があると筆者は思います。なぜならば **誰にも見せず知らせずに働きかけてもらえる** 数少ない手段だからです。ひとりで集中したいときや、誰にも見せたくないことをしている場合など、誰にもどこにも見せたくない場面（シークレット・シチュエーション）はあると思いますが、それはすなわち外部の干渉に頼れないことを意味します。脱線に勝てるかどうかも意志次第となってしまいます。が、意志などここまで見てきたように現実的ではありません。だからこそ通常は対処ではなく防止を重視します。Passive Strong の項でも防止にかかっていると書きました。具体的には Minimize Negatives――マイナスを避けたり、あるいは純粋に環境を変えたりするのだと書きました。そしてこれらは、わざわざ小難しく解説するまでもなく、ストレスフルな人からは距離を置く、カフェや図書室で作業する、など誰もが体感的に知っていることです。しかし、それではシークレット・シチュエーションを担保できません。このジレンマを打ち破れるのがリズムリマインドなのです。

リズムリマインドを育てる

ではリズムリマインドはどのように獲得、強化していけばいいのでしょうか。

まずリズムリマインドは、体内からツッコミが来るイメージです。これぱっかりは経験しないとさっぱりわからないと思いますが、リズムが染み付いてると「次の行動が控えてるけどいいの？」とのツッコミが体内から来る――そんなイメージです。そこでちゃんと向き合って、踏みとどまれば脱線はしません。逆に「知るかよ」と一蹴してしまうと、そのうち麻痺してしまって全く効果がなくなります。ツッコミをどう扱うか次第で、効果もそちら側（扱っている側）に寄っていきます。

つまり、リズムリマインドの精度を上げたければ、ツッコミと受けて実際に脱線に抗う（リズムを死守する）、との実績を積めば良いと言えます。

これは結局生活リズムを守るモチベーションをどれだけ高められるかにかかっています。たとえば朝6時に起きて夜22時に寝る、というリズムを毎日欠かさず繰り返すとして、ここにどれだけモチベーションを絡められるでしょうか。もちろん 6 と 22 が正しいとも限らず、人によっては別の数字を模索せねばなりません。模索のためのモチベーションが湧くでしょうか。これぱっかりは人次第

なので何とも言えませんし、もちろん日中リズムリマインドに頼りたいなら日中の範囲でリズム化が必要です。起床と就寝を固定する程度で根を上げているようでは正直話になりません。

仮にアドバイスをするとしたら、すでに自分に刻まれたリズムを尊重して改良していく（ハビットエンジニアリングと同様、微調整していく営みになる）くらいでしょうか。刻まれたリズムがないなら、何とかして刻むところから始めます――といっても、何らかの環境に飛び込むなりして生活を丸ごと変えるくらいしかありません。少なくとも刻まれたリズムがない状態から育てていくことは難しいと思います。

整理します。

- 1: まずは「リズム」を刻むところから始めます
  - ここでリズムとは「内からツッコミが来る（リズムリマインドが起こる）」程度のものです
  - 来たら刻まれていると言えますし、来ないならまだ刻まれていません
  - 問題は、このリズムの刻み方がほぼ無いということです
    - 何らかの環境に飛び込んで浸かって過ごしてたらそのうち刻まれた、くらいしかありません
  - 今現在リズムが刻まれているか、あるいは最低でも **過去に刻まれたリズムがある** との実績は必要です
- 2: リズムがある or 実績があるなら、それをベースに微修正していきます
  - ハビットエンジニアリングと同様、日々少しずつ加えたり削ったり変えたりなど微調整していくことで自分に最適なリズムにしていきます
  - たとえば筆者の場合、今は右記のようなリズムです――朝5時に起きて、6時に仕事を開始して、9:30時に昼食、12時までに2杯目のカフェイン、13時におやつ、16時に夕食、18時までにすべての仕事と作業を済ませて、20時に歯磨きや食器洗いなど寝る準備、21時に就寝
    - 細かい数字は必要に応じて変えていくことができます

副次効果としてノイズも減る

リズムリマインドが起きるほど生活リズムが染みつくと、実は生活上のノイズも減ります。要は余計なことを考えたり、必要以上に情報を貪ったりしなくなります。その分、誘惑の原因も少なくなりますから誘惑される機会も減ります――つまり脱線しづらくなります。

書籍『道は開ける』では悩むのは暇だからだとでも言うべき金言が強調されています。これは忙しくしていれば悩む暇なんてない、だから悩みたくなければ忙しくなれというものです。忙しさというと通常は仕事か養育を浮かべると思いますが、筆者はもう一つあると考えます。それが生活リズムです。生活リズムを守ることに忙しくなることでも暇はなくせます。暇がなくなれば悩まなくなります。あれこれ考えるのはもちろん、余計な情報を読み漁ったりもしなく（しづらく）なります。

生活リズムについて、よく「そこまで厳しくする意味なんてあるの？」との疑問をいただくのですが、「自分を忙しくできる手段の一つ」という意味では意味があるのです。

忙しくできるなら別に生活リズムじゃなくてもいい

となると結局、忙しくできれば誘惑にも勝てる（というより負ける暇がない）わけでして、だったら別に生活リズムを刻む必然性はないわけです。仕事か養育でカバーできるならそれでも構いません。

ただしすでに述べたとおり、生活リズムは唯一他者からプライバシーを守れる手段であり、シークレット・シチュエーション下で誘惑に勝つのに重宝します。

別に誘惑に勝てる手段を一つだけ使う道理もないのですから、使えるだけ使えた方がお得です。仕事や養育と同様、かんたんに手に入るものではないですし、日々のメンテナンスや微修正も必要ですが、誘惑に悩まれてる方はぜひ挑戦してみると良いでしょう。

まとめ

- ワープとは
  - 割り込みと脱線がある
  - 弊害は「パフォーマンスが阻害されるから」と「コンテキストスイッチングが発生するから」の二点
  - 構図は「今行っているタスクAに別のタスクBがやってくる（そしてBを始めてしまう）」と捉えることができる
- ワープと向き合う
  - 後回しという概念がある
    - Aを捨てるか（弱い後回し）、Bを捨てるか（強い後回し）、どちらかを必ず行うという前提を取る
  - 割り込みと脱線/後回しの強弱、の二軸でマトリックスにできる
  - これを SWAP マトリックスといい、ワープとどう向き合うかを 4 パターンで捉える
- Passive Strong（脱線に対して強い後回しを行う＝誘惑に負けない）における対処は難しい
  - 1: Explicit Attention（注意の明示）。書き続けることで注意を引きつけ続ける
  - 2: 生活リズムの固定（リズムリマインド）。染み付いた生活リズムからの逸脱に敏感になる≒誘惑にも踏みとどまりやすい

=== Chapter-20 振り返り ===

タスク管理というとタスクをいかに上手く（≒効率良く）消化するかを考えがちですが、一方で消化だけで乗り切れるほど単純な時代でもなくなってきています。現状を分析する、今後について考えるなど、単なる消化を越えた思考や検討を行う機会も増えてきていると思います。そう

でなくとも、漠然とした不安を抱えてあれこれ考えてしまう経験は誰しも一度はあるのではないでしょうか。

このような活動や不安に役立つのが「振り返り」です。タスク管理と対応付けると、振り返りにて戦略を考え、タスク管理にて戦略をなぞると言えます。どのようになぞるかという戦略を考えるのに、振り返りが役立つのです。もちろん、人や環境から言われたとおりに従うだけでも良いですが、そうではなく、自ら主体的に考えたい方にとって振り返りは役立ちます。

本章では振り返りについて解説します。なお、プロジェクトタスク管理に代表されるようなプロセスは扱わず、個人で行う分をコンパクトに解説します。

振り返り

振り返りという言葉は多義語です。文脈によって、また人や組織によって意味が変わりますが、それだと扱いづらいので、本章では定義から定めます。もちろんこの定義が絶対的に正しいというものではなく、このように捉えると扱いやすいですよ、との筆者の解釈と考えてください。

振り返りの定義

振り返りとは **過去の情報をインプットにして考察を行い、アウトプットを出すこと** です。

ポイントが3点あります。

- 1: インプット。特にインプットとして過去の情報を使うこと
- 2: 考察。
- 3: アウトプット。後述しますがアウトプットとはタスクとモットーとコンテキストのこと

インプットして、何か処理をして、アウトプットが出てくる――のような世界観は数多くありますが、振り返りもその一つです。

インプットについて

インプットとは過去の情報を使うことを指します。

過去とあるように、あくまで過ぎ去ったものを扱います。未来について考えるとか想像するとかいったことは（インプットの時点では）しません。

情報については何でも構いません。正式な仕事やプロジェクトといった「活動」もあれば、ただの行動や言動や行為といった「動き」もあるでしょう。もちろん「過去の天気や湿度や気圧」といった、まさに情報と呼べるものも含まれます。

考察について

インプットした情報を考察するとのことですが、考察とは何でしょうか。

細かい定義はさておいて、強調したいのは **自分の頭で考えて決める** ことと、**必要に応じて主観以外の情報にも頼る** ことの二点です。よく挙がるのが考察と洞察の違いですが、洞察の方が直感やひらめきに頼るウェイトが大きいニュアンスがあります。しかし、それだけではなく、後者のとおり主観以外の情報にも頼りたいので、考察という言葉を選びました。

また、分析や解析、審査や検査といった重たい言葉も使っていません。個人で行う振り返りですから、気軽にできてもいいのです。そのニュアンスを出すために重たい言葉は使わず、考察を選びました。

アウトプットとはタスクとモットーを出すこと

最後にアウトプットですが、これは **タスクとモットー（あと可能ならコンテキスト）を出すこと** と定義しています。

そもそも定義にアウトプットを含めているのは、振り返りをしたことで次に繋げたいからです。振り返りというと、よく「学び」「気づき」「よかったこと」「悪かったこと」といったものが挙がりがちですが、これらを挙げただけでは次につながりません。これらを挙げただけではまだ「考察」の段階にすぎません。次につなげるためには具体的な何かを出す必要があります。

何かとは何か――タスク管理の文脈で言えば、まずはタスクとモットーです。何らかの行動を示すタスクを出すか、あるいはタスクではないが方針や方向性や抱負などを示すモットーのいずれかです。これらを出すことができれば、じゃあ次はタスクをこなそうとか、モットーを心がけるようにしよう、といった形で次に繋げられるのです。

もう一つ、コンテキストという言葉がありますが、これは努力目標です。できれば出しておきたいものです。ここでコンテキストとは「その振り返りにおいてどんな判断や意思決定をしたか」「なぜそうしたか」といった文脈のことです。後述しますが、これらアウトプットは未来でまた使うことがあります。そのときにコンテキストがわかりづらいと使いづらいので、このタイミングでコンテキストも出しておくのです。必須ではありませんし、**コンテキストを出すのが辛くて振り返りに支障が出るくらいなら出さない方がマシです**。余裕がある場合に出しておく、くらいの認識で構いません。

振り返りに正解はない

ここまでインプット、考察、アウトプットの三要素を述べました。これらをどう行うか次第で、振り返りには無限のバリエーションがあります。つまり **振り返りに正解はありません**。

とはいえ「あとは各自頑張ってください」だと放棄ですから、以降ではこうすれば比較的上手くい

くというヒントを出していきます。

振り返りのPROC

振り返りの定義はインプット、考察、アウトプットとしましたが、ここに登場していない重要事項があります。それが頻度と目的です。振り返りをいつ行うのか、また、なぜ行うのかも考えた方がスムーズです。

このあたりの検討要素を示したのが PROC です。

英語	意味	解説
Purpose	目的	なぜ振り返るか
Remembering	想起	インプットをどうやって集めるか（思い出すか）
Oppotunity	機会	いつ振り返るのか
Continuation	継続	一度振り返ったらおしまいなのか、それとも何度も続けるのか

Purpose/目的については、漠然としているとモチベーションが続かないため、はっきりとした目的があった方が良いという話です。この後詳しく述べますが、振り返りは地道な営みです。モチベーションが無いと十中八九挫折するか、あるいは意味もなく惰性でだらだら続けて形骸化します。

Remembering/想起は、インプットをどうやって集めるか・思い出すかというものです。おおまかには右記の三点があります――何らかの記録を取ってそれを見る、記録なしで記憶を思い出す、思い出しやすいヒントを眺めながら思い出す。

Oppotunity/機会は、振り返りをいつ行うかを指します。定期的に行うやり方と、思いついたときや空いたときに突発的に行うやり方があります。

Continuation/継続は、わかりにくい概念ですが、たとえば「今年2023年の転職活動について振り返ろう！」なる振り返りを開催するとして、年末のある日、3時間かけて行ったとします。ここでこの振り返りを終わらせた場合、つまり2023年の転職活動の振り返りはもうしない場合、継続はナシと言えます。逆に「いや、この後も週一くらいで引き続き続けたい」といった場合は継続はアリと言えます。アリの場合、2024年年始が過ぎた後も毎週末1時間くらい費やして2月末までかかった、といったことがありえます。

振り返りというと、おそらく「一度振り返りイベントを開催して、その中で終わらせる」とのイメージが強いですが、そうではないということです。継続アリの場合、一度の開催では終わりません。もっと言えば、**一度の開催で無理に終わらせる必要はない** ということです。極論言えばだらだらと続けてもいいのです。だらだらしすぎもよくないですが、じっくり時間をかけることで見えてくることもありますし、直後に振り返るよりも一年後に振り返った方が（自分が成長しているので）

良いアウトプットが出ることもあるのです。ただ、一年前のことなんて覚えているはずがありませんから、思い出せる程度の工夫（Rememberingの部分ですね）は必要になります。

定期的に振り返るのが鉄板

振り返りは定期的に行うのが鉄板です。

仕事の振り返りは毎週金曜日の午後にしようとか、趣味のFPSゲームの振り返りは毎週日曜日の午前にしようとか、プロジェクトAでは毎日朝会があって昨日を振り返っているとか、毎月月末は家計簿をチェックして支出を振り返ろうなど色々な振り返りが混ざりますが、いずれも定期的に行っています。この例では毎週、毎週、毎日、毎月という頻度になっています。

DWMY

定期性にはDay（日）、Week（週）、Month（月）、Year（年）の単位があり、これを **DWMY** と呼びます。

DWMY のメリット

振り返りの頻度を設計する際は、この DWMY を意識すると決めやすいです。たとえば Day に注目すると、以下が考えられます。

- 毎日（1日1回）
- 2日に1回
- 3日に1回
- 毎日朝と夕方（1日2回）

DWMY ベースで決めることができたら、カレンダーなどツールを使って仕込むことができます。また頻度が多すぎるとか少なすぎるといった場合も、2日に1日から毎日(1日1回)に増やすかとか、逆に3日に1日に減らすかと言った形で定量的にコントロールできるようになります。

DWMY ヒエラルキー

振り返りはそうかんたんにはいかないことがあります。

特に振り返りの対象が時期的に広い場合、インプットの収集が困難になりがちです。たとえば今月について振り返りたい場合、今月一ヶ月分のインプットを頑張って集めたり思い出したりする必要があります。少なくとも思い出すのは難しいでしょうし、単に集めるだけでも相当な手間

暇がかかりそうです。

かといって範囲を狭くして、Day や Week の頻度のみで振り返りをすると、やりやすいのはいいですが、Month や Year といった長い期間で俯瞰することができなくなります。場当たりになりがちですし、俯瞰するからこそ気付ける何かを見逃す機会損失もありえます。

このジレンマを解消するのが **DWMY ヒエラルキー** です。これは **一つ前の段階で出したアウトプットをインプットにする** というものです。つまり、Day 単位の振り返りを行ってアウトプットをつくり（Day Output）、Week 単位の振り返りではこの Day Output をインプットにして振り返りを行い、次の Month 単位では Week Output を使って振り返りを行い――と段階的に活用していくのです。

DWMYヒエラルキーを使うと、インプットの数が絞られてずいぶんと楽になります。

- Day の振り返り時: -
- Week の振り返り時: 7個
- Month の振り返り時: 4〜5個
- Year の振り返り時: 12個

たとえば毎日振り返りをしているならアウトプットは7個あるはずで、Week の振り返りではこれをインプットにすれば 7 個で済みます。1個1個にはさらに複数のアウトプットがあると思いますが、それでも7日分、7個という単位で捉えられるので現実的です。少なくとも一週間分のインプットを一から頑張って集める手間はなくなります。そのかわり **毎日律儀にレビューする手間は生じます**。

そういう意味ではトレードオフです。毎日行う手間を負うことで後々のインプットを楽するのか、それとも毎日行う手間をなくすかわりに後々のインプットで苦労するのか。どちらが良い、というものではなく、自分に合った方を使えば良いのです。筆者は自分の記憶をあてにしておらず、記録に頼りながら生きるタイプなので、前者の毎日行う手間を選んでいます。まさに GTD がいうように日次、週次、月次レビューは全部行っていますが、これを行えるのは DWMY ヒエラルキーにより下位のアウトプットを使っているからです。

DWMY ヒエラルキーの流派として以下があると思います。

- DWMY、全部行う。律儀な人向け
- DM、毎日行いつつ俯瞰として毎月も行う。俯瞰はしたいが Week は面倒というずぼらな人向け
- DW、毎日と毎週だけ行う。俯瞰は要らないが週単位ではやっておきたいという人向け
- WM、毎週と毎月だけ行う。日単位は細かすぎて要らないが、ある程度の俯瞰はしておきたい人向け
- MY、毎月と毎年だけ行う。俯瞰を重視する、かついきなり Month から始めてもインプットを集められる自信がある人向け

日単位で細かくやりたいのか、月や年など俯瞰したいのか。また手間とインプットのトレードオフ

とはどう向き合うのか――最適解や合う合わないは人それぞれですから、自分に合ったものを選びましょう。最適との出会いには試行錯誤を要しますが、直感はあてにしていいです。仮に「たぶん私は週に一度やるくらいがちょうどいい」「日に一度はだるくて続かない気がする」と感じたなら、おそらくそれで合っています、W のみでいいでしょう。あるいは上述したように DW から始めてみて感触を見ましょう。

振り返りは予定として扱う

振り返りという行動自体はタスクというよりも予定として扱った方が上手くいきます。

予定とは約束であり、開始時間と終了時間が存在しますし事前の準備も要します。一方、タスクとは事前準備なしにその場で場当たり的にこなしておしまい、のニュアンスが強いと思います。振り返りは前者、予定のようなものとして扱う方が上手くいきます。なぜなら、振り返りはそれなりに集中してじっくりと取り組むものだからです。考察を行ってタスクやモットーといったものを導くところで集中を要するのです。

振り返りは（本章では）個人的な営みであり、個人的だからこそ何をどう考察して何をアウトプットするかも全部自分が決めねばなりません。内省という言葉がありますが、まさにそうで、自分と深く向き合うことも少なくありません。そもそも正解もありませんから、こうすればよい、がすぐわかるとも限りませんし、わからないことの方が多いです。仮説検証とか暫定といった言葉の方がまだ似合います。そういう世界なのです。

人と会うときはちゃんと予定を入れて、その時間中はその人のためだけに使うと思います。振り返りもこれと同じ意識で望んでください。そこまでしないと振り返りはまともに機能しません。これは振り返りを行えるかどうかの試金石でもあります。厳しい言い方をすると、**振り返りを予定として確保することすらできない人は、スタートラインに立てていません**。まずは予定として確保できるだけの余裕なり覚悟なりモチベーションなりを手に入れてからです。

外部の力を借りてスタートラインに立つ

ひとりで振り返りを（予定として確保して）行う力がない場合、外部の力に頼るといいでしょう。

かといってプロジェクトタスク管理のようにプロジェクトの振り返りがしたいわけではありません。やりたいのは自分の、自分による、自分のための振り返りです。ひとりで行うことは欠かせません。ただ、ひとりだけだとスタートラインに立てないので、外部の力を借ります。

借り方は二つあります。

1: 集中しやすい場所に出かける

一つは集中しやすい場所に出かけることです。カフェや会社やコワーキングスペースで仕事をする方は多いと思いますが、まさにそうです。振り返りという予定を行う用の場所をどこか見繕って、そこに出かけます。

必要なインプットは持参しましょう。事前にテキストや写真などで集めておいて、出かけた先でPC やスマホで見ながら考察するとやりやすいと思います。紙などアナログなインプットを使いたい場合は工夫して持参してください。

場所はどこでも良いですが、人目をどれだけ気にしないといけないか次第です。人目が気になるか、見られると困るインプットや考察内容があるなら、カフェや図書室・自習室などオープンな場所は避けて個室にしたいです。場所というと、飲食店やレンタルスペースを思い浮かべがちですが、意外と色々あります。それこそ人気（ひとけ）の少ない公園の東屋でも可能ですし、トリッキーな選択肢としてはカラオケやラブホテルもあります。

後者、ラブホテルは元々数時間の休憩をベースとしたサービスであるため、ホテルのような宿泊ほど高い料金はかかりません。近年ではラブホ女子会の利用例もありますし、同伴者が後から来るケースも想定されており元々ひとりでの利用も可能です。カラオケよりも高額ですが、防音性が高く、広くて、トイレも楽ですしベッドに寝転ぶこともできたり、と快適に過ごせます。ただホテルによってはピンキリなので下調べはしておきたいところです――

と、あえてラブホテルの解説をしましたが、場所の選択肢を工夫できることを強調したかったのです。また筆者は自転車で近所の人気（ひとけ）のない山道に出かけて、自然の音を BGM にしながらのんびり振り返ることもあります（雨季夏季冬季はさすがに厳しい）。すでに述べたとおり、振り返りは予定のように扱って集中してこそです。ひとりでできないなら、場所を模索する手間をかける価値はあります。

2: もくもく振り返り会

もくもく会 は 2008 年頃に pha さんが自身のブログで紹介したもので、各自もくもくと作業するための集まりです。

集まりというと通常は何らかの目的があって、皆が活発に交流・連携しながら達成していくようなニュアンスがありますが、そうではなく、もくもく会は単に集まるだけです。会場に集まって、適当に席をキープして、いつものようにもくもくと作業するだけ。下手をすると参加者同士の挨拶すらなく、一言も喋らず帰ることすらあります。現在もくもく会について検索すると「勉強会」のようなニュアンスが出てきますが、元々はそうではなく、本当にただ集まって各自作業するだけでした（と思います）。ニュアンスとしては **突発型の自習室** です。「今週この日時にこの場所でもくもくしようぜー」といった形で自習室をつくる・募集するイメージですね。自習室ですから、むしろ喋る方が敬遠されます。どうしても喋りたいならメインの会場から出てからです、あるいは事前に参加者間で喋ってもいいキャラですよと公言してやりたい人同士でやります。

さて、二つ目のやり方は、このもくもく会において振り返りを行うというものです。振り返り限定のもくもく会でもいいですし、何も限定しないもくもく会を開催して自分は振り返りを行う、でも良

いです。重要なのは自習室のニュアンスを厳しく守ることです。少なくとも自習室内での私語厳禁は必須にしてください。自習室という誰にも干渉されない聖域は必ず守ってください。これができないなら正直やる意味はありません。

もくもく会の開催は自分で行ってもいいですし、どこかのイベントに頼ってもいいです。ただし近年は自習室のニュアンスを持つものが少ないですし、そもそも開催自体がレアなので自分で開催するしかないと思います。ネットで募集してもいいですし、身近な同僚や友達や家族を誘ってもいいでしょう。個人的には、親しい間柄だと私語厳禁を守るのが難しい（ファミリア・デバフは前の章で取り上げました）ので、赤の他人を集める形態が良いと思います。

「そんなことをして意味があるのか」と思われるかもしれませんが、あります。方法 1 はどこかの場所に出かけるものでしたが、本質的にはこれと同じです。本項の方法 2 は、いわば場所を（もくもく会という制約をつけて）つくっているだけです。場所という外部環境に頼る点は変わっていません。場所に頼ることの効果はワープの項でも述べましたが、改めて述べると、自宅などいつもの場所と違うことで誘惑が無くメリハリがつくことや、人がいるので注意が上手く分散して集中しやすくなる、というより気が紛れづらくなる等です。

振り返りの設計

ここまで解説したとおり、振り返りはその場の思いつきですぐに始められるものではありません。事前にどんな振り返りをどのように行うかをある程度設計しておいた方が機能しやすいです。

設計のやり方ですが、正解はありませんし、要は上述した PROC や DWMY を自分なりに決めればいいだけですが、参考までにやり方のいくつかをここで解説します。

Enum and Assign 方式

やりたい振り返りを列挙した後、それぞれについて、いつ行うかを考えて配置するというものです。

- 1: 何の振り返りを行いたいかを洗い出す
  - たとえば「仕事の進捗」「日々の健康」「趣味のFPSゲームの戦歴」「現在のキャリア」であれば4対象
  - これを **振り返り対象** という
- 2: 振り返りを行う枠（予定）を取る
  - DWMY を意識して定期的な予定として取る
  - 一回で終わらせる場合でも予定として取る
  - 1で洗い出した対象各々に対して行う
    - 6個あるなら6予定分を設定することになる
- 3: 各振り返り対象のインプット、考察、アウトプットを設計する

- どんなインプットを使うのか
- 考察中は何を行うのか
- アウトプットとして何を出すのか

メリットはわかりやすいことです。洗い出して、枠をとって、準備して、と愚直にやっているのでわかりやすいです。

デメリットは大変なことです。愚直に行う分、作業も多いですし、一度決めた振り返りが未来永劫有効に続くはずもなくて細かいチューニングや試行錯誤は何度も起きます。

振り返りアワー方式

大学では先生がオフィスアワーという「生徒からの質疑に応答するための時間」を設けることがあります。これの振り返り版と考えてください、つまり、事前に振り返りを行う時間を取っておいて、当日時になったら何かしら振り返りを行います。

オフィスアワーと違うのは、登場人物が自分ひとりであることです。何の振り返りをどのように行うかはその都度自分で考えます。

シンプルややり方ですが、軽く試したいならおすすめです。実は難しいのは何するかを考えるよりも、振り返りアワーという予定をちゃんと取るところだったりします。スタートラインの話を前述しましたが、スタートラインに立つ（ちゃんと振り返りのためだけの予定をつくる）こと自体がそもそも難しいのです。現代人はただでさえ忙しいですし、振り返りという効用もわからなければ個人的で面倒くさそうな営みにいきなり時間を費やせるかというと、心理的に難しいのです。

だからこそ試金石としても機能する、シンプルながら有効なやり方です。

Diary Based DWMY

毎日日記を書くことと、DWMYヒエラルキーに基づいた振り返りを何か行うこと、の二点を守るやり方です。

まず日記については何でも構いません。業務日誌のようなビジネス寄りでもいいですし、X につぶやくだけのカジュアルなものでもいいです。もちろんアナログな手帳に書いたものでもいいですし、デジタルのノートに入力したものでも構いません。何でもいいですが、できればこまめに書き込みたいです。一日の終わりにまとめて書こうとしても、あまり細部を思い出せません。というより、**一日の終わりにだけ書くというやり方は、それ自体が振り返り** になっています。それで通用するならそれでいいですが、資質を持つ人にしかできません。誰にでもできるものではありませんし、通常はできません。だからこそこまめに記録を取る（インプットのための情報をつくっていく）ことが必要で、それを「こまめに書き込む」と表現しています。

次に振り返りの頻度ですが、DWMYヒエラルキーに基づいて行いたいです。つまり毎日行い、

毎週行い、毎月行い、毎年行いたいです。D、W、M、Yのすべてが苦痛ならDWだけ、WMなどアレンジしてもOKです。とにかく、こまめに書いた日記を使って何かしら振り返ります。

最後に振り返りとして何を行うかですが、自由で構いません。アウトプットとしてはタスクとモットーが期待されていますので、まずは「次やりたいこと」と「次から心がけたいこと」の二点を出すことを目指すといいでしょう。日記（あるいはDWMYヒエラルキーに基づいた下位のアウトプット）を見ながら考えたりひねりだしたりするイメージです。たまにひらめきや思いつきが出てくることもありますが、そういうものこそ重要だったりするので、逃さず捉えてください。

最終的に何を掲げるかは自分次第ですので、無理なく掲げましょう。まずは「次やりたいこと」を一つだけ、くらいにハードルを落とすことをおすすめします。最初から複数個を掲げてもまずそのとおりにならないですし、一つずつであってもすぐにたくさん溜まって収拾がつかなくなります（1日1タスクをアウトプットしたとしても7日経てば7個になる）。重要なのは、無理なく一つずつ、少しずつ前身していくことです。

たとえば1日1タスクだけアウトプットするとした場合、一週間経てば7タスクになります（前日と同じタスクを続けたならばもっと減る）。このままだとタスクがたくさん溜まるわけですが、これらをどうするかはWeekの振り返りで検討できます。最も重要なタスク一つだけを続けようとか、どれも微妙なので全部捨ててしまうかなどと判断すればいいわけです。正解はないですが、ちゃんとやれば少しずつでも前進していけます。仮に一週間中に1タスクしかできなかった場合でも前進ですし、何なら1タスクすらできなかった場合でも「1タスクすらできなかった」という実績があるわけで、実績を得たという意味では前進です。

5分振り返りんぐ

振り返りんぐとは振り返りに-ingをつけたもので、単に振り返ることです。5分とあるので、5分で振り返りを行うと想像できます。

5分振り返りんぐとは、きりのいいタイミングで突然5分の振り返りを行うことです。ポイントは大げさなロールプレイで、「5分振り返りんぐのお時間でーす！」「さてさて今回の振り返りはどうしましょう？」「そうですねぇ、今日は……」のように**大げさなに一人二役（三役以上でもいい）を演じて会話します。**

馬鹿らしく聞こえるかもしれませんが、演じることで脳の使い方が変わるので意外と色々と思い浮かんだり思いついたりするものです。プログラマーの世界では「テディベア効果」「ベアプログラミング(Bear, 熊のぬいぐるみ)」なるプラクティスがあります。プログラムのバグが見つからない、どうしようってときに人に相談するかわりに**ぬいぐるみに相談する**というものですが、これで案外原因や解決策をひらめいたりします。ひとりで考えるときと人に説明するときとでは脳の使い方が異なるので、前者ばかりしてて思いつかなかったことが思いついたりするのです。後者の使い方は、一人で演じるだけでもある程度できるのです。

振り返りが個人的な営みであることはすでに述べました。誤解を恐れず言えば、振り返りは個人的かつ創造的な営みでもあります。何をインプットにし、何を考察して何をアウトプットするかが自分次第だからです。正解もありません。しかし、プログラミングやその他仕事とは違って他者においそれと相談できるものでもありません（※1）。だからこそ難しく、詰まりやすく、それゆえ頼れるものならたとえ演技だろうと頼りたいのです。

- ※
  - 1 相談できるのならもちろん相談した方が良いですし、その方が捗ります。ただし本章では、そして本書自体が一人で自立的にこなせることを目指すテイストとなっていますので、このようにひとりで何とかしようとするアプローチを取りがちです。これは別に意固地になっているのではなく、タスク管理が本質的に個人的なものだからです。個人的なものは無闇に人に見せられないですし、自分で何とかしない限りはスッキリしません（他者に正論言われたりするとイラッとしたりしますよね）。筆者の持論にすぎないかもしれませんが、私はタスク管理、特に個人タスク管理とは、いかに自分で自分と向き合うかの営みだと考えています。ひとりで向き合うことからは逃げられませんし、逃げずに向き合うことこそが重要である、とは本書で伝えたいメッセージの一つです。

なぜ振り返るのか

振り返りに必要な道具は一通り揃えましたが、肝心の動機面にまだ言及していません。振り返りのPROCのPはPurpose/目的です。そもそも振り返りの目的は何でしょうか。なぜ振り返るのでしょうか。

納得できる目的を持っていないと、そもそも振り返りが成立しません。ここまで話したように、振り返りは面倒くさいものですが、形骸化させないためにはそれなりのモチベーションが必要不可欠です。もちろん、どんな目的が刺さるかは人それぞれですし、同じ人でも状況によって違います。

そこで本項では主な目的を俯瞰的に捉えることで、読者への気づきを促すことを狙います。

振り返りの目的は以下の3つに大別できると思います。

- 知るため(To Know)
- 進むため(To Advance)
- 守るため(To Protect)

知るため

知るためとは「知りたい」です。

過去自分が何をしてきたかとか、状況はどうなっているかといったことを単に知りたいから振り返ります。本章の定義上はアウトプットが必要なので、単に知りたいだけでは不適切に聞こえますが、実は問題ありません。むしろ無理してアウトプットをつくる必要はありません。振り返りは成立させること自体が難しい営みですから、まずは成立させたいのです。アウトプットの無い、ただ知るためだけの活動であろうと、それで続くのならばしめたものです。アウトプットは続いた後からつくるようにしても遅くはありません。

というわけで、もし単に知りたいという気持ちが芽生えているのなら、ぜひとも大切にしてください。その気持ちを満たすために振り返りを始めてみる、それこそ遊んでみるくらいのつもりで構いません。まずは始めることが第一です。

好奇心という言葉がありますが、これは自分の過去についても当てはまります。一方で、自分の過去なんて全く興味がないし、振り返りなんて絶対つまらないでしょと考える人もいます。むしろこのタイプが多数派かもしれません。その場合は別の目的を考えましょう。合う合わないはあります。

進むため

進むため とは「前進したい」です。

前進するためには正しい努力が必要です。課題と現実を正しく認識（わからないなら仮説的に設定）して、そのギャップを埋めるように微調整していく、またそもそも足りない武器があればその入手と習得から始めます。ゲームやスポーツ、あるいは教育や指導や OJT のようにつきっきりで学ぶ場合はわかりやすいですが、残念なことに生活の大半はそうではありません。そもそも課題も現実もまともに認識できていません。感覚でこなしていける感覚派もいないことはないですが、レアでしょう。というわけで、生活の大半を対象に、より前進したいのであれば、課題と現実を把握する必要があります。

ここで振り返りが使えます。現状をインプットし、何かしら考察を加えて、次の行動や指針をアウトプットする。そしてそのアウトプットを使って行動して、またどこかのタイミングでもう一度現状して――と、繰り返します。このような営みはしばしば「サイクル」または「ループ」と名付けられ、PDCA サイクルや OODA ループは比較的有名です。

守るため

守るため とは Minimize Negatives です。小さなネガティブが慢性的に発生するのを避けるため、あるいは大きなネガティブが生じるのを未然に防ぐために、日頃から手間暇をかけます。ネガティブを Minimize（最小化）するわけですね。

この目的でも振り返りは重宝します。仕事にせよ私生活にせよ、あるいは長期的な趣味の進捗などでも構いませんが、定期的に状況を振り返ることで最悪に備えることができます。

生活で言えば、健康診断がわかりやすいと思います。あとあと大事にならないように、たとえば年に一回くらい健康診断を受けて、その結果を振り返るわけです。もっと頻度を上げたければ体重を測定したり、睡眠時間をアプリで日々モニタリングしたりしますよね。パンデミックになってからは一時期温度の測定も流行りましたし、今なお習慣として続けている人もいるのではないでしょうか。

より高頻度な振り返りでいえば、忘迷怠がわかりやすいでしょう。忘れたり迷ったり怠けたりを減らすことはタスク管理の主目的ですが、実は振り返りでもある程度は行えます。記憶と一緒で、少しでも意識したり考えたりすれば思い出しやすくなります。仮に「毎日5分だけ今日の仕事を振り返る」という習慣を入れたとして、これはとてもシンプルな振り返りですが、これだけでも何もしないよりはだいぶ違ってくるはずです。当たり前のことを言っているように聞こえますが、そのとおりです、振り返りという言葉を使って無理矢理丁寧に説明しているだけで、内容自体は至極シンプルなことです。ただし、すでに述べたとおり、この5分を確保して実際に振り返りを行うというところが本当に難しいのです。しつこく言いますが、守るためには腰を上げねばなりません。

ヒント集

ここからは振り返りの各ステップ――インプット、考察、アウトプットの検討に使えそうなヒントを取り上げていきます。

インプットのやり方

すでに述べたとおり、やり方は大別して 3 つあります。

- 1: 記録を見る
- 2: 記憶を思い出す
- 3: ヒントを借りて思い出す

1 の記録については、計測値のような定量データから日記みたいな定性的記述まですべてが対象です。事前に仕込んでおくか、まめに書き込んでおくことが必要になります。もっと言えば、振り返りを事前設計し、何を記録しておけば考察時に使えそうかを事前に練っておいて、日々記録しておく――というレベルまで必要です。ただし、いきなり記録対象や記録方法を絞るのは難しいので、まずは適当でもいいのでこまめに記録する仕組みや習慣を整えるのが先決です。

2 の記憶については、記録ではなく脳内の記憶をインプットにします。人の記憶は信用できませんし、そもそも脳内で何とかできるなら振り返りなんてしなくても済む（その場で対処できるか、やろうと思えばできる）からです。ただし直近印象的なことがこびりついていて、どうしてもこれを何とかしたいという場合は、何とか機会をつくって集中的に、あるいはリラックスして向き合うと良いでしょう。

3 のヒントについては、1 のように日頃の記録から忠実にアウトプットを出すというより、もっとざっくりとしたヒントを見て、思いついたことをベースにアウトプットするイメージになります。このヒントをインプットとするには、どうやって集めればいいでしょうか。筆者は一時期、定時退社で帰宅後フードコートで外食をしながら、行き交う人々を眺めていましたが、これはその場の人々をヒントに自分の人生を振り返っています。たとえば高校生の楽しそうな集団を見ながら「ああいうのを取り戻したいか？」「いや、もうそういうノリが楽しめる年じゃないよなぁ」「今は一人で楽しむ方向を選んだし、もうちょっと踏ん張ってみるべきでは」などと自分なりに思考します。これはヒントが毎回変わるものであり **動的なヒント** と呼びます。動的なヒントは、集めるというよりも、それが起こる場所に行くことで享受できます。浴びるインプットと言えるでしょう。次に、これから後述するヒント達は、考察のために事前に（あるいは考察時でもいいですが）用意するもので、一度用意したら変化せず使い続けられるため **静的なヒント** と呼びます。静的なヒントは誰かがつくったものを集めるか、自分でつくることで手に入れます。集めるインプットと言えます。

これらヒントの使い分けですが、基本的には静的なヒントを集めた方が良いです。振り返りは何回も続けて少しずつ要領を得ていくものなので、ヒント自体も固定した方がいいのです。一方で、そればかりだと味気ないですし、思考が膠着したりもしますから、たまには動的なヒントを浴びてみるとバランスが取れます。

考察時に使えるヒント

考察のやり方は様々ですし、必要ならいかようにも設計できますが、本書では割愛します。かわりに考察時に使えそうなヒントとしてフレームワークとトリガーリストを紹介します。

振り返りフレームワーク

近年はチームの時代であり、チームを対象とした仕事術は無数に存在します。振り返りもその一つで、様々なフレームワークがあります。

postalk さんの記事がちょうど良いと思いますので、ここで紹介します。いずれもチームで行うものですが、個人で振り返る場合でも参考にできる部分はあるはずです。

- 振り返り手法10選！アジャイル開発に使える、海外の面白い手法もご紹介 | postalk park

詳細は記事を参照してください。ここでは一部をかんたんに取り上げます。

KPT や YWT は比較的知られていると思います。KPT は「Keep 続けたいこと」「Problem 抱えている問題」「Try やりたいこと」を挙げ、YWT は「Yatta やったこと」「Wakatta わかったこと」「Tsugiyaru 次にやること」を挙げます。個人で使う場合は YWT の方がやりやすいでしょう、やったとかわかったといった観点は雑にでも洗い出せそうです。一つ注目したいのは、どちらも次にやることを挙げさせる点です。本章でも振り返りにはアウトプットがあり、タスクとモットーを挙げよと書きました。次の行動を決めることは大事です。

mad, sad, glad や WRAP の A（Appreciations/感謝したいこと）あたりは感情を扱っており、振り返りにおいて感情をインプットにしてもいいとの示唆が得られます。振り返りに正解はありませんし、私たちは人間であり感情の影響は避けられませんから、ちゃんと向き合うのももちろんアリです。たとえば日々感情を記録していって、どういうときにハッピーになるのか、またアンハッピーになるのかの因子を特定できたらQoLが上昇しそうです。

記事の後半はリーンコーヒーやマリオカートなど、どちらかと言えば議題や論点をうまく可視化・分類して議論をサポートする方向のフレームワークが目立ちます。この考え方は振り返りにおいてもまさに有効で、何を振り返るか対象を定めたり、どこまで考察したかを可視化したりといった作業に役立ちます。記事は postalk というホワイトボードツールの宣伝記事ですが、まさにホワイトボードも重宝します。Miro など他のツールもありますし、付箋とペンで机上で行うこともちろん可能です。

トリガーリスト

本書では連想リストとして説明していますが、オリジナルのトリガーリストとは GTD における概念で、頭の中のもやもやや気になることを外に出すための自問自答集を指します。このトリガーリストは、振り返りにて考察を行う際の観点として重宝するかもしれません。ネット上にいくつか例があるので紹介します。

- 【2018年版】私なりの『GTDトリガーリスト』- BrownDots
- 【コピー可能】教師のためのトリガーリスト – Google Workspace for Educationつうしん by パイダゴーゴスつうしん
- GTDトリガーリスト | PDF
- 主婦のためのGTDトリガーリストを作ってみた ミセス・かんちがいのブログ日記
- 超自分流トリガーリスト！ - めい記

こちらについても詳細は各記事を見ていただくとして、ここでは筆者なりに本質をまとめます。

まず観点の大カテゴリなるものの存在が目につきます。記事を眺めると「仕事」「プライベート」「趣味」「お金」「健康」「家族」あたりがあるでしょうか。筆者個人としては「住まい」「結婚」「常識」などもよく使います。これら大カテゴリを見ると、色々思い浮かんでくると思います。特に日頃おざなりになっているものがあるなら、良い機会ですので振り返りの対象にしてみましょう。仮に出会いや結婚がおざなりになっているなら、週一くらいで「結婚どうしましょ振り返り会」を開催して、現状はどうなのか、自分はどうしたいのか、次何をするのかといったことを少しずつ進めていきます。すでに振り返りを予定として扱った方が良いとの点は述べました。予定として確保してしまえば、普段忙しくてもその予定の最中は振り返りに集中できるはずです。たったの1時間、いや30分かもしれませんが、それだけでも積み重ねるとじわじわと効いてきます。

次に自分がよく関わっている分野に関しては、さらに細分化できることがわかります。教師のためのトリガーリストがわかりやすいですが、担任クラス、学年団、分掌・委員会などのカテゴリが設けられています。ここは人によって異なるでしょう。あるいは単純にやるなら抱えている案件や

所属しているチーム、主なステークホルダーなどで設けることができるはずです。それらを眺めることで、振り返りたいことが思い浮かんでくると思います。

もう一つ挙げるなら、各項目は MSCW で分けられそうだともわかります。MSCW とは Must（やらねばならない）、Should（やるべき）、Could（できる）、Wouldn't（やりたくない）です。つまり、やらねばならないことをやるための振り返りとか、やるべきことをやるための振り返り、やりたくないことを避けるための振り返りなど前進する系の振り返りを考えるのに使えます。

振り返りの4Rマトリックス

振り返りは「何に関してか」と「主観と客観のどちらを重視するか」の二軸で 4 パターンに分けることができます。これを **振り返りの4Rマトリックス** と呼びます。どのパターンを使うかはケースバイケースですが、それ以上に好き嫌いもわかります。振り返りは（本章の立場では）個人的なものであり、続いてこそですので、まずは好き嫌いで選んでみることをおすすめします。

マトリックスは以下のとおりです。

	自分に関して	他人や活動や仕組みに関して
客観重視	1 Reassessment	2 Review
主観重視	3 Reflection	4 Recollection

客観とは記録やデータやエビデンスや他人の声、主観とは記憶や思いつきや推測です。マトリックス上で「重視」と書いているように、両者の配分はグラデーションです。客観100%で主観0%といった偏りはありません。客観を重視するか、主観を重視するかになります。

各パターンに入りましょう。1 の Reassessment/リアセスメントは、自分に関して客観的に振り返ります。たとえば健康やお金を管理していて、普段アプリなどで記録しているデータを見て振り返りたい場合がこれに相当します。自分について定量的に振り返るわけですね。その分、定量的なデータの記録は事前に整備するか、律儀に行わねばなりません。

2 の Review/レビューは、自分以外の何かに関して客観的に振り返ります。仕事の文脈だと基本的にここになると思いますし、潮流としてもデータドリブンだとエビデンスだのファクトだのといった言葉はよく聞きます。なので、ここをやっておけば間違いはないです。ただ面倒くささもダントツで、仕事でもなければ進んでやりたくはないでしょう。要はその「何か」に関するデータを集めないといけないのです。1 のリアセスメントは自分に関してだけなのでどうとでもできますが、レビューはもっと大変です。そもそも事前にデータとして記録されているとは限らないし、記録させるのも通常は難しいので、人におうかがいを立てるケースも多くなります。実際にすでに紹介したフレームワークも、チームでその場で話し合うことが前提となっています。原始的ですがそうするしか現実的な方法がないのです。さて、個人で行う振り返りにおいて、この手間を負えるか（振り返りを継続的に行うなら継続的に負えるか）というと、たいていはノーなので、実際は主観を増やさざるをえず後述の 4 になりがちです。

3 の Reflection/リフレクションは、自分に関して主観的に振り返ります。リフレクションという言葉自体が多義語ですが、「内省」の二文字が似合う言葉だと思います。主観で良いが、その代わり深く考えて向き合うイメージです。実際そのとおりで、深く向き合うつもりでやらないと何も出ません。振り返りの設計でもいくつかやり方を挙げていますが、自分と向き合うことを要求するテイストだと思います。

4 の Recollection/リコレクションは、自分以外の何かに関して主観的に振り返ります。要は他人なり活動なり仕組みなりを主観で捉えて考察します（リコレクションの原義は「回想」や「想起」）。ビジネスの潮流が 2 のデータドリブンであるように、このリコレクションはあまり芳しくないパターンに聞こえがちですが、そもそもはっきりとしたデータを得られる方が少ないです（※1）。現実的にはリコレクションに頼ることが多くなります。もちろん主観が正しいとは限らないので、後々検証が必要です。そういうわけで仮説検証のサイクルやループにならざるをえず、振り返り自体も定期的に続けていくことになります。この仮説検証が大事で、これをしないとただの決めつけや憶測に終始してしまいます。リフレクションであれば自分のことなのでそれでもいいですし、なんだかんだ自分のことを一番知っているのは自分なので納得感も出やすいですが、このリコレクションは自分以外の何かを自分の主観で捉えるという話なので、せいぜい出来の悪い仮説にしかなりません。（決めつけておしまいにしたくないのなら）仮説検証の営みは必須です。

- ※
  - 1 データに頼ることも不可能ではないですが、難しいです。昨今ではデータドリブン経営などデータに頼ってビジネスをもっと上手くやる潮流が目立っています。またリーンスタートアップでも仮説検証を行うためにデータ項目の定義や集め方の仕組みにかなり力を入れます。両者に共通しているのは、事前にちゃんと設計し、できれば IT の力で自動で記録できるように整備することだと筆者は考えます。どちらも専門の人材が必要になるレベルで高度な仕事です。ここまでしなければ役に立つデータは得られないのです。逆に、ここまでしていない場合は、データの量や質が悪かったり、整備の手間がかかりすぎたり属人性が高かったりして形骸化したり政治化したりします。

アウトプットのやり方

アウトプットとしてタスクとモットー、できればコンテキストを出すのでした。どうやって出せばいいのでしょうか。

タスクを出す

考察を上手く行えているのなら、次に行うべきタスクも自然と出てくるはずです。出てこない場合は考察が足りないか、インプットが足りない可能性が高いです。それでも出てこない場合は、仮説でもいいので何かしら定めます。あるいはあえて何も定めず様子を見る、でも構いません。

タスクとしては「即席のタスク」と「ルーチンタスク」を使い分けると良いでしょう。即席とは一回限

りのタスクで、一度やったらおしまいというものです。ルーチンタスクとは指定の頻度（1日1回とか3日に1回など）で継続的に行うタスクで、終わるまでに時間がかかるか半永久的に続くことが多いです。まずは即席のタスクをアウトプットして、どれをいつ行うかをやんわり設定します。ただリストアップしているだけでは先送りするだけなので、予定に仕込むとか、（人が絡むタスクなら）初動のコンタクトをもう送ってしまうなど、腰が上がりやすい仕掛けをつくった方が良いです。そして、即席だけだと足りずに「続けてみないとわからないなー」という場面が多いので、そんなときはルーチンタスクをつくります。詳細はリンク先を参照してください。

いずれにせよ、自分の生活に「振り返りでアウトプットしたタスクが新しく追加される」構図となります。新しいタスクが増えるわけです。普段からタスク管理をしていないと対処できないでしょう。そもそも生活が忙しすぎる場合は対処する暇さえもありませんので、その程度の余裕はつくることが先決となります――というふうに、結局はタスク管理が必要になってきます。もし振り返りでアウトプットしたタスクをこなせない、というのであれば、まずはご自身のタスク管理と余裕を整備するところから始めるべきです。

一つの目安は、**振り返りでアウトプットしたタスクは、次の振り返りまでに完了させておくか、少なくとも一度以上着して何らかの進捗や気づきを得ておく**です。アウトプットしたタスクは、いわば負債（タスクデット, Task Debt）であり、次のタイミングまでに返せないと（着手できないと）どんどん積み上がっていきます。TODOリストが膨れ上がる現象はよく知られていますが、あのようになります。借金と違ってペナルティはありませんが、だからこそすぐに積み上がってしまいます。**タスクデットが積み上がらないバランスを心がけてください**。積み上がる場合、タスクのアウトプット量を減らした方がいいかもしれませんし、振り返りの頻度を落とした方がいいかもしれません。あるいは単に考察が甘くてタスクが大きすぎるのでもっとスモールスタートできそうな小さなタスクを出すのが良いかもしれません。

モットーを出す

モットーについてはモットーの章を参照してください。

振り返りの文脈では、以下の 2 点を使い分けると良いと思います。

- 1: ある場面において気をつけるべき「限定的なモットー」を出す
- 2: 人生全般において心がけたいモットーを出す

1 については、たとえば仕事に関する振り返りで「次からは A さんには高頻度で声かけてフォローするようにしよう」とのモットーを導くといったことです。これは仕事の、A さんに関するモットーであり、限定的なものです。今すぐ何か行動できるものでもないので、タスクではなくモットーとして出しているわけです。ただ、この場合は「A さんが了承してくれるのなら週一で 1on1 ミーティングをしよう」などタスクに落とすこともできます。モットーの章でも述べましたが、モットーをモットーのまま抱いておいて応用できるタイプ（アプライア）と、応用は苦手だがタスクや習慣をつくって律儀に行動を続けられるタイプ（アスリート）があるので、自分に合ったやり方を使いましょう。

2 については、「人生振り返り会」「来月以降どう立ち回るか会議」などモットーを出す用の振り

返りを開催することで出します。そもそもモットーとは、1 のような限定的なものよりも、哲学や信念や性格といった深いレベルのものです。そしてそれは日々漫然と（あるいは多忙に）過ごしているだけでは出てきませんので、捻り出すための機会をつくる必要があります。上述した 4R でいえばリフレクション――つまりは内省的に振り返るのがおすすめです。このとき、他の人や組織のモットーが役に立ちます。モットーの章ではエモットー（External Motto）と書きましたが、外のモットーは参考になるのです。しかしモットーは個人的なものであり、自分なりに最適化が必要です（イモットー、Internal Motto）。つまりエモットーを参考にしてイモットーをつくりあげるのです。これにはリフレクションが最適です。というより、客観的な情報を使っているだけではイモットーは導けません。他人の人生を生きるだけの奴隷になってしまいます（※1）。自分はどうなのか、を導くためには内省が不可欠です。

- ※
  - 1 他人の言葉や客観的なデータのみに頼る生き方が悪いと言っているわけではありません。むしろ自分のくだらない信念やプライドに依存しない分、柔軟で融通が利きやすいです。ただ自分がないのでふらふらしがちか、少なくとも多忙になりがちです。度がすぎるとかえって騙されやすくなります（詐欺やカルトが手強いのはこのためです）。じゃあ本文のとおり、内省を重視して「自分」をつくっていけばいいかというと、これはこれで視野が狭くなったりプライドが強化されてがんじがらめになったりするリスクがあります。要はバランスなのですが、難しいところです。かくいう筆者もまだまだです。このような話題で思い出すのは、千葉雅也さんの「勉強の哲学 来たるべきバカのために」です。同書では「何らかのノリにのるしかない」「自身の享楽的なこだわりに基づいてのる先を決めるのが良い」「ただ変化できないと固執にしかならない」「変化するためには勉強が要る」「勉強しろ」のようなことが書かれています（いると思います）。ここで勉強とは学問を学んで多様な世界観やメンタルモデル、そもそも知識を得るニュアンスだと理解しました。だとすると、「学問を学ぶ」は同書から導けるエモットーの一つです。これをどう取り入れてイモットーにするかは自分次第で、かつそのためには内省して深く向き合い深く考えることが必要です。そうしないと「ふーん」とか「いやまあ頭ではわかっているけどさ……」で終わってしまいます。

コンテキストを出す

コンテキストについてはコンテキストの章を参照してください。

振り返りにおいては「ロードコンテキストを出す」という意味です。ロードコンテキストを出しておけば、次の振り返り時にそれを見ることで前回の文脈を思い出せるため、コンテキストスイッチングの負担が小さくて済みます。

特に「なぜこのようなタスクやモットーを出したのか」という根拠や理由、あるいは妥協を含むならその旨も書いておくことがポイントです。これらを書いておくことで、次の振り返りのときも参考にできるからです。前回の文脈も参考にしながら振り返りを続けていくことで、次第に隙をなくしていくことができます。逆に、文脈を踏まえずに毎回場当たり的に振り返りをするだけでは、その場限りを脱せません。振り返りの醍醐味でもある「本質が見えてきた」「なんとなくわかってきた」と

いった感覚にも恵まれません。この感覚は面白さであり、快感でもあって、振り返りを続けられるかどうかの分かれ目にもなっています。

意識としては、**未来の自分という他人** に引き継ぐつもりで臨むといいでしょう。未来の自分はもはや他人のようなものであり、他人ゆえに自分のことなんて伝わりませんし覚えてもいません。だから丁寧に情報を残しておいて、伝えてあげます。つまりアウトプットとしてコンテキストを残しておいて、次に振り返るときの自分（未来の自分という他人）に役立ててもらうのです。

ただし、すでに述べたとおり、コンテキストをアウトプットすることは努力目標です。コンテキストを出す部分が辛くて振り返りが続かないくらいなら、コンテキストは出さない方がはるかにマシです。まずは振り返りを続けることが大事です。コンテキストを出すのは、振り返りの習慣が定着してからでも遅くはありません。

まとめ

- 振り返りとはインプットを元に考察を行い、アウトプットを出すこと
  - インプットは記憶か記録
  - 考察は何でも良い
  - アウトプットはタスクとモットーとコンテキスト
    - アウトプットにより次に繋げるのが必須。考察だけでは意味がない
- 振り返りは個人的な営みであり、正解はない
  - 各自で最適なやり方とバランスを自分で模索する
  - 最初は雑でいいし、アウトプットも一つでいい
    - 習慣と同様、一つずつ・少しずつ増やしていく
  - まずは続くことが第一
- 振り返りの諸性質
  - PROC
    - 目的、想起（インプットの集め方）、機会（いつ振り返りか）、継続（一度の開催で終わらせるか）
  - DWMY
    - 振り返りは Day, Week, Month, Yearの定期性で行うのが良い
    - 一つ前のアウトプットを使うとインプットの手間を抑えられる（DWMYヒエラルキー）
  - KAP
    - 振り返りの目的は3つに大別
    - Know（知りたい）、Advance（進みたい）、Protect（守りたい）
  - 4R
    - 「自分/自分以外の何か」を「主観で/客観で」振り返る、で4パターンある
- プラクティス
  - 振り返りは予定として扱う。予定のつもりでしっかり取り組むべきもの
  - ひとりで開催や継続ができない場合は、外部の力に頼る（スタートラインに立つ）

- 振り返りの設計のやり方して Enum and Assign 方式などいくつか紹介
- 考察時は振り返りのフレームワークやトリガーリストなどを使うと便利

=== Chapter-21 タスクの属性 ===

タスクには様々な属性があります。たとえば「名前」「開始日」「終了日」「優先度」「担当者」「タグ」などはすべて属性です。タスク管理的にはタスクとはnの属性を持つ項目と言えますし、属性を使ってソート（並び替え）やフィルタリング（抽出）も行えます。そもそもツールや手法ごとにどの属性を使うかが異なっています。

と、このようにタスク管理と属性は、実は切っても切り離せないものです。通常、意識することはありませんが、もし自分でタスク管理をつくったり、ツール上でカスタマイズしたりするのであれば知っておくと便利です。

本章ではタスクが持つ属性をリファレンス的に取り上げます。

タスクの 3A

タスクは以下の 3 つから構成されます。

- Attribute/属性
- Arrow/関係
- Attachment/機能

本章では属性について扱い、次節から解説します。本節では残る関係と機能について軽く触れておきます。3A を紹介するのにちょうどよいタイミングだったのでここで取り上げます。属性とは関係がないため、興味なければ読み飛ばしてください。

関係

関係（Arrow） とは他のタスクと関係――包含、参照、依存を扱ったものです。英訳が特殊に見えますが、矢印（Arrow）が向いているからだと考えてください。3A で揃えると覚えやすいため、少々強引ですがそうしました。

さて、関係はさらに 3P でまとめることができ、Parent-Child（親子関係）、Point（ポイント関係）、Pre and Post（依存関係）があります。

親子関係 は最も身近な関係です。親タスクに対して子タスクがぶら下がっているとか、タスク A にサブタスク A1、A2、A3 が存在するとかいった言い方をします。また粒度の面でも登場するこ

とがあり、たとえば「引っ越しをする」という粗いタスクは、分解すると「引越し先を洗い出す」「次の引越し先を一つ決める」「引越し業者を洗い出す」「引越し業者を決める」――といったようになるでしょう。

- 「引っ越しをする」
  - 「引越し先を洗い出す」
  - 「次の引越し先を一つ決める」
  - 「引越し業者を洗い出す」
  - 「引越し業者を決める」
  - ……

これは 4 つの子タスクをつくったことに等しいです。このように親子関係は、タスク管理を知らない人であっても自然と行き着くくらいには自然なものです。網羅性を出したいときには重宝しますが、愚直で泥臭いため負担も大きいです。仕事以外ではやりたくない人も多いでしょう。

次に **ポイント関係** ですが、これはリンクでつなげた関係です。タスク A からタスク B にリンクしたとき、A がリンク元で B がリンク先となります。リンク、もっと言えばネットワークの構造は、対人関係やインターネットのウェブサイトなど多くの場面で登場しますが、タスク管理ではあまり使われません。ありえるとすれば、Scrapbox や Obsidian などリンク機能を持つノートツールを用いてタスク管理を（も）行う場合です。たとえば「引っ越しをする」ページと「将来住みたいところ」ページがあったときに、前者のページから後者のページにリンクすると、以下のような関係になります。

「引っ越しをする」 → 「将来住みたいところ」

引越し先を絞り込むために、「引っ越しをする」ページから「将来住みたいところ」ページを開くかもしれません。逆になんとなく「将来住みたいところ」ページを見ているときに、被リンク一覧に「引っ越しをする」ページがあるのを見て「そうだよなー、そろそろ引っ越しを考えるか」と意識が変わるかもしれません。ポイント関係は、タスクとタスクをゆるく繋いでおいて、未来の自分が目にして何か化学反応を起こすのを待つといった使い方で重宝します。親子関係ほどの網羅性は期待できませんが、ゆるく始められるし続きやすいのがメリットです。実は、このノートベースのタスク管理はそれなりに奥深いため、本書でも後で一章分を使って解説します（文芸的タスク管理）。

最後に **依存関係** ですが、これはワークフローの文脈で使われます。タスク A は、タスク B と C が終わったあとに開始したい、といった場合、B,C → A のような依存をつくっておくのです。専用のツールを使います。すると、いきなり A を始めようとしても「まだ B や C が終わっていません」と警告してくれたり、自動化できるのであれば B と C が終わったときに自動的に A も終わらせたりできます。たとえば今忙しくしている案件 X が終わった後に引っ越しを済ませてしまいたい場合、案件 X の最後のタスクである「佐藤さんへの引き継ぎ」が完了した後に「引っ越ししたい！」タスクが浮上するようにしたいとして、依存関係を設定すると、以下のようになります。

現在のタスク:
・佐藤さんへの引き継ぎ

将来浮上するタスクと浮上条件
・引っ越ししたい！（条件="佐藤さんへの引き継ぎ" is done）

これで実際に「佐藤さんへの引き継ぎ」タスクを完了させると、「引っ越ししたい！」タスクが浮上するので「あ、そうだった、引っ越ししたいんだった」と気付けます――と、このとおり、依存関係は親子関係以上に関係の設定が面倒ですし、設定というよりもはや設計が必要なレベルですが、その代わり、上手く作れれば最も楽ができます。特に自動化との相性が良く、この手の機能はすでに様々なタスク管理で見られる他、IFTTT や Zapier のようなワークフロー専用ツールもあります。そうでなくともチャットツールでも他ツール連携は珍しくありませんし、ローコードやノーコードといった「パズルゲーム感覚でプログラミングを行う」潮流も来ています。これらの本質は「X → Y（もし X が起きたら Y をする）」の定義であり、連携元・先のツールが自動操作に対応している前提でそれらを上手く組み合わせることで X → Y → Z → A → B のように処理を連鎖させていくわけです。といっても、あなたが抱えているタスクを自動でこなしてくれるわけではなく、あくまで「自動操作に対応しているツール」同士をその機能の範囲内で連携・連鎖させるだけです。どんなピースが存在して、どう組み合わせられるかはちゃんと調べないといけません。これは実はプログラミングの本質でもあります。タスク管理というよりもプログラミングの両分なのです。本書でもこれ以上の解説はしません。

機能

機能（Attachment）とはタスクが持つ各種機能のことです。以下に例を挙げます。

- コメント
- 添付ファイル
  - Attachment（添付）の英訳はここから来ています
- チェックリスト
- リマインダー
- リアクション（いいね等スタンプをつけることができる）

機能を充実させたタスク管理ツールもよく見かけますが、これらはあくまで機能であってタスク管理ではありません。本書でもこれ以上の解説はしません。

一つだけ言っておくと、機能の利便性や世界観に惑わされて、タスク管理をするのだとの本質を見失わないようにしてください。まずは何も使わない前提で始めて、気になるものや便利そうなものを一つずつ試していくのが良いでしょう。合わなければやめればいいですし、良さそうならそのまま定着させれば良いだけです。いきなり全部を使おうとすると持て余してしまいます。

タスク管理ツールに限らず、ツールというものは私たちの心をつかむために機能を拡充させがちです。ふらっと誘われて、あれこれ真面目に試しているつもりでも、それはタスク管理ではない、

ただの遊びにすぎません。趣味としてそうしたいのなら構いませんが、本当にタスク管理を行いたいのであれば、機能に惑わされないようにしたいところです（※1）。懐疑的な目を向けるくらいでもちょうど良いくらいです。機能よりも関係と属性に目を向けましょう。関係と属性を使って己のタスク管理をどうこなすのかを考えるべきなのです。

- ※
  - 1 機能自体が害悪だとか有用ではないなどと言っているわけではありません。むしろ機能が主役級に有用であるケースもあります。たとえば（個人からは離れますが）パートナータスク管理ではチャットという機能は主役になります。

属性

詳細は次節で扱うこととして、ここでは定義と軽い説明だけ書きます。

属性（Attribute） とはタスクの構成要素です。どの属性を採用するかは、そのタスク管理ツールやメソッドの根本でもあり、開発者の腕の見せ所でもあります。

わかりづらいので例を見ます。タスクリストを考えましょう。

まずは属性として「名前」だけ採用してみます。以下のようになります。

- 引っ越しをする
  - 引越し先を洗い出す
  - 次の引越し先を一つ決める
  - 引越し業者を洗い出す
  - 引越し業者を決める

これは一見すると普通ですが、実はとても不便です。たとえば引越し先の洗い出しが終わったのでその旨を反映したいとしても、できないのです。なぜなら属性として「名前」しか使っていないからです。「終わったかどうか」は「名前」ではありません。

ですので属性をもう一つ追加しましょう。「状態」を追加します。なお状態として未着手・未完了の [ ] と、完了の [x] を使うとします。

- [ ] 引っ越しをする
  - [x] 引越し先を洗い出す
  - [ ] 次の引越し先を一つ決める
  - [ ] 引越し業者を洗い出す
  - [ ] 引越し業者を決める

引越し先の洗い出しが終わったことがわかります。

次のタスクを行うとしましょう。ここでわからないことが一つ出ます。このタスクリストは上から順番にこなさないといけないものでしょうか？それとも順番は適当で良いのでしょうか？もちろん、そ

んなことはわかりません。属性がないからです。

というわけで属性を追加しましょう。「No」という属性を追加します。これは番号の意で、色んな意味をもたせられますが、ここでは「常に小さい順から実行しないといけない」「番号が同じなら実行順不同で良い」とします。

```
- [ ]引っ越しをする
    - [x] 1 引越し先を洗い出す
    - [ ] 2 次の引越し先を一つ決める
    - [ ] 1 引越し業者を洗い出す
    - [ ] 3 引越し業者を決める
```

1〜3 の番号を振りました。洗い出し系は独立して行えるので 1 にしています。引越し先を決めるのは、洗い出しが終わった後じゃないとできないので 2 にしています。業者決めは引越し先も決まってから考えたいので 3 にしました。これを見れば、どれから取り組めばいいかがわかりやすいですね。

ここまでで 3 つの属性を採用しました。「名前」「状態」「No」です。これくらいならある程度実用的なタスクリストとして使えるでしょう。紙やテキストファイルに自分で書いて運用することもできますし、既存のタスク管理ツールで運用したり、自分でツールから開発したりすることもできるでしょう。いずれにせよ、名前、状態、No の 3 つの属性を意識しながら運用することになります。

もちろん属性はこれだけではなく、他にも多数あります。かといって全部を採用したところで持て余すだけです。特に各属性は値を入れたり更新したりといったメンテナンスが必要であり、属性が多いほど（できることも増える代わりに）メンテナンスコストも増えます。これを **属性のジレンマ** と呼びます。要はトレードオフであり、重要なのはバランスです。自分にとって必要な属性だけをコンパクトに取り入れることです。

読み方

本章の残りでは、リファレンス的に各属性とその解説を行っていきます。

二つの立場が混ざります

解説には利用者としての立場と、開発者としての立場の両方が混ざります。明示的に区別はしませんので、読者の方で適宜読み解いてください。

利用者とは属性を使う者です。読んだり書き込んだり更新したりします。開発者とはタスク管理のツールやメソッドなどをつくる者です。どのような属性を選び、どのような見せ方をして利用

者に使わせるのかを設計・実装する必要があります。

1-段落 1-性質です

内容については、1-段落 1-性質の形で並べていきます。段落が変わる度に話題が変わる（別の性質を述べる）ものと考えてください。箇条書き的に並べていると捉えても問題ありません。

性質の他にプラクティスも解説します

属性の性質を述べる以外に、扱い方や使い分けといったプラクティスも解説します。

===

記述系属性

タスクの内容を表現します。

表現には言語を用います。つまり言語化できないとタスクも表現できません。経験的に言語化されていないタスクをツールで管理するのは難しいと感じる人も多いと思います。言語化されていないタスクとしては物タスクなどがありますが、このようなタスクを管理するツールは現状ありません。将来的には画像管理ベースで、物タスクを可視化するような形のツールも生まれるかもしれませんが、仮にそんなツールがあったとすると属性は「画像系属性」等になるでしょう。本書では扱いません。

名前/Name

タスクの名前。タスク名とも。

名前といっても表現は多用です。「引っ越し」「引っ越しをする」「将来住みたいところをリストアップする」「北海道について調べる」「都会と田舎の違いについてAさんに聞く」は全部名前です。単語に限らず文章でも良いですし、むしろ文章にして具体的にした方が扱いやすく取り組みやすいです。

名前の中に他の属性を持たせることがあります。たとえばtodo.txtは、理論的には「1行1タスクで名前を列挙したもの」であり、タスクとして `先方にお礼の電話をする` と書くこともできれば、`(A) 2017-07-14 先方にお礼の電話をする @phone +HogeProj` と書くこともできます。後者は名前に

優先度や実行日やタグが混ざっています。哲学的な話になりますが、「属性」なのか、それとも「名前（の中に書かれた属性）」なのか、の違いに正解はありません。一つの明快な定義としては、ツールの力で「属性を扱った処理（ソートやフィルタリング）」を行えるのだとしたら、それは属性です。逆に行えない場合、たとえば人間が読んで判断するだけであれば、それは属性ではなく名前です。

名前は通常、1行かつ100文字以内のテキストを想定します。複数行になったり、100文字（100という数字に大した意味はありません）を超えるような長いテキストは想定しません。長いテキストが必要な場合は詳細属性を使います。

詳細/Description

タスクの詳細。タスクの記述とも。

名前は1行のテキストでしたが、詳細は複数行のテキストです。特にプロジェクトタスク管理のような本格的なツールで使われますが、個人レベルのタスク管理では持て余しがちです。

名前以上に他の属性も持たせることもできますが、複数行のテキストに書かれた属性を人間が見て判断するのには限界があり、容易に形骸化しますので推奨はされません。たとえば締切が 2024/07/22 だからといって、詳細属性の中に「2024/07/22 までお願いします」などと書くのは良くないです。属性を使いたければ、タスクの属性として別途きちんと扱うべきです。この場合は、締切日属性に 2024/07/22 を設定するべきです。

ちなみにコメントやノートといった機能がありますが、それは 3A における機能（Attachment）であって属性（Attribute）ではありません。機能は属性ほどタスクと紐づいていないので、タスク管理の最中には軽視されがちです。軽視されずにちゃんと扱いたいのであれば、よりタスクと密に紐づいた属性である詳細属性を使いたいです。しかし、そうは言っても、詳細属性自体が複数行のテキストとなり、文章量が多くなりがちで読まれづらいです。ベストプラクティスとしては、詳細はなるべく使わないか、分量を少なくするか、テンプレート化して読み書きの負担を減らす（統一化することで減らす）ことです。

状態系属性

タスクの状態を表現します。

状態とは「未着手」「進行中」「完了」といったもののことです。

状態/Status

タスクの状態。英語では Completion や Done など「完了」寄りの意味で表現されることもあります。

状態が取りうる値は事前に定義されます。2値だとオープン/クローズ、3値だと未着手/進行中/完了、4値だと未着手/進行中/ペンディング/完了、5値だと未着手/進行中/ペンディング/完了/中止、となるでしょう。唯一の正解は無く、定義は様々ですが、理論的には「開始前」「開始後」「終了前」「終了後」の 4 ステップがあります。少し分かりづらいですが、進行中は「開始後」、ペンディングは「終了前」に属します。特に「終了前」がわかりづらいですが、これは「スムーズな進行を妨げる何らかの兆候が見られる（終了に倒れるかもしれない）」といったニュアンスですが、状態として表現するのは難しく、現在知られているのは保留中とか審議中とかレビュー中とかいったペンディングの概念です。

状態は少なくとも未完了と完了の 2 値を扱うのが望ましいです。特に完了したタスクにはもう用がありませんから、削除するなり、別の場所に退避（ログ化やアーカイブ化）するべきです。これを怠ると、タスクリストの中に未完了と完了が混ざってしまい、見る度にこれは完了したのかしてないのかといった認知コストを消耗します。部屋でたとえると、使っていないものを片付けずに放置したり、ゴミをゴミ箱に捨てずに机や地面に置いているようなものです。

状態の更新は頻繁に行う操作の一つなので、できるだけ少ない手間で行えるべきです。GUI の場合はクリックやタップ一回で行いたいです。テキストベースの場合は、多くても数回以内のキー入力で済ませたいですが、[ ]にxを追加して[x]にするなど追加する方式と、未完了に!がついているのを消せば完了になるなど削除する方式があります。更新を行わずに別の場所に移動させることもできますが、一般的には進捗（完了と未完了の比率）を見たいと思いますので、移動は少し時間を置いた方が良いです。日や週の単位がきりが良いでしょう。

識別系属性

タスク個々を識別します。

ID

タスクの ID。

ID があると同じ名前のタスクを複数扱えます。たとえばID=1の「引っ越しする」タスクと、ID=13の「引っ越しする」タスクは、名前は同じですが別物です。ID=13の方を「妹の引っ越しを手伝う」に変えたとしても、ID=1の方は変わりません。両者を別物として扱うことができます。しかし同じ名前のタスクを複数扱う用途は、タスク管理においてはあまりありません。ID はもっぱらタスクごとに専用のページをつくりたいときに使われます。逆に単に task.txt にタスクリストを書いているだけ、など個人でシンプルに運用している場合は出番がありません。

IDの表現方法は多用ですが、重複を防ぎたいのとプログラムで処理すればいいのとがあって、人間には読めない文字列になりがちです。利用者が目にする機会も通常はありませんが、URLを見れば載っていることがあります。あるいは #1 や #152 など連番が使われることもあります。

番号/Number

タスクの番号。

番号はIDの一種ですが、「連番や連アルファベットなど連続した英数字である」ことと「順序性がある」のが特徴です。本質は順序性であり、番号が若いほど「先に実行するべき」とのニュアンスがあります。このニュアンスがなく、ただのIDとして連番や連アルファベットを使っているものは番号ではなくIDです。

番号を用いたツールの好例はタスクシュートです。クラウド版でなくExcel版の話です。かつ単純化した説明になります。タスクシュートではタスクを操作してはソートして並び順を維持する、という営みになりますが、このソートは単に行ごとに辞書順で並び替えているだけです。なのでタスクを意図した並びにしたければ、先頭に001とか002と番号をつければいいのです。昇順ソートの場合、番号が若い方が前に表示されます（かつ00のようにパディングして揃えるのがより確実です）。このようなソートを常に想定しているため、タスクシュートでは各タスクに自動で連番を振ります。しかし利用者が「これを前の方に入れたい」と考えて001などに修正すると、それはそのまま反映されます（なのでソートすると001のとおり前の方に並びます）――といった世界観となっています。これはテクニカルな例ですが、もっと単純な活用例を言えば、属性の項でも述べたように、手作業で番号をつけて運用することもできます。

番号が被ったときの対処は考えておいた方が良いでしょう。タスクシュートでは番号の次の属性を見る、となってます。属性の項の例では「どちらから実施しても良い」としています。

日付系属性

タスクに紐づいた日付を定めます。

日付の表記は様々です。240722、20240722、2024/07/22、2024-07-22、2024/07/22 (Mon)、2024年7月22日(月)、などはすべて同じ日付を示しています。ツールの場合は自動で読みやすい形式になると思いますが、手作業で運用する場合はなるべくシンプルな表記が使いやすいでしょう。デジタルであれば現在日時を自動で挿入する機能を使えば少しは楽できます。

曜日は通常含めません。年月日がわかれば曜日も自ずと計算できます。しかし私たちは曜日で時系列（過去現在未来）をはかるところがあるため、常に表示されていた方が読みやすかっ

たりします。しかし、常に表示していると表示領域を圧迫します。特に1行1タスクで表示するようなツールや、スマホのような幅の狭い画面だと削らないとかえって見づらくなることが多いです。

※タイムゾーンなど高度な話題は本書では扱いません。それはプログラミングの両分となります。また本書ではJST――日本標準時のみを想定して解説します。

実行日/Starting Date

タスクを実行する日。開始予定日とも。

たとえば 2024/07/22 であれば、そのタスクは 2024/07/22 に開始することを示します。

実行日が有効なのは計画と先送りです。まず計画については、大きな何かを成すためにまずはこれ、次はこれ、と小さく分解して一つずつこなすよう設計していくと思いますが、この小さなタスクを配置していくのに実行日を使います。タスク A は 2024/07/22、タスク B は 2024/07/23、タスク C は 2024/07/24 といった形ですね。この用途だと番号属性の意味合い、つまり順序性も事実上持つことになります。この例ですと、A → B → C の順に行うことが想定されていますね。次に先送りについては、Dailerの戦略がわかりやすいですが、日ごとにタスクリストをつくる運用がまずあったとして、このとき実行日を変えることで先送りができるという意味です。たとえば今日が 2024/07/22 だと、タスクリストには 2024/07/22 に行うタスクが並んでいるわけですが、そのうちタスク D を「今日はだるいので明後日くらいでいいか」と捉えたとします。捉えただけでは忘れてしまいますので、D の実行日を 2024/07/24 に変えます。そうしておくと、明後日 2024/07/24 のタスクリストに D が出現するので忘れません（忘れていても見れば思い出せます）。

いわゆるカレンダーは、実行日属性（としばしば開始時間属性も）のついたタスクを格子状に表示しているものと言えます。改めて言うまでもないですが、実行日をつけたタスクを週や月といった単位で俯瞰することは非常に便利です。これは実行日という概念を導入しているからこそ行えることです。俯瞰の方法は一つではなく、たとえばカレンダーを使わずに予定をリストで書いて管理したり、2024/07 のタスクを順序性を無視して表示したりといった方法も考えられます。タスク管理は本質的に個人的なものであり、合理的な手段だから続くとは限りません。続かないと意味がないわけで、続くのであれば多少非合理な方法でも実は構いません。あえてカレンダーではないトリッキーな俯瞰の仕方を構築することもアリなのです。筆者も一時期はカレンダーを使わず予定をリストで管理していた時期があります。とはいえ、合理性を考えるならば、やはり俯瞰はカレンダーか少なくとも順序性を保持してソートして表示するのがベストです。特にカレンダーを超える俯瞰方法をまだ人類は編み出していません。物好きでなければ、俯瞰にはおとなしくカレンダーを使うのがベストです。もっと言えば、もしお使いのツールにカレンダー機能がない場合で、それでも俯瞰したい場合は、何とかしてカレンダーに落とし込むべきだと言えます。カレンダービューを自製するか、既存のカレンダーツールとの連携（API）を調べて連携することになるでしょう。

作成日/Creation Date

タスクを作成した日。作成したときの日付を記録する属性です。

個人タスク管理においては通常、この属性は不要です。強いて言えば、作成日を記録しておくことで、あとで取り組むときに「作成してから二週間も経ってるなー」など経過感を知ることができる程度です。一見すると、有益かもしれないと感じます。経過感がわかれば、たとえばある程度経っているとわかったら優先度を上げるなり、逆に要らないと判断して消すなりできるからです。しかし、そういった判断をいちいち行うのは認知コストを食います。筆者としては使わなくて良いと考えます。認知コスト節約の方が大事です。

作成日属性は、自動化が前提であれば重宝することもあります。まずツール側で自動で記録するのであれば利用者のコストはありませんし、現在日時との差分（何日経過したか）も自動で計算できるのなら、ワークフローに組み込めます。要は「n日以上未着手であれば催促する」あるいは「用がないとして削除する」といった処理を実現できるのです。扱うタスクの量が多くて未着手のまま放置されることが多い場面では役に立ちます。が、個人タスク管理レベルでは通常はないと思います。逆にプロジェクトタスク管理では珍しくありません。特にGitHubなどオープンソースの文脈で、活発なソフトウェアになると毎日何十というIssuesが発行されますが、メンテナーは限られておりいちいち整理などしていられません――というわけでBotという自動巡回プログラムをつくっており、「14日間誰からも返事がなかったIssueは自動的に閉じる」といったことをしています。

締切日/Due Date

タスクの締切。完了予定日とも。

締切という言葉が指すニュアンスは利用者次第です。目安程度の軽さもありえますし、基本的に厳守であり守れないなら報連相が必要ですねもありえます。

個人タスク管理では形骸化しがちな属性です。実行日属性を使って代替できるからです。たとえば2024/07/22が締切のタスクAがあるとしたら、締切日=2024/07/22と設定すると思いがちですが、実は実行日=2024/07/22でもまかなえます。要は締切当時に行えば良いわけです。ここで「いやタスクAは実行に数日以上かかる大きなタスクなんだよ、実行日にしちゃうと22日当日しかできないじゃないか」と思われるかもしれませんが、そもそもそんな大きなタスクは分解するべきです。たとえばA1, A2, A3の3つに分解できたとして、それぞれの実行日を2024/07/19、2024/07/20、2024/07/22とします。もちろんA1〜A3は1日以内に終われる程度の大きさです。このように分解をうまく行えるなら締切日は要らず実行日で代替できるのです――と書きましたが、原理的にはもちろん締切日属性を使って律儀に締切を意識することも可能です。十中八九形骸化するでしょうが……。意識ほど脆いものはありません。

締切日属性が重宝するのは、プロジェクトタスク管理など本格的な管理を行うときです。このよ

うな場面だと、通常は綿密な計画が立てられ、タスクごとに締切が引かれて全体の流れが設計されます。締切日が設定されていれば「あと 2 日しかないのに進捗が乏しいから追加でフォローしよう」といった調整も行いやすくなります。この性質上、タスクを締切日でソートして俯瞰する機能も事実上必要です。一方で、現代では計画にあまり頼らず、軽めの仮説検証を繰り返すアジャイルなスタイルも台頭しており、この場合は締切日属性は無用です。アジャイルはたいてい「今週は何するか」「次の二週間では何をするか」など週単位で捉えるので、実行日ならぬ実行週属性なるものが使えれば事足ります。2024 年現在、実行週属性を扱ったツールはないと思いますし、本書でも解説はしませんが、GitHub Issues のマイルストーン機能など近しいものはあります。

完了日/Completion Date

タスクを完了した日。終了日とも。

完了日があると振り返りに役立ちます。前提としてサイクルを回している必要があります（週単位でまわすアジャイルの例は前述しました）――つまり一サイクルごとに振り返る際に、この完了日属性を使って（振り返りで使う）インプットを計算します。たとえば締切日属性と比べると締切をどれだけ守れたかがわかりますし、作成日属性と引き算すれば経過がわかるので、たとえば平均して何日経過して終わらせたか等がわかりますし、実行日属性があるなら当日完了した率もわかります。ただし表面的に比率を求めたところで、せいぜい目安にしかなりません。調子が悪ければペースなどかんたんに乱れますし、かんたんなタスクが多かったり逆に難しいタスクがあったりした場合などでもばらつくからです。それでも、生産性を読みやすい「作業」系のタスクが多い場面では役に立ちます。いずれにせよ、少なくとも他の日付系属性を用いた計算を要するのが大変です。その機能がツールにないか、あるいは自分で手計算する手間を負えないのであれば、完了日属性は意味がないので無視して構いません。

完了日属性は状態属性の代替になりえます。つまり、完了日属性が記入されたことをもって「状態＝完了」とみなせます。再びtodo.txtの話になりますが、このメソッドでは `YYYY-MM-DD YYYY-MM-DD ティッシュ箱を買う` のようなフォーマットを使うことができます。前の日付が完了日で、後の日付が追加日です。たとえば `2024-07-22 2024-07-18 ティッシュ箱を買う` とあるなら、ティッシュ箱を買うタスクを 18 日に追加して、22 日に終わらせたことがわかります。

時刻系属性

タスクに紐づいた時刻を扱います。

単に操作日時を記録する用途ではないことに注意してください。時刻系属性の目的は日付系属性よりも――つまり日の単位よりも細かい単位で管理することです。時、分、秒という細かさで管理したいからこそ導入します。逆を言えば、日単位で済む程度の融通があるのなら時

刻系属性は要りません。

セクション/Section

タスクを実行する時間帯。

セクションは実行日ならぬ「実行時間帯」だと捉えて差し支えません。実行日属性は時間軸を日単位で区切って、日ごとにタスクを配置することで扱いやすくするものですが、この単位を少し細かくしたのがセクションです。もう少し細かくすると開始時間属性になります。つまり単位として日、時間帯、時・分・秒の 3 種類があって、セクションは 2 番目の単位を示しています。あまり一般的な単位ではありませんが、人には生活リズムがあり、生活リズムはセクションから構成されていることが多いです。詳細はコンテナの章を参照してください。

セクションが取りうる値に正解はありません。わかりやすいのは朝/昼/夕方/夜ですが、これだとたいていは粗すぎます。セクションは生活リズムと結びついており、自分にとって最適なものを定義するべきです。たとえば会社員であれば出社前/出社中/出社後午前/昼休憩/午後1/午後2/退社中/退社後/就寝前くらいの粒度になるでしょう。また平日と休日でも変わってきますし、旅行や出張など普段と異なる生活パターンになれば当然壊れます。雑に扱いたい人は就業中の時間帯を「会社」という単一のセクションに押し込めてしまうこともできます――と、運用はどうとでもできますので、セクションの値は自在にカスタマイズできるようにするべきです。

セクションを取り入れたツールの例はタスクシュートです。

開始時間/Staring Time

タスクを開始した時間。

※本項ではタイムトラッキング全般の話も扱います。

開始時間は終了時間属性とともに用いられ、そのタスクにかかった時間を計算するのに使います。計算は単純で、終了時間から開始時間を引くだけですので、自動計算のないアナログなタスク管理でも（面倒ですが）不可能ではありません。しかし、タスクを始めたり終わらせたりするたびに記入や計算を行うのは思っている以上に面倒であり、容易に形骸化に繋がりますので、できる限り自動化するべきです。現状の最適解は、ボタンを一回押す程度の手間で開始する（開始時間を記録する）ことです。あと数年すれば音声でも問題無く開始できる水準が手に入り、そのようなツールも登場すると思います。

開始時間属性によりタスクにかかった時間を計算できますが、それだけでは意味がなく、計算結果を俯瞰して振り返りに役立てる必要があります。この分析や可視化も手作業でつくるのは大変なので、通常はツールの機能に頼ります。この用途に特化したジャンルがタイムトラッキングであり、Togglなど専用ツールがあります。通常のタスク管理ツールでもサポートしていることがあります（タスクシュートなど）。また、タスク名をいちいち指定させず、単に開始と終了の記録

だけを取るタイマーのジャンルもあり、ポモドーロ・テクニックが有名です。

並列実行――同時に複数のタスクを開始する場合の取り扱いは、事前に検討しておくべきです。通常は並列実行は許可しないか、少なくともツールのコンセプトとして想定しないことが多いです。というのも、並列実行を認めてしまうと、後の計算や分析に支障が生じるからです。たとえば朝9時から12時の間は3時間ですが、並列実行していると3.3時間とか、雑な記録だと4.5時間になったりしてめちゃくちゃです。あえて言うなら自分のマルチタスク具合を可視化する用途としては使えますが、そもそも並列実行のすべてを律儀に記録する（開始と終了の二回、かつ開始時はおそらくタスク名の入力も必要）ことができずに形骸化します。並列実行するシチュエーションそのものが忙しいがゆえに、記録する暇や余裕が普段より無いという点も拍車をかけます。

扱う単位としては秒レベルが望ましいです。というのも、開始と終了が一分以内に終わるケースがあり、秒レベルで扱えていないとここの記録が粗くなる（かかった時間が 0 分になるか 1 分になる）からです。特にタイムトラッキングのように細かく行動を記録するシチュエーションだと、一分未満に終わったタイニーなタスクが十数個以上存在することがあります。仮に 15 個だとすると、最大 15 分もの誤差になります。内部的に秒で管理していると、秒レベルの計算で正確にわかります。しかし、表示時に秒まで出すのは細かすぎてノイジーですので、表示は分が良いでしょう。一方で、神経質な利用者だと秒レベルで見たいことがあります。理想を言うと秒と分、どちらも切り替えられるようにしておくことです。

終了時間/End Time

タスクを終了した時間。

連続的にタスクを行う前提であれば、次のタスクの開始をもって現在のタスクを終了させることで終了操作を省力化できます。たとえばタスク A を 10:37 に開始し、11:03 にタスク B を開始した場合、A は終了したとみなして終了時間を 11:03 にしてしまえば A の終了操作は不要です。

終了時に追加のアクションを行うどうかは設計しておくべきでしょう。たとえばタスクシュートは「行う」コンセプトであり、その場で感想や振り返りを記入できるようになっています。一方で、タイムトラッキングツールでは「行わない」がメインであり、あくまでもタスクの実行と開始終了の記録に専念します。一つの目安として、定量的な記録を用いて分析したい場合は「行わない」が良く、定性的な感想も入れたい場合に新鮮な感想をすぐ書きたいなら「行う」が良いです。

その他省力化に繋がる操作を検討できる余地があります。たとえば「このタスクをもう一度開始する」ボタンや「今日終了したタスクを選択してもう一度開始する機能」があれば、同じタスクを即座に開始できて便利でしょう。共通するのは終了という操作が契機になっていることです。タスクの終了とは終了時間を記録することですので、終了時間の記録は次のタスクを行う直前のタイミングでもあると言えます。次のタスクの選択・開始をシームレスに上手くサポートできたら、グンと快適になるのです。開発者の腕の見せ所でしょう。

キャンセルタスクの扱い方も検討しておきたいです。**キャンセルタスク**とは開始したタスクを取り下げること、特にその取り下げることになるタスクを指したものです。開始時間を消してその名のとおりキャンセル（実行しなかったことにする）するか、あるいは終了時間を記入していったん終わらせるか、のどちらかに対処が必要です。どちらを選ぶかは利用者の好みになりますが、キャンセルタスク自体も情報の一つであり、分析時に把握できると便利ですので、可能ならツール側で支援したいです。たとえば上述した追加のアクションとして「キャンセルマークをつける」機能があるとワンタッチでキャンセルできて便利ですし、あとで振り返るときも区別できます。そうでなくとも終了時にメモを記入させる設計であるなら、メモとしてキャンセルの旨を書くことができます。

見積時間/Estimation Time

タスクにかかる時間の見込み。Expect Timeとも。

見積時間属性の目的は、タスクの見積もりを行うことです。開始時刻属性と終了時刻属性により「かかった時間」がわかります（実績時間）が、それだけでなく、開始前に見積時間を定義しておくことで予定化・計画化に役立ちます。たとえばタスクA、B、Cをこの順で、30分、45分、45分で終わるものと見積ったとします。つまりA、B、Cの3タスクを2時間で終わらせたいという計画です。この後、タスクをこなして開始と終了を記録していくと、実際かかった時間がわかります。仮に40分、50分としましょう。この時点で1.5時間かかっており、タスクCは予定では45分かかるが残りは30分しかない、まずいかもしれない、とわかります。見積もりは奥深いジャンルであり、本書でもこれ以上は扱いませんが、タスク単位で見積もりを行うためにはこの見積時間属性が必要です。

デイリータスクリスト上のタスクすべてに見積時間を設定すると、「今日のすべてのタスクがいつ終わるか」がわかります。終わりが見えるため精神衛生を保つのに重宝します。逆を言うと、扱うタスクのすべてに、あるいは少なくとも一連の連続したタスクのすべてに対して見積時間を設定しないと意味がありません。たとえば午前中にタスクがA〜Hの8個あったとして、Cの部分だけ見積時間を記入して見積もりができたとしてもさほど意味はないでしょう。少し極端に言うと、All or Nothingです。Allができないのなら見積時間はスルーした方がマシです。ツールの例としてはタスクシュートがあり、現在のクラウド版はわかりませんが、Excel版の時代では一コンセプトとして強調されていたと記憶しています。

実績時間/Actual Time

タスクの実行にかかった時間。

開始時間属性と終了時間属性から求めることができます。もっと言えば、開始時間の記録と終了時間の記録から求めることになりますので、記録の仕方に問題があれば現実と乖離します。たとえばタスクAを13:10に終了したのに終了操作（終了時間の記録）を行わず、あとあと14:00に気付いて終了したとすると、記録上は50分多めにかかったことになってしまいます。

もちろん「実際は 13:10 くらいに終わってたから……」と修正するのも面倒なことです。

定期系属性

タスクを繰り返し実行するための設定値を扱います。

タスク管理では、繰り返し行うタスクをルーチンタスクと呼びます。ルーチンタスクは出現の仕方が指定されたタスクです。たとえば 2 日に 1 回行うタスクであれば 2 日ごとに出現しますし、毎週水曜日であれば毎週水曜日にのみ出現しなくてはなりません――と書いてもわかりづらいですが、前提としてデイリータスクの概念を使っている必要があります。実行日属性の項では「日ごとにタスクリストをつくる運用」と表現しました。その上で、じゃあどの日に出現させるのかを制御するのがルーチンタスクです。つまり定期系属性とはルーチンタスクを管理するための属性です。

ルーチンタスクを実際にどのように出現させるか――もっと言えばどのように設置するか、には複数の方法があります。最もわかりやすいのはカレンダー等に見られる Set all firstly 方式であり、一度定期系属性を設定しただけで、該当するすべての日にタスク（カレンダーの場合は予定）が刻まれます。ですので「この日だけはスキップしたいから消す」といった例外対応もやりやすいです。次にタスクシュートなどが採用している End and Set 方式は、終了した（終了時間を記入した）ときに、次のタスクを生成します。2 日に 1 回行う設定なら 2 日後にのみ作成します。カレンダーのように高度に抽象化されたツールですと Set all firstly 方式にて無限に広がる未来のすべてに設定できるのですが、ツール上そうもいかない場合はこの End and Set 方式で直近にのみ設定することになります。他にもHabiticaや習慣トラッカーなどが採用する Counter 方式があり、これはとりあえず毎日出現させるというものです。やるかやらないかは利用者が決めて、やったのならその旨を記録します。ルーチンタスク管理では力不足ですし、いちいちやる・やらないを判断するのは認知コストを酷使するため疲れますが、最もわかりやすい方法ではあります。

以下に 3 つの方式の特徴をまとめておきます。

- ルーチンタスクの設置方式
  - Set all firstly
    - 定期系属性の設定値に基づいて、該当する未来日すべてにタスクを設置する
    - カレンダーなど抽象化しやすいツールでは使える
  - End and Set
    - タスク終了時に、定期系属性の設定値に基づいて、次のタスクのみ設置する
    - Set all firstly 方式が使えない場合はこの方式でしのぐ
  - Counter
    - とりあえず毎日表示して、実行するかどうかは利用者に任せる
    - 定期系属性を使わないシンプルな方式だが、ルーチンタスク管理には力不足

タスクAをある日Dに出現させたい場合は、Aの実行日属性をDにします。たとえば今日が2024/07/24水曜日で、「可燃ごみを捨てる(@毎週月金)」ルーチンタスクがあった場合、このタスクの次の実行日は2024/07/26金曜日になります――と、ルーチンタスク管理を行うなら(開発者だけでなく)利用者もこのような考え方を心がける必要があります。この考え方ができて、はじめてルーチンタスクを自在に操れるようになります。たとえば今週の金曜日のごみ捨てをスキップしたい場合、「実行日2024/07/26の可燃ごみ捨てタスクを削除すればいい」と捉えなければなりません。そうしないと実際にツール上にタスクの情報を反映できないからです。

繰り返し頻度/Repeat Frequency

タスクを出現させる頻度。

繰り返し頻度は「n日ごとにx回行う」との形で記述できます。たとえば「1日1回(毎日)」「2日に1回」「7日に1回(週に1回)」などです。xは通常x=1のみを想定するのが良いです、というのも管理上扱いやすいからです。「1日に2回」など日よりも密な頻度を表現したい場合は、単に「1日1回」のタスクを2個つくります。2個つくったタスクをいつやるかは時刻系属性も使って制御してください。繰り返し頻度含む定期系属性はあくまで日以上の単位を扱うべきです。この棲み分け、いわば役割分担を徹底することで、タスク管理が煩雑にならずに済みます。

繰り返し頻度には表現力に限界があります。たとえば毎週末(毎週土曜日と日曜日)、毎月30日、月末、第2水曜日といった頻度は、繰り返し頻度属性だけでは表現できません。あるいはEnd and Set方式で設置したタスクの実行日を都度修正する手間が発生します。この表現力を使いたいなら、後述の繰り返し条件属性を使ってください。

繰り返し条件/Repeat Condition

タスクを出現させる条件。

繰り返し条件属性は、繰り返し頻度属性では表現しきれない細かい頻度を実現するために使います。

利用者に対してはわかりやすい見せ方をする必要があります。開発者の腕の見せ所ですが、一から考える必要はなく、既存のカレンダーツールの「定期タスク」の設定画面を見れば事例はすぐに集められます。たとえば「毎週水曜日と金曜日」のような頻度を指定させるには、月から日まで7つのチェックボックスを並べて、出現させたい曜日にチェックをつけさせる――といったインタフェースが使われています。一方で、設定画面は全体的に煩雑になりがちで、(一度設定したら終わりな)予定の管理はまだしも、(一日何十回何百回と操作する)ルーチンタスク管理として使うには厳しいものがあります。

繰り返し条件は、開発者としては負担が大きい属性です。複雑な日付計算を負うことになりますが、日付計算ライブラリだけでは足りない可能性があります。特に「第三水曜日」や「会社

休日」など文化固有、組織固有の表現や制約も混ざってくると自製する手間からは逃げられなくなります。仮につくりこみが甘くて、本来出現するべき日に出現しなかったなんてことがあっては大問題ですから、品質のつくりこみやテストも大変です。そもそもルーチンタスク管理などという神経質な管理は（鉄道の正確性にも見られるように）日本的な文化であり、世界的に見てもマイナーだと感じます。どこまでつくりこむかにもよりますが、繰り返し条件属性を用いた本格的なルーチンタスク管理ツールをつくりたいのであれば、（2024/07 現在では）自らつくって品質を担保する手間を負うことになると考えてください。

繰り返し条件として有用なものの一つに Manual Listup があります。これは出現させたい日を利用者に選択させるものです。たとえばカレンダーを表示して、出現させたい日を選択させます。既存の繰り返し条件や繰り返し頻度を使って事前に選択させておき、気に入らないところだけ追加や削除させると（一から全部選択する手間がなくて）なお良いでしょう。この方式の主な問題点は利用者に選択させる頻度とその周知ですが、頻度は週一か月一のいずれかにして、かつ忘れないようツール側でリマインドさせると良いでしょう。

分類系属性

タスクの性質を表現します。

分類系属性の目的はフィルタリングを可能にすることです。タスク A に分類項目 X を付与すると、「X が付与されている」という条件で A を（もちろん他にも X が付与されたタスクも）洗い出すことができます。タスク管理ではしばしばタスクの数が何十、何百、下手すると何千にもなるので、このようにフィルタリングできることは重要です。

3A における関係（Arrow）は扱っていないことに注意してください。カテゴリーなど一見すると包含関係を扱ってそうな名前に聞こえますが、関係における親子関係では **タスクがタスクを** 包含（同族包含）するのに対して、属性における分類系属性では **分類項目がタスクを** 包含（メタ包含）します。同族包含ではフィルタリングよりも関係の可視化を重視します。メタ包含では前述のとおり、フィルタリングを重視します。

他の属性を分類系属性として使うこともできます。たとえば日付系属性は、日付という分類項目が付与された分類項目と呼ぶこともできます。しかしこれはあくまで（本来別の用途として存在していた）時間軸的情報を、ついでにはフィルタリング用途としても使うというだけの話です。フィルタリングの便宜は限定的です。一方、分類系属性は、時間のみならず他の体系や、それこそ言語そのものも使いますので、分類項目が非常に柔軟です。あまりに柔軟すぎて、適切に採用・運用できなければすぐに形骸化してしまいますが、それは利用者の問題でありタスク管理ツールの問題ではありません。ここに踏み込みすぎると、タスク管理ツールではなく情報整理ツールとなってしまいますので開発者は注意しましょう。

分類系属性はアナログなツールでも使うことができます。バレットジャーナルでは項目の先頭にアイコンをつけることで意味を付与します（ラピッドロギング）し、あな吉手帳術では「夫がいない

ときにできること」エリアをつくって、そこにタスク（を書いた付箋）を貼り付けておくといったことができます――と、マークをつけるなり領域で区切って集めるなりすれば分類系属性の付与と同等のことを実現できます。

その他詳細はコンテナの章も参照してください。

カテゴリー/Category

タスクに 1 つだけ付与できる分類項目。

カテゴリーとはタグの一種であり、タスクに 1 つしか付与できないタグだと考えて差し支えありません。したがってカテゴリー C を削除しても、C の付与されたタスクは削除されません。これが関係（例: 大タスクに包含された中タスク）であれば削除されますが、分類系属性としてのカテゴリーはただのタグであり付与なので、タグという付与が外れるだけです。逆を言えば、削除が連動されるような「入れ物」を実現したいのなら、分類系属性としてのタグではなく、関係としての親子関係を使うべきです。

カテゴリーは自由に CRUD（作成・参照・更新・削除）できるようにするべきです。何のカテゴリーをどれだけ使うかは利用者次第であり、利用者が自ら CRUD していくことになるため、かんたんに CRUD できる方が便利だからです。一方で、カテゴリーはタグとは違って乱用するものではない（後述）ため、あえて CRUD のハードルを上げるのもアリです。

カテゴリーの CRUD は乱用するべきではありません。巷に存在する「分類」と呼ばれる体系がそうであるように、カテゴリーとは事前に注意深く設計した上で採用し、厳格に分類していくものです。タスクに 1 つしか付与できないという制約もそのためです。タスクをどのカテゴリーに入れるかを厳選するべきなのです。これは厳選が正しいと言っているのではなく、単にスタイルの問題です。カテゴリーという分類項目は厳選に基づいた分類を行う体系だ、というそれだけの話です。この厳しさを許容できない場合は、タグを使ってください。

タグ/Tag

タスクに単一また複数付与できる分類項目。

タスクには複数のタグをつけることができます。近年では SNS において投稿物にハッシュタグを付けますが、概念的にはこれと同じです。対象の性質を示す札（Tag）を貼り付けるイメージです。

タグはたくさんつけた方が良いと思われがちですが、タスク管理においてはそうではありません。たくさんつけた方が良いのは、SNS などアクセス数を稼ぐ文脈に限った話です。タスク管理は違います。目的は前述のとおり、フィルタリングにありますので、実際にフィルタリングできなければ意味がありません。「～～に当てはまるタスクを知りたい」というときに、「～～」に相当するタグ（の名前）をすぐに思いつけるか、または発見できるかどうかが重要となります。ですので、たくさんつ

けていると、タグ個々の印象が薄くなって思い出しづらいですし、一覧から探すにも大量になって混乱します。カテゴリーほど厳選する必要はありませんが、無闇につければ良いというものではありません。一つの意識としては、タグを新しく作成する、タグ名を変更する、タグを削除するときに毎回軽微なお金がかかるとしたら、を考えてみることです。10 円でも 20 円でも 50 円でも構いませんが、蓄積すれば下手なサブスクに至るような金額です。それくらいお金がかかるとしたら軽率なことはできず、多少は慎重になると思います。と、これくらいの意識が望ましいのです。

理論的には属性はタグで表現できます。なのでツールがサポートしていない属性を使いたい場合でも、タグがサポートされているなら擬似的に再現できます。たとえば実行日属性ならぬ実行月属性なるものを実現したいとするなら、単に 2024/07 や 2024/08 といったタグをつくって付与するだけです。2024/07 のついたタスクは「実行月=2024/07」と解釈でき、2024/07 タグでフィルタリングすることで表示できます。もちろん、「2024/07 以前のすべて」のような高級な処理は行えません（あるいは 2024/07、2024/06、など該当するタグすべてを指定してフィルタリングする必要がある）。とはいえ、このような手動の運用は手間であり、容易に形骸化します。サポートしていない属性をタグで再現するよりも、サポートしているツールを探して使った方が確実です。元のツールに離れられないくらいの愛着がある場合は、元のツールと、そのサポートしたツールの 2 ツールを使い分けることになるでしょう。一見するとこちらの方が手間ですが、手動運用の面倒くささによる形骸化は非常に手強いです。この手ごわさを減らせるツールがあるなら（新しく導入してでも）使った方が良いのです。

ラベル/Label

タスクに単一また複数付与できる分類項目。

ラベルは色付きタグです。ラベルにはラベル名とラベルカラーがあり、視覚的に識別しやすくなっています。

ツールの例としてはGitHub Issuesなどの BTS です。重大なバグには赤色のラベルを、重複したイシューや対応不要に倒したイシューには灰色のラベルを、といったように色のニュアンスを尊重しながら付与すると、視覚的な識別のしやすさがグンとあがります。

ラベルはタスクの分類を素早く認識するのに便利です。医療現場におけるトリアージも黒・赤・黄・緑の四色に分けており、たとえば黒色を見るだけで一瞬で「既死」「蘇生の可能性がない」と判断できます。文字で「死亡」と書かれたものを読むよりも素早く判断できることは容易に想像できるでしょう。人は一般的に言語よりも非言語（色はこちらです）情報の方が素早く認識できます。とはいえ、個人タスク管理において、このレベルの素早さが求められることは稀だと考えます。あるとしても、プロジェクトタスク管理として非常に慌ただしく、かつ大量のタスクを扱っていて、秒単位で素早く動くことが求められるようなシチュエーションでしょう。トリアージはまさにその例の一つです。

ラベルは色付きタグとして使うべきです。つまり色による視覚的な識別のしやすさ、特に早さを

重視するために使います。ツールによってはフィルタリングを備えていることもありますが、タグ属性のように扱うと（フィルタリングと色の 2 つの観点を扱うことになるので）かえってノイジーです。フィルタリングを行う場合は色無しのタグ属性を、そうではなく一覧から視覚的に探したい場合はラベル属性を、という形で使い分けることを推奨します。もしタグ属性がなくラベル属性のみがある場合は、ラベルの一部をタグとして使うと良いでしょう。たとえば白背景黒文字のラベルをタグとして使う、などです。

スター/Star

タスクに 1 つだけ付与可能な、カスタマイズ製のない分類項目。ブックマークとも。

カテゴリーは利用者により CRUD できましたが、スターはできません。スター属性は単に印をつけておくものであり、つけたかつけてないかの二値となります。こうしておくと、あとで「スターがついているタスク」をフィルタリングできます。能力的にはカテゴリーの下位互換ですが、CRUD さえさせないシンプルさが特徴です（スターは常に一つであり、たとえばスター1とスター2の二つをつくって使い分けよう、といったことはできません）。

ツールの例としては、タスク管理よりもシンプルなツールによく見られます。たとえば Slack などチャットツールにはメッセージをブックマークしたり保存したりする機能がありますが、これはスターをつけているに等しいです。あとでブックマーク一覧や保存メッセージ一覧を見ることで確認できます。スターをつけて、一覧画面で確認する――非常にシンプルです。タグはもちろん、カテゴリーよりも表現力は乏しいですが、これだけでもそれなりには便利です。

スターの典型的な利用例はインボックスです。「あとで読む」や「未読」の概念はよく知られています。これらは原理的には「あとで読む」マークや「未読」マークをつけておき、マークのついたものをフィルタリングしているだけ、とスターそのものです。

スターはシンプルですが、形骸化しやすい属性でもあります。原因の大半は外すコストの高さです。「あとで読む」「あとでやる」といった用途でスターを用いている場合、あとで読んだりやったりした後はスターを外す必要がありますが、このコストが馬鹿になりません。たとえスターを外すというワンクリックの操作であっても面倒くさいのです。そのためメールやチャットのように自動で外す（未読は一度でも読むと自動で解除されます）ようにして、未読にしたいなら利用者の側で明示的に行わせるのが望ましいです。利用者の心理として「勝手に外されたくない」「自分の意志で操作して外したい」と思いがちですが、実際は外すコストが高くて外さなくて形骸化してしまうため、この心理には従わないでください。どうしても自分でやりたい人向けには、「自動で外すのやめる」のような設定を入れておくと良いでしょう――と小難しく書きましたが、これらはすべてメーラーが実装しています。開発者としてスター属性を扱う場合、メーラーの設定やインタフェースを参考にしてください。よりモダンなもの（特にすっきりした見せ方など）がほしければチャットツールや SNS も参考になりますが、筆者としてはメーラーの方が勉強になると感じます。メールはメッセージというよりもタスクに近い概念であるため、メーラーには（スターに限らず）タスク管理に必要なエッセンスが色々と盛り込まれています。ただ詰め込み方や見せ方が上手くないだ

けです。

カテゴリーやタグでスター属性を代替することもできますが、あまり望ましくはありません。既に述べたとおり、スターはシンプルさと素早さ、特に外す部分を自動化するレベルで最適化したいため、カテゴリーやタグでは力不足になる可能性が高いです。タグの項でも述べましたが、お使いのツールがスター属性をサポートしていなくて、かつそれでもスターを使いたい場合、代替よりも「スターをサポートした別のツール」をさらに使った方が定着しやすいです。心理的には代替的であっても今のツールで使いたいと考えがちですが、力不足な代替では形骸化しやすく定着しません。直感には反しますが、しっかりとサポートされたツールを（新たに増やしてでも）使った方が、最終的には定着しやすいのです。これを **ツールのパラドックス** と呼びます。キッチン用品などはわかりやすいと思います。包丁でも箸でも皿でも何でもいいですが、一つの種類を使うよりも、複数の種類を使い分けた方が上手くいきます。

優先度/Priority

タスクに 1 つだけ付与できる分類項目。

優先度属性は優先順位を制御するために使います。たとえば高、中、低の三段階の優先度を用意し、各タスクに付与することでタスクごとの優先度がわかりますし、「高優先度のみ表示する」といったフィルタリングも可能になります。

優先度の値は多岐にわたります。高/中/低、極高/高/中/低、High/Middle/Low、5/4/3/2/1。また問題管理や障害管理の文脈では問題の深刻度を指す指標として使うことがあり、この場合は Emergency/Alert/Critical/Error/Warning/Notice のような定義になることもあります。いずれにせよ、どのような値を使うかは自由であって正解はありません。

個人タスク管理では通常、優先度属性は使用しません。優先度属性はプロジェクトタスク管理など本格的な管理で用いるものです。というのも、これは定義をしっかりと定めて、使用も周知させることでようやく機能するものだからです。たとえば「高」はこういう条件が当てはまったら付与します、高をつけたタスクには必ず担当者をひとり以上つけて、チャットでも全体メンションを打って周知させます――のように運用のレベルで定めます。個人の範疇でここまで行うモチベーションは通常無いでしょう。逆に、個人であってもこのレベルでしっかりと運用できるのならば、使っても構いません。よくあるのが主観で判断して優先度をつけることですが、判断にブレが生じますし、ブレた分類項目はすぐに形骸化します。感覚的に区別したいのならスター属性（高/それ以外、の二値から成る優先度とも言えます）で十分です。

トリガーコンテキスト/Trigger Context

タスクに単一また複数付与できる分類項目。

トリガーコンテキストは、そのタスクを実行できる状況を示します。詳細はコンテキストの章を参

照してください。

トリガーコンテキスト属性は通常ツールではサポートされません。代わりにタグ属性やラベル属性として――つまりタグ名やラベル名としてトリガーコンテコストを書いて使います。たとえば電車やバスなど移動中に行いたいタスクを管理したいとして、「移動中」という名前のトリガーコンテキストをつけたいとするなら、「移動中」タグや「移動中」ラベルをつくります。ただしツールによってはサポートしていることがあります。GTD ベースの OmniFocus では「コンテキスト」と呼ばれていますし、タスクシュートでは「モード」という言葉が使われていますが本質的には同じです。

トリガーコンテキストはタスクに複数付与できますが、可能なら一つのみ付与することを推奨します。なぜなら複数付与は疲れるからです。たとえば「移動中」と「パートナーがいないとき」の二つのトリガーコンテキストが、またタスクとして「タスク管理を噛み砕くを読む」があるとします。このタスクにはどのトリガーコンテキストをつければいいでしょうか。一見すると「移動中」も「パートナーがいないとき」も両方つけておけば良いだろう、どちらのシチュエーションでも探し出せて便利なはずだ、と思えますが、実はアンチパターンです。トリガーコンテキストが複数ついているということは、そのタスクは複数の状況で実行できることを示しているわけですが、これは言い換えると、その複数の状況（この例では移動中とパートナー不在時の両方）が来る度に実行するかどうかの判断が必要だと言えます。判断の数――つまりは認知コストを食う機会が増えるため、疲れるのです。タスクが少ない、特に PC やスマホなどでスクロールせずとも見切れる程度に少ないなら問題になりませんが、タスクの数が増えれば増えるほどこの形骸化は加速します。認知コストの負荷を舐めてはいけません。それよりも「移動中に読むか」と割り切って、「移動中」のみつける方が良いのです。あるいは「パートナーがいない静かな自宅でゆっくり読むか」と考えて「パートナーがいないとき」をつけても構いません。別の言い方をすると、あるトリガーコンテキストでフィルタリングをしたときの表示数（表示されるタスクの数）はなるべく小さくしてください。複数付与を許すと、すぐに長いリストになってしまいます。長い TODO リストが容易に破綻するように、長いフィルタリング結果も形骸化します。

主観系属性

タスクに関する所感を表現します。

主観系属性は主観的に決めるものであるため、正解や厳密性はありません。むしろこれらに頼らず、率直な所感を素直に反映するべきです。主観である分、厳密な記録としては役に立ちませんが、それでも自分が感じたことの記録ではあるため何もしない場合よりは役に立つ可能性があります。

主観系属性は定量的に決めるべきです。何らかの文章や単語を書くような営みは、すべて記述系属性か機能（Attachment）として扱ってください（※1）。ただし、いくつかの単語や絵文字から選択させる場合は定量的と言えます（本質的には 1〜n 番目の選択肢から 1 つを選んでいるに等しい）。なぜ定量的にするべきかというと、タスク管理――特にあとで実行するタスクを

選んだり、実行したタスクを振り返ったりするときの認知コストを減らしたいからです。文章を読んで理解することにはかなりのコストを費やします。文章と向き合いすぎて認知コストを酷使するせいで続かない、はよくある失敗例の一つです。そうではなく、数字や選択肢の中から選ぶだけ、という形にしておくと、書くときもかんたんですし、あとで読むときも認知コストは少なくて済みます。自分でつくった体系に慣れる必要こそありますが、定性的な文章を扱って認知コストを酷使するよりははるかにマシです。

- ※
  - 1 記述系属性として扱う場合、（本章では取り上げていませんが）備考属性や所感属性なるものをつくって、そこに書くと良いかもしれません。結局はあとで読み返せれば良いだけなので、詳細属性を使っても良いでしょう。

重要度/Importance

タスクの重要性。

重要度属性の目的は、利用者にとって重要なことを可視化して行動を促すことです。重要度属性のついたタスク、特に重要度が高いとなっているタスクは、利用者にとって重要ですので優先的に向き合うべきです。人は重要なことを見失いがちですし、自覚がないこともありますが、重要度属性を運用することでそこに明示的に介入できます。

定義の仕方については、優先度属性と同じで構いません。というより、重要度属性は本質的には優先度属性と同じものです。優先度の定義方法として「主観的な重要度」を使うと、この重要度属性になります。

重要の基準は主観で決めますし、文脈から決めても構いません。つまり同じ人であっても、あるタスクの重要度が 3 日前は「低」だったのに今日は「高」と感じる、といったことが起こります。統一感が無くて、つい基準をしっかり定めたくなりますが、堪えてください。主観系属性はあくまで主観に頼った属性であり、その場その時の主観を大事にしてみるというアプローチです。もしこれが嫌で、もっと統一的な基準に基づいて運用したいのであれば、優先度属性を使ってください。

重要の定義に正解はありません。タスク管理における重要、と聞くと時間管理マトリクスがよく挙がりますが、ここでも重要の定義はありません。何が重要かは人それぞれですから当然のことです。

重要の定義に重要以外の意味を含めないでください。たとえば時間管理マトリクスでは重要と緊急を区別しており、単に緊急性の高いタスクが重要であるとは限りません。緊急性、収益性、稀少性といった因子はよく重要の定義に混ざりがちですが、これらは状況や性質を表しているにすぎず、利用者自身が考える重要と必ずしも被っているわけではないのです。

重要度の高いタスクをどう処理するかは利用者が柔軟に決めます。「すぐに着手するべき」と考えがちですが、これは前述のとおり緊急性の観点にとらわれすぎています。重要だがすぐには実

行できないとか、重要だからこそまずは考えを深めておくべきとか、重要だが現状は成す術がないのでとりあえず静観していこうなど、どう動けるかは様々です。もちろん、タスク管理のセオリーとしては、ただ単に決定するだけでは意味がないため、具体的に予定なりタスクなりを仕込んだり、モットーとして掲げたりするなり何らかの行動に移した方が良いでしょう。

難易度/Difficulty

タスクの難易度。

難易度属性を使うと、タスクの実行タイミングを主観的に調整することができます。難易度の高いタスクは取り組み方を工夫しなければなりませんし、逆に低いタスクは後回しにできるでしょう。もちろん人によって「かんたんなものから先に全部済ませる」派だからまずは難易度の低いタスクから潰す、もありえます。このような判断を行えるようになります。特に難易度の定義はしばしば「自分が持つコンテキスト」と重なるため、難易度を上手く捉えることでミキサーのように「一気に処理する」ことも可能になります。

定義の仕方については、優先度属性を参照してください。難易度属性も本質的には優先度属性と同じです。優先度の定義方法として「主観的な難易度」を使うと、この難易度属性になります。

難易度の定義に正解はありません。いくつか取り上げると、不確定性（決まってないことが多い）、技術的難易度（技術的にむずかしい）、時間的難易度（時間が足りない）、好き嫌い（※1）、ロードコンテキストの量（先に進めるために思い出すことが多くて再開が苦しい）、拘束の程度（場所や時間を指定される＝拘束が厳しい）などです。

難易度属性は事後評価の用途で使うべきではありません。たとえばタスクを実行したあとの記録として難易度を書いておく、等がありがちです。難易度属性はあくまで事前（実施前）の難易度を決めるものです（※2）。

- ※
  - 1 好き嫌いについては、嫌いだから難易度が上がる（好きだから下がる）とは限りません。好きだからこそこだわりすぎて衝突してしまうため難易度が上がる、ということもありますし、嫌いゆえにどうでも良くて最低限の対応で済ませるから難易度は案外高くない、といったこともあります。また最低限で済むという意味ではかんたんだが、精神的には疲弊するので自分にとっては難易度は高い、といった捉え方もあります。主観系属性ですから捉え方は自由です。
  - 2 本書では解説しませんが、主観的な難易度を用いた見積もりも可能です。本項で取り上げた難易度属性は「事前評価」的な難易度であり、注釈元の「事後評価」的な難易度と組み合わせると、事前はどう感じて事後はどうだったか、という難易度の印象の見積もりが行えます。見積もりというと通常、時刻系属性で示したような時間量――もっと言えば「蓄積される数量」が使われがちですが、難易度のような離散値でもできないことはないのです。ただし、これをサポートしたツールやメソッドは筆者の知る限

り、無いと思います。効果も想像の域を出ませんが、おそらく難易度との向き合い方を最適化することで QoL が上がるのだと思います。仮に筆者が名前をつけるとするなら、Difficulty Driven Work-style（難易度駆動）とでも名付けるでしょう。

協調系属性

タスクに紐づいた人を表現します。

協調系属性は個人タスク管理では扱いませんが、プロジェクトタスク管理では（特に担当者属性は）欠かせないものです。

※本書は個人タスク管理を扱った本であるため、協調系属性の解説は軽く行うに留めます。

担当者/Assignee

タスクの実行を担う者。

担当者属性の目的は二つあります。一つ目は、タスクを実行する（状態属性を終了や完了に持っていく）責任者を指定することです。これにより担当者はタスクの遂行はもちろん、状態の変更といった操作を行う責任を負います。もう一つは利用者自身が抱えるタスクの洗い出しです。通常、プロジェクトでは多数のタスクが存在し、自分が担当するタスクを探し出すのは容易ではありませんが、担当者属性の設定が適切なら「担当者＝自分」でフィルタリングするだけで済みます。

発行者/Owner

タスクを作成した者。

発行者属性は自動で記録するべきです。

発行者属性は通常使われません。タスク管理において重要なのはタスクの内容、そして実際に責任を負う担当者であり、誰が発行したか（作成したか）に価値はないからです。仮に発行したタスクに問題がある場合でも、完了操作などを行ってクローズすれば済みます。もし発行者属性を使用している場合は、良くない兆候だと考えられます――過度に複雑な管理になっているか、政治が発生している可能性が高いです。ただし、これは複雑な管理や政治が不要と言っているわけではありません。

貢献者/Contributor

タスクに携わった者。

貢献者属性を使うと、そのタスクに誰が絡んだかを一目で知ることができます。BTS などタスク上（タスクの個別詳細ページのコメント欄など）で議論が活発になる世界では、どの議論をどれだけ読むかという取捨選択が重要となりますが、このときに使う端的なテクニックが「誰がどの議論に絡んでいるかを確認する」ことです。貢献者属性はその役に立ちます。

貢献者属性は自動で記録するべきです。

貢献者属性は、メンバーのアバターアイコンを並べる等の形で視覚的にわかりやすく強調することを推奨します。可能ならばタスクの一覧画面でも表示して、一覧的に視認できるようにするとさらに便利です（いちいちタスクの個別ページを確認せずに済みます）。一方で、これは個別ページを読まない怠慢も後押ししてしまい、特定の、たとえば気に入った人のタスクだけ読むとか嫌いな人のタスクは読まないとかいった事態が生じます。政治色を強めるのです。そういう意味で、貢献者属性には読み手のコストを削減する効果がありつつも、やりすぎると政治力を強める危険性があります（ **貢献者属性のジレンマ** ）。

購読者/Subscriber

タスクを監視している者。Watcher とも。

購読者属性は、購読――更新通知を実現するために使います。タスク A を購読したい人は A を購読します。すると A の購読者にその人が追加され、A の状況が更新されたときに A に通知が行きます。あまり動きがないタスクの動きを知りたいときや、逆に重要なタスクで動きがあったらすぐに知りたい場合などに使えます。当然ながら SNS やチャットと同様、購読しすぎると通知量が増えて首が絞まります。決して万能ではなく、ここぞで使う程度がバランスが良いでしょう。

購読者属性を重視したタスク管理は現在開拓されていません（※1）。

- ※
  - 1 開拓の余地があると思います。購読の概念は RSS リーダーや SNS などですでに有用性が示されており、その本質は「自分が購読したものを」「更新順で」知らせることですが、意外なことにタスク管理では積極的に使われていません。特に BTS など議論を行うような場面では重宝するはずです。これに近い使い方ができるのが Scrapbox で、ページをタスクとみなし、一覧のソート順序を Date Modified（更新順）にしてアイコンフィルターも適用すると、ほぼ実現できます（ただし購読の機能はなく「自分のアイコンを書き込んだこと」を購読とみなす）。筆者としては、この概念を盛り込んだタスク管理はキラーアプリになるポテンシャルがあるとさえ思います。特に自律的にあちこち関わりつつ、活発に議論しながら進めていく今風の組織（たとえばティール組織）と相性が良いはずです。ティール組織も現在は出社など物理的に固まって密に活動するのが主流のようですが、これを開拓できればリモート化でも実現できる気がするのです。

=== Chapter-22 □第5部「高度なタスク管理」===

高度なタスク管理の考え方や手法をいくつか扱います。

特にテキストベースのものを扱います。タスク管理にて扱うタスクとは「言語で書かれたもの」であり、言語で書かれたもの――つまりテキストを上手く扱うことが成功の鍵を握ります。といっても、ここまで見てきたように、通常はシンプルな手段や既存のツールを使えば事足ります。「テキスト」などというコンピュータ寄りの表現を利用者が意識することはあまりありません。そうしなくても済むように手段やツールが整備されているからです。一方で、それらでは満足できないことがあります。

満足できない場合は自製するしかありません。自分の、自分による、自分のためのタスク管理を自分で開発することになるのです。そうはいっても、プログラマーでもなければツールなどつくれませんが、それでもテキストの扱い方を工夫すれば、ある程度のところまではつくれます。本チャプターでは、テキストなるものを上手く飼いならして、自分なりのタスク管理をつくるためのヒントを扱います。さらに後半では、管理の常識を越えた、新たな働き方を実現するための新しいタスク管理も提案します。

なお本チャプターではデジタルを想定しますが、根底の考え方はアナログでも同じため応用は一応可能です。たとえば手帳とペンでタスク管理を運用する人が、自分なりのタスク管理システムをつくりたい場合にも重宝するかもしれません、しかしながら厳しいとは思います。デジタルの利便性が前提となっているからです。本チャプターでもアナログによる実現は想定しません。

=== Chapter-23 プレーンテキストによるタスク管理 ===

はじめに

プレーンテキストとは

タスク管理は言語で扱うものであり、読み書きの営みは避けられません。ですので読み書きに手間がかかると容易に形骸化してしてしまいます。それでも仕事の文脈なら我慢して使うでしょうが、個人タスク管理ではそこまでしないでしょう。

では、この「読み書きのしづらさ」はどうやって解消すればいいでしょうか。幸いにも先駆者がいます。プログラムコードや大量のドキュメントを扱うエンジニアです。彼らはデジタルな読み書きの考え方とツールを知っています。少しでも生産性を上げるために研鑽や工夫を惜しまない人もいます。そのような人達によって蓄積されてきた叡智があります。

この叡智は、一言でいうと**プレーンテキスト（Plain Text）**です。直訳すると「素のテキスト」となりますが、キーボードやタッチパネルで文字入力したときに入力される文字情報を指します。コンピュータが扱う共通言語ともなっており、プレーンテキストは PC でもスマホでもあらゆるデバイスやツールで扱うことができます。しかし、文字サイズや色やリンクといった装飾情報は含んでいません。たとえば Microsoft Word で「大きな赤色の文字」を書いたとして、これを LINE や X に貼り付けてもそのとおりにはなりません。おそらく装飾を失った、単なる文字が貼り付けられるでしょう。装飾情報はツール固有であるため、他のツールでは通用しないのです。しかし元の文字情報、プレーンテキスト部分は共通言語であるため通用します。

私たちは言葉を読み書きしますが、デジタル上で読み書きするための共通言語がプレーンテキストなのです。

小難しく書きましたが、以下二点を押さえてください。

- 1: デジタル上で読み書きを扱う際には「プレーンテキスト」を扱う
- 2: プレーンテキストとは装飾の無い、素の文字情報を指す

アナログのような柔軟タスク管理をデジタルでやるには

手帳などアナログな手段でタスク管理する人も多いと思います。デジタルの方が一般的には便利ですが、なぜあえてアナログを使うのでしょうか。

その理由の一つが**言語情報を自由自在に扱えるから**です。紙面の好きな場所に書けますし、要らなければ打ち消し線を引いて潰したり、重要なタスクを目立たせたりできます。紙面やインクの許す限り、どこにでも、どのようにでも書けるのです。特別なスキルは不要で、読み書きができるのなら誰でもできます。

逆を言うと、デジタルの流儀を知り、スキルを身につければ、アナログと同じように自由自在に扱えるようになります。すでに述べたとおりデジタルには操作の早さ、修正のしやすさ、データ同期によるアクセスのしやすさといったメリットがあるのでした。つまり、**アナログの専売特許だった「自由自在」を、メリットの多いデジタルにおいても実現できる**ようになるのです。

そして、そのデジタルの流儀やスキルにあたる部分、その根本がプレーンテキストです。デジタルな世界ではプレーンテキストで読み書きされるので、プレーンテキストについて知ることが「自在に扱えるようになること」への近道となります。

プレーンテキストを知る

プレーンテキストによるタスク管理を始める前に、まずはプレーンテキスト自体の解説をしていきます。すでにご存知の方は読み飛ばしてください。

紙やペンを自在に使える人は多いと思いますが、なぜかというと紙やペンの仕様を知っているからです。たとえば文字を消したい場合は鉛筆（シャーペン）と消しゴムを使いますし、消す回数が少なければボールペンで済みます。あるいはボールペンでミスをしても、打ち消し線を引いたり、すぐ別にスペースに移動したりすればどうとでもなるので、紙の広さや枚数を潤沢に用意する手もあります。何らかのレイアウトがほしい方は、自分で一から線を引いてつくらなくても市販の方眼紙や日記帳が使えます――と、当たり前に聞こえるかもしれませんが、これは私たちがアナログの仕様を知っているからです。

さて、デジタルの仕様は、ここまで書いたようにプレーンテキストです。本節ではプレーンテキストの仕様を解説することで、アナログと同じように自在に使えるようになることを目指します。

はじめに

- PC 、かつ Windows を対象とします
  - スマホは当てはまらないかもしれません、Mac など他の OS も当てはまらないかもしれません
  - 適宜読み替えてください
- 可能ならばエディット領域を適当に用意して、動かしながら試すと良いでしょう
  - 複数行入力が可能なものなら何でも構いません
  - メモ帳などのテキストエディタ、ウェブサービスやウェブサイトの入力フォーム etc
  - 筆者の方でも用意してみました: https://stakiran.github.io/textarea/
- 以下は解説しません
  - 基本的な文字入力方法

より本格的に学びたい方は、拙作ですがこちらの書籍もおすすめします。タスク管理無関係の、執筆者向けの本ですが、プレーンテキストを知って使いこなすための解説にも力点を置いています。

行を並べる世界です

プレーンテキストは行を並べる世界です。

たとえば下記は「あああ」という行が 3 つあります。

あああ
あああ
あああ

いいい

ううう

その後、空行 と呼ばれる空の行が 1 つあって、「いいい」行があります。さらに空行を 2 つはさんで、「ううう」行があります。

このようにプレーンテキストは行から構成されます。アナログだと二次元的に縦横に自由に書き込むことができますが、プレーンテキストではできません。行を並べることしかできません。したがって、タスク管理を行う場合も「1-行 1-タスク」のようなルールで運用されることが多いです。

使える文字は様々です

プレーンテキストで扱えるのは文字だけですが、色んな文字が使えます。

ひらがな
カタカナ
漢字
空白文字　半角スペース「 」、全角スペース「　」、タブ「	」
記号類その1 （^_^） ← このように顔文字もつくれます
記号類その2 ■ ® ※ ∝ ∞ ……、「きごう」で変換すると入力できます
絵文字 □ □ □ □ □、「うさぎ」「でんしゃ」などで変換すると（絵文字があれば）入力できます

タスク管理を行う場合、単調な見栄えにいかにアクセントをつけてわかりやすくするかは肝ですが、これらの文字を上手く工夫することになります。たとえば「ごみ捨て」タスクを終了させたい場合、以下のような書き方が考えられます。スペースで微妙に間隔を開けたり、Xも大文字だったり小文字だったりと細かいですが、どれが正解ということはなく、個人の好みとなります。

□ごみ捨て
□　ごみ捨て
[X]　ごみ捨て
X　ごみ捨て
xごみ捨て
xごみ捨て

余談ですが、使える文字の一覧を手早く見たい方は拙作のこちらの記事も役に立つと思います: 半角英数字記号と全角英数字かなカナ記号の一覧まとめ #記号 - Qiita

保存する単位も様々です

アナログでは紙に書くだけで自動的に残りますが、プレーンテキストでは何らかの形で保存する必要があります。

PC 上で行う場合、テキストエディタで開いて、書いたものを **ファイル** として保存するのが最も馴染み深いでしょう。たとえば task.txt などです。この task.txt を開けば、書いたものが見えます。続きを書いて保存することもできます。

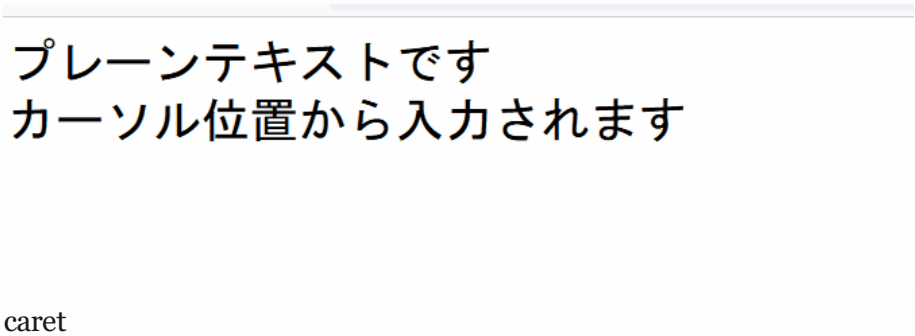
何らかのアプリを使う場合、アプリ上で書いて保存するか、あるいは自動保存されます。このとき使われる単位は様々で、たとえば以下のような名前が使われます。

- プロジェクト
- ワークスペース
- ページ
- ファイル
- ノート
- ドキュメント

タスク管理を行う場合も、この保存単位は意識することになります。ファイルで運用する場合、task.txt に全部書くのか、2024_07.txt みたいに月単位で書くのか、それとも「タスク」フォルダをつくったあとに「引っ越し.txt」「案件A.txt」みたいにタスク単位でファイルをつくるのかなど、様々考えられます。

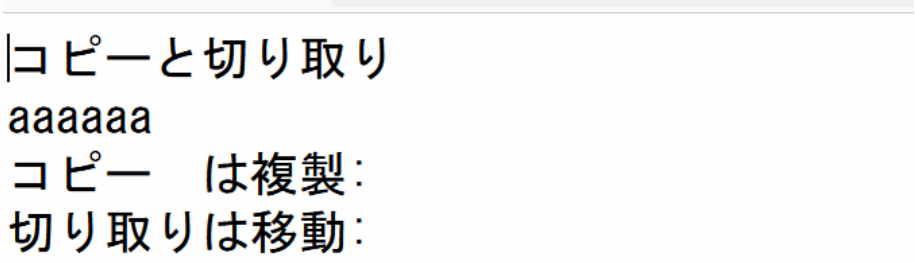
カーソルを中心に操作します

プレーンテキストはカーソル位置から操作します。文字の入力も消去も同様です。以下に例を示します。



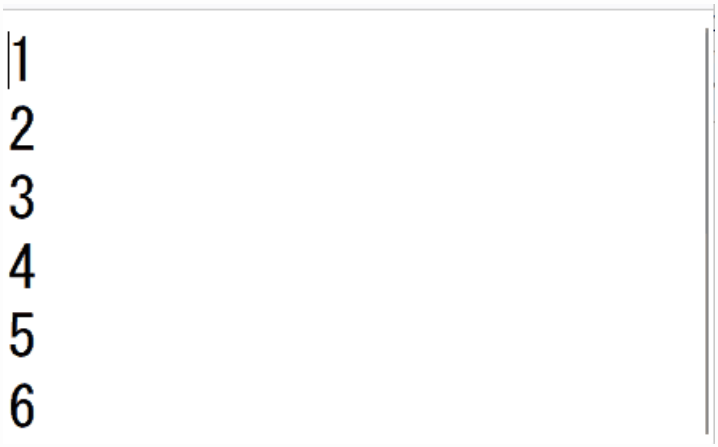
caret

範囲選択してから行うコピーや貼り付けなども操作の一つです。以下に例を示します。



caret

また行が長いテキストのどこを表示するかについても、現在カーソルが存在する位置の周辺が常に表示されます。仮に 300 行のテキストがあるとすると、おそらく画面内に表示しきるのは不可能でしょう（仮にできたとしても文字が小さくて読めません）。行末の付近を表示したいなら、カーソルを行末まで移動させる必要があります。以下に例を示します。



caret

厳密には「表示範囲」と「カーソル位置」は別物で、カーソル位置を変えずに表示範囲だけ変えることもできます（いわゆるスクロール操作がこれです）が、何かしら操作を行うとカーソル位置まで戻ってしまいます。結局、操作はカーソル位置基点でしか行えないので、常にカーソルを移動させる必要があります。

ここまでいくつかの例を示しましたが、何が言いたいかというと**プレーンテキストの操作はカーソル操作をいかに上手くやれるか** にかかっています。アナログにおいて、ペンを所定の位置に素早く持っていくのはかんたんですが、デジタルでこれに相当するのがカーソルを操作することです。

タスク管理を行う場合も、当然ながらカーソルの操作は頻発します。上述した GIF は筆者による手慣れた操作であり、これくらい自在に行えるとペンのように難なく扱えます。逆に、操作がままならないと、カーソルキー（矢印キー）の連打や押しっぱなしなどで原始的に移動するくらいしかできなくなり消耗しやすいでしょう。消耗しやすいと形骸化に繋がります。

デジタル特有の操作も可能です

デジタルな手段の良い点は、アナログの制約を超えられることです。プレーンテキストの操作においても便利なものがいくつかあります。例を示します。

- 複製がかんたんに行えます
  - いわゆるコピペ（コピー＆ペースト）
- 検索ができます
  - アナログだと人間が目で見て探す必要がありますが、デジタルだと一瞬です
- 失敗してもすぐに消せます

タスク管理では同じタスクを使いまわしたり、どこかに書いたはずのタスクを探したり、もちろん文字入力も行うので入力ミス時に修正したりもしますが、アナログのときよりもかんたんに行えます。かんたんに行えるからこそ、アナログでは実現できなかった管理も実現できようになります。たとえば人は（些細な行動やルーチンも含めれば）一日に数十ものタスクを抱えていますが、これのすべてを管理することも可能です。アナログのペンと紙で行うのは苦しいでしょう。デジタルだからこそです。

まとめ

プレーンテキストを知ると題して、プレーンテキスト自体の性質をざっと解説しました。

- 行を並べます
- 使える文字は様々です（絵文字もあります）
- 保存する単位も様々です（ファイルだけではありません）
- カーソルを中心に操作します（ので、カーソル操作に慣れることが大事です）
- 複製、検索、修正が用意などデジタルの恩恵があります。

アナログだと紙とペンという制約に引きずられますが、デジタルではこのプレーンテキストの制約に引きずられます。これらを理解しておくと、この後プレーンテキストによるタスク管理を考える際にも役に立ちます。

プレーンテキストを扱う道具を知る

続いてプレーンテキストを扱うための道具について解説していきます。こちらについても、ご存知の方は読み飛ばしてもらって構いません。なおディスプレイやキーボードといったハードウェア（デバイス）の話はしません。

まず道具は大別すると以下の 5 種類です。

- 1: 文字入力エンジン（IMEと呼ばれます）
- 2: エディットツール
- 3: ファイル形式と記法
- 4: 情報管理ツール（保存単位を扱うツールの話）
- 5: ヘルパー

1: 文字入力エンジン

IME(Input Method Editor) の名前で知られており、私たちが普段文字入力を行う際にバックグラウンドで動いているプログラムです。これがないと文字入力、特に変換は行えません。

メジャーなのは以下の 3 つです。

- MS-IME（標準）
- Google 日本語入力
- ATOK

通常、私たちが意識することはありません。何もしていなければ標準の IME が使われています。Windows なら MS-IME ですし、Mac だと「日本語入力プログラム」です（昔は「ことえり」という IME でした）。これら標準が気に入らない場合は、Google 日本語入力や ATOK を別途インストールすることになります。昔は標準 の IME は低性能、Google 日本語入力は語彙力とスピード、ATOK は変換の精度が特徴でしたが、最近ではほとんど差がなくなっています。標準の IME で問題ありません。

では、どんなときに意識するでしょうか。細かい設定やカスタマイズは奥深いので割愛しますが、よく使われるのが **辞書登録** です。たとえば「じしょ」で変換すると「辞書」という漢字を出せますが、これは「じしょ」という読みに「辞書」という漢字があらかじめ登録されているからです。これをカスタムで登録できる機能が IME にはあり、よく使う単語を登録しておくと便利なのです。

筆者の例をいくつか挙げます。

- 「だん」で「□」
  - □ を完了マークとして使うと便利なのでよく使います
  - Done の読みで「だん」です
- 「ぼうめいたい」で「忘迷怠」
  - 忘迷怠についてはすでに解説しましたが、筆者の独自用語です
  - 独自用語を使っている場合、通常の変換では出せないので、このように登録しておくと便利です
- 「になって」で「担って」が 出ないようにする
  - **Google 日本語入力には** 抑制単語機能があり、変換で出ないようにできます
  - 「〜になって」という言い方はよく使いますが、その度に「担って」が出てきて鬱陶しいので、出ないようにしています

2: エディットツール

入力先となるツールのことです。エディタと呼びます。

主に 3 つあります

以下のとおりです。

- テキストエディタ
- IDE
- ウェブアプリ

テキストエディタはプレーンテキストの編集に特化したエディタです。動作が軽く、オフラインでも動作します。動作面は重要であり、プレーンテキストで管理タスクを行う際にはまず考えたい選択肢です。デメリットとしては、本当にプレーンテキストを書くことしか能がないため、それ以外で不便になりやすいことです。たとえばファイルの管理を行う機能はありません。もう一つ、見た目が味気ないため、見た目を重視する人には厳しいかもしれません。

IDE は統合開発環境とも呼ばれ、元はプログラミングをこれ一つで行うための「エディタ機能も付いた多機能道具箱」でしたが、プレーンテキストを扱うのにも優れています。これを使ってタスク管理を行う人もいますし、筆者も Mac ではこれを使っています。昔は日本語の扱いに難がありましたが、今ではこれだけで書籍の執筆さえ行えるほどになりました。メリットは豊富なカスタマイズ性の高さと、検索やファイル管理などの周辺機能が充実していることです。デメリットは習熟に時間がかかることです（元はプログラマー向けのツールです）。IDE はテキストエディタと同様、PC 上でオフラインで動かすものですが、最近ではオンラインで動かせるものも出てきています。

ウェブアプリはオンライン前提で動くツール全般を指しますが、エディット機能を持つものも少なくありません。日記やブログを書くためのサービスもそうですし、X など SNS もそうなら、Slack や Teams などのビジネスチャットもある意味そうです。基本的に複数人で使うか、公開するかが前提となっていますが、非公開にしてひとりで使えばエディタとしても使えます。メリットは、現代人であればおそらく最も慣れている（本章では対象にしていませんがスマホでも使えます）ことです。デメリットは、エディタではないためテキストエディタや IDE ほど利便性には優れないことと、オンラインであるため動作の重さや接続切れといった問題が生じうることです。またデータはサービス側に保存されるため、アカウントを停止されたりサービスが終了したりするとデータも失ってしまいます。

3 つの特徴を比較

表形式で特徴を整理してみましょう。

手段	動作の軽さ	オフラインか	エディタのポテンシャル	習熟のしやすさ	カスタマイズ性
テキストエディタ	□	□	□	□	□
IDE	□	□	□	□	□
ウェブアプリ	□	□	□	□	□

では、プレーンテキストでタスク管理を行う際のおすすめはどれでしょうか。答えると**まずはテキストエディタを検討してください。IDE でも行けるなら構いません。ウェブアプリは「無い」と考えてください** ――となります。

まずウェブアプリは外してください。プレーンテキストを扱うには貧弱ですし、何よりオンラインで通信が発生することが最大のネックです。オンラインという性質上、多少アプリの出来と通信回線が良くても軽微な待ち時間や接続不良はどうしても生じます。プレーンテキストをガシガシ操作している最中に、アットランダムでこれらが起きるのは本当にストレスです。たとえば数秒の遅れであっても、です。車でたとえるなら、前方の車が急にアットランダムで速度を緩めるようなものです。秒数的には大した違いにはなりませんが、イラッとする感じがわかるかと思います。ウェブアプリは、タスク管理を行う手段自体としては優秀ですが、**プレーンテキストによるタスク管理の手段としては貧弱です**。あえて言うなら論外です。選択肢からは外してください。

とするとテキストエディタか、IDE かになりますが、どちらでも構いません。ご自身に合うものを使ってください。ただし標準のエディタでは物足りなくなることが多いため、人気の無償エディタか、有償のエディタを使うのがおすすめです。もちろん、まずは標準のものを使ってみて、物足りなくなってから別のを試すこともできます。強引ですが家でたとえると、テキストエディタは田舎の家で、IDE は都会の家のようなものです。テキストエディタは田舎なので自分で DIY を頑張る形になりますが、小回りがききます。IDE は都会であり色んな道具やグッズやサービスを駆使して比較的手間なく工夫できますが、それらを学ぶコストはかかりますし中身もブラックボックスなので小回りがあまり利きません。どちらが合うかは人それぞれです。

各エディタの例

最後に現存するエディタの例を挙げておきます。

※なおプログラマーの方にはお馴染みのエディタ類（Vim や Emacs など）は割愛します

- 「テキストエディタ」
  - 標準
    - Windows ではメモ帳
    - Mac ではテキストエディット
      - プレーンテキストを扱う場合は、設定からフォーマットを「標準テキスト」にしておく必要があります

    - 無償
        - Windows では SAKURA Editor
        - Mac では mi、CotEditor
    - 有償
        - Windows では秀丸エディタ
            - 筆者は普段これを使っています
- 「IDE」
    - 無償
        - Visual Studio Code
            - VSCode の略称で親しまれています。IDE のデファクトスタンダードです
- 「ウェブアプリ」
    - チャット
        - LINE、Teams、Slack
    - ノート
        - Notion、Google Docs、OneNote
    - Wiki
        - Scrapbox、esa
    - ブログ
        - note、はてなブログ
    - SNS
        - X
    - BTS(バグ管理システム)
        - GitHub

テキストエディタにせよ、IDE にせよ、選択肢はあまり多くありません。エディタというジャンルは歴史が深い上に、参入も容易ではないからです。特に IDE は、2024 年現在では VSCode がデファクトスタンダードだと考えて差し支えありません。

IDE を試したいなら VSCode を試してみてください。これで挫折するなら IDE はまだ早いか、向いていません。テキストエディタにしてください。

3: ファイル形式と記法

自然言語には「英語」や「日本語」といった言語の種類があります。また日本語にも口語体や文語体、小説や技術文書、箇条書きや散文など色んな書き方があります。

プレーンテキストも同じです。具体的には **ファイル形式** と **文法** があります。

ファイル形式 とは、プレーンテキストとして記載する情報をどんなファイル（の種類）で扱うかを示したものです。以下に例を挙げます。

- task.txt（テキスト形式）

- task.md（Markdown形式）
- task.json（JSON形式）

最初の txt は汎用的な形式で、たとえるなら「自然言語」のようなものです。英語でも日本語でも何でも書けます。最後の json は機械用の汎用的な形式で、たとえるなら「プログラミング言語」のようなものです。人間が通常コミュニケーションを取るためにプログラミング言語を使うことはありませんが、同様に、人間が通常テキストを読み書きする際に json を扱うことはありません。json はあくまでプログラムが情報を扱うための構造、つまりは「データ」にすぎません。プレーンテキストによるタスク管理として使うこともありません。

真ん中の md は Markdown 形式で、特定の表現をつくるために設計されたものです。これを**マークアップ言語** と呼びます。たとえば「見出し」を表現するのに # 大見出し と書きます（※1）。半角のシャープを使え、というルールです。そうすると、プログラム側でこれを読み取って、実際に大見出し（文字を大きく表示したり目次に含めたり）で表示してくれます。つまり「こういう感じで書けば、こういう感じで表示しますよ」が事前にルール化されているので、それに従って書くことで、そう表示させることができるのです。プレーンテキスト自体は素のテキストであり、装飾情報は持たないと冒頭で述べましたが、マークアップ言語はいわば **書くときはプレーンテキストで、読むときは（プログラムの力で装飾情報をつけて）読みやすく**という仕組みなのです。

つまりタスク管理を行う際は、次のいずれかを使うことになります。

- 汎用的な形式 → txt
- マークアップ言語 → たとえば Markdown

次に **記法** ですが、上述した Markdown や JSON は記法です。「特定の表現をつくるために設計されたもの」がまさに記法です。つまり Markdown は Markdown 記法ですし、JSON は JSON 記法とも言えます。

ファイル形式と記法の関係ですが、記法をファイルの形で示したのがファイル形式です。Markdown 記法をファイルで示す場合、.md をつけて task.md とするのです。この .md の部分は拡張子と呼び、記法ごとに存在します。その気になれば、独自のタスク管理記法を考案して、独自の拡張子をつけることもできます。拡張子は（個人で使う分には）自由に使っていいので、たとえば筆者（sta）が独自のタスク管理記法 sta 記法なるものをつくって、task.sta としてもいいのです。

この記法がどこで役に立つかというと、**テキストエディタや IDE では拡張子ごとに表示設定を変えます**。txt には txt の表示設定、md には md の表示設定があるわけですね。たとえば md の表示設定では、# 大見出し を強調表示したり、目次を自動生成したりできます。これはエディタが md の表示設定としてそのような設定を組み込んでいるからなのです。つまり、独自に sta 記法などをつくった場合でも、（エディタの表示設定を自分でつくれるのなら）独自の表示を行わせることができます。ただし、ここにはエディタの高度なカスタマイズが必要となるため、本書では割愛します。

ここまでをまとめます。

- プレーンテキストには様々な記法が存在し、記法ごとにファイル形式（拡張子）なるものがある
- ファイル形式がなぜ要るかというと、形式ごとに（拡張子ごとに）エディタが表示設定を独立させるため
- 主な記法は以下
  - テキスト形式(.txt)。汎用的
  - Markdown形式（.md）。マークアップ言語
- その気になれば独自記法もつくれる

タスク管理を行う場合、いずれかの記法を使うことになります。汎用的な .txt を使って好き勝手に書いてもいいですし、Markdown に従うこともできます。Markdown は文章を書く記法のデファクトスタンダードですから知っておくと何かと便利です。あるいは、自分で独自の記法を考えることもできます。ただエディタをカスタマイズして表示設定を作る部分が高度（本書でも解説しません）です。単に自分で書いて、自分で見るだけなら、.txt にて独自記法を書くだけで問題ありません。

- ※
  - 1 マークアップ言語としてよく知られたのが HTML です。普段見ているウェブサイトは、HTML で書かれたものをブラウザが読み取って装飾をつけて表示したものです。HTML では大見出しは `<h1>大見出し</h1>` と書きます。いちいち囲まないといけないので、人間が書くにはたいへんです。Markdown だと # をつけるだけで済むので、比較的楽です。そういう意味では、Markdown は軽量マークアップ言語と呼ばれることもあります。人間にとってより軽量に、かんたんに書きやすいからです。

4: 情報管理ツール

ファイルなど保存単位を扱うことを情報管理と呼ぶことにします。プレーンテキストを扱う際は、この情報管理の目線も事実上要求されます。

アプローチとしては以下があります。

- 1 使わない
- 使う
  - 2 ファイラー
  - 3 ウェブアプリ

1 の使わない選択肢は、保存単位内で構造をつくるということです。たとえば日単位でタスクリストをつくる形の運用がしたい場合、情報管理的に考えるなら 2024-07-30.txt のように日単位でファイルをつくろう等と考えますが、1 のアプローチはそうではなく、task.txt の中に ■2024/07/30 のようなエリアをつくろう、といった考えをします。戦略の章にて Monolith を取り

上げましたが、まさにこのアプローチです。実は、プレーンテキストでタスク管理したい場合はこのアプローチがベストです。**情報単位間（たとえばファイル間）を行き来するコストは高い** ので、極力減らした方がいいのです。記法やエディタの使い方を工夫できれば減らせます。

2 のファイラーは、ファイルを管理するツールであるファイラーを指します。標準でも備わっており、Windows でいうエクスプローラー、Mac でいう Finder がこれにあたります。フォルダがあって、その中身を表示するウィンドウがあって――といった使い心地ですよね。あれもファイラーなのです。タスク管理では、標準のファイラーでも十分機能します。ただし運用のルールをきちんと定めることと、ルールを守るための日々の行動・メンテナンスをきちんと行うことが求められます。

仮に 1 日 1 ファイルをつくる運用にするなら 2024/07/30 になったから 2024-07-30.txt をつくる、前日のタスクリストからも転記するといった行動も自分で行わないといけません。またファイルが溜まったら整理もしたくなるでしょう。つまり**ファイラーのアプローチでは整理整頓の営みから逃れられません**。一方で、シンプルな運用を確立できるのなら逃れることもできます。たとえば 1 フォルダ内にすべてのファイルをフラットに置くようにすれば、整理は要りません。1 日 1 ファイルつくるとしても、年で高々 365 個ですから、実は整理など不要です。むしろ私たちは整理したがる欲求を持っており、いかに抗えるかの戦いとなります。

標準以外のファイラーもあります。前述の VSCode はファイラーの機能を持っており、VSCode の画面内でファイルの追加や削除ができます（検索もできます）。またTablacus Explorerのように、エクスプローラーよりも便利な機能とカスタマイズ性を備えた高度なツールもあります。そもそもエンジニアなど PC の玄人は、エクスプローラーのようなグラフィカルなファイラー自体を使わずコマンドで済ませており、多数の便利コマンドを駆使してファイラーと同等レベルの利便性を確保していたりもします。要はファイルを上手く運用できればそれでよく、方法は色々あるのです。

3 のウェブアプリは、PC 上にファイルを保存する以外のアプローチです。情報はクラウド上に保存され、利用者が見ることとなる単位も様々です。ページ、ノート、ドキュメントもあればプロジェクトやワークスペースといった高級な単位で扱われることもあります。

5: ヘルパー

プレーンテキストを書くことを支援するツールや仕組みが存在します。前述した IME の辞書登録もその一つですが、他にもあります。

- Text Expansion
  - たとえば `d[[` と入力すると現在日付（2024/07/30など）が挿入されます
  - このように「短縮語」から「スニペット」を挿入する手法です
    - `d[[`が短縮語で、2024/07/30がスニペット
    - スニペットとして複数行の文章や、日付を始めとする動的な情報を扱うこともできる
- クリップボード履歴

    - コピーした情報を複数保持し、後で利用できるようにしたもの
    - 従来は専用ツールが必要でしたが、Windows 10 からは標準機能として使えます（Win + V キー）
- アウトライン
    - テキストエディタの機能ですが、見出しを検出して目次をつくるものです
    - 目次を見て内容全体を俯瞰したり、特定の見出しにジャンプしたりできます
    - これを使いこなすにはエディタ側のカスタマイズが必要で、高度なトピックスです
        - 一方で既存の文法を使うのならば、既存の設定をそのまま使えます
        - たとえば VSCode で Markdown を使うと、見出しベースの目次を（自分でカスタマイズせずに）使えます
- OCR
    - Optical Character Reader の略で、画像内の文字をプレーンテキストに変換する技術です
    - この技術を使うと、画像を見ながら手打ちで書く手間がなくなります
        - 特に画像からタスクをおこしたい場合の手間を減らせるため、人によっては重宝します
    - ツールとしてはGyazo Pro、PowerToysのText Extractorなどがあります

他にも様々なヘルパーがあります。要は **一から手打ちで書いたり、原始的なスクロールなどで探したりといった手間を減らす** 手段が色々あるということです。

おすすめの道具構成

ここまでを踏まえた上で、（プレーンテキストでタスク管理する際の）おすすめの道具構成を解説します。

文字入力エンジン――IME については、何でも構いません。重要なのは辞書登録であり、よく使う単語は登録しておきたいです。ただでさえプレーンテキストで見栄えに差がないため、いくつかの強調（特にタスクの終了を示す□や[x]のような書き方）をすぐに入力できると便利です。

エディットツールについては、テキストエディタ/IDE/ウェブアプリがありましたが、テキストエディタか IDE のいずれかにしてください。ウェブアプリは向いてない上にストレスフルです。すでに使い慣れたエディタがあるなら、それで構いません。プレーンテキスト自体はどのエディタでも編集できます。必要ならエディタを変えればいいだけですので、あまり肩ひじを張らず、自分にとって使いやすそうなものをとりあえず使えば良いです。

ファイル形式と記法については、まずは汎用的な .txt を使うと良いでしょう。記法は自分で考えれば良いのです。タスク管理は本質的に個人的なものである、とは本書の主たるメッセージですが、プレーンテキストでも例外はありません。十中八九「**自分は** こういう記法でこう書くのが合っている」のような癖が出てきます。.txt であればどう書いても問題ないので、まずは .txt で構いません。そのうち慣れてくると「見栄えにメリハリをつけたい」「見出しを書いて目次も出せる

ようにしたい」といった希望が出てきますが、そうなったら次は Markdown 記法に手を出しましょう。Markdown 記法は、学ぶのは少々大変ですが、文章を書くのに十分な能力を持っており各エディタも色んな便宜をサポートしています。Markdown に従えばこれら恩恵にあやかれます。ただし学ぶのが少々大変なので、まずは .txt で慣れるのが良いと筆者は考えます。本書でも解説はしません（拙作に譲ります or 解説記事も多いので独学も可能です）。

情報管理ツールについては、まずは Monolith 的に単一ファイルでの運用をおすすめしたいです（つまりファイラーは使わない）。それで苦しくなってきたら、別途ファイルを増やしていき、ファイルの数自体が増えてきて初めてファイル管理を考えたいです。いきなりファイル管理を考え始めても難しいので、まずは小さく始めたいのです。

ヘルパーについては、なくても困りませんので無理して使う必要はありません。あれば便利なのは間違いないので、余力があれば一つずつ勉強・試行して取り入れていくのが良いと思います。

まとめ:

- IME は何でも良い
  - むしろ重要なのは辞書登録。強調を担う書き方はすぐに入力したい
- エディタはテキストエディタ or IDE のいずれか
  - エディタ自体は重要ではない。どのエディタでもプレーンテキストは扱えるため
  - 自分に合ったもの、合ってそうなものを使えば良い
- ファイル形式と記法は、まずは .txt を使う
  - おそらく自分の癖に従った記法が出てくるので、素直に反映する
  - 慣れてきて不便を感じてきたら Markdown にも手を出す
- ファイルの管理・運用はまずは考えないで、単一ファイルに全部書く、で良い
  - ファイルが増えてきてから考えれば良い
- ヘルパーは無くて良いが、あれば便利なので余力があればぜひ

PTTM の基礎

ここまででプレーンテキストを扱うための基礎が整いました。ここからは **プレーンテキストによるタスク管理（Plain Text based Task Management, PTTM）** の話に入っていきます。

PTTM の目的は「自由自在」の獲得

プレーンテキストタスク管理（以下 PTTM）とは、プレーンテキストとエディタを用いてタスク管理を行うことです。

目的は **自由自在にタスク管理を行うこと** にあります。アナログに管理している人は、紙とペンを用いて自由自在にタスク（を示した単語や文章や記号、場合によっては絵も）を扱っていると思いますが、この感じをデジタルでも行えるようにしたいのです。

タスク管理というと、特にデジタルでは煩雑な画面、画面遷移、通信や処理による表示待ちなどを伴いますが、これらは無視できないストレスになりがちです。そもそもこのようになっているのはプロジェクトタスク管理の影響で、ビジネスにも通用して、かつ多人数で管理することも成功させるためには、どうしても煩雑になってしまいます。統一感も必要（むしろ重要）ですから、皆に同じツールを使わせることにもなります。

一方で、個人タスク管理を行う場合は、そこまでは要らないことが多いです。むしろ、自分なりのタスク管理を定着させるために、いかに自分に馴染めるようなやり方や工夫ができるかを探す方がはるかに大事です。既存の、ビジネス向けのタスク管理ツールでは中々難しいことです（できるにしてもツールのやり方に私たちが無理やり適応せざるをえません）。かといって、自分でタスク管理ツールをつくるのはエンジニアでもなければ不可能ですし、エンジニアであっても多大なコストがかかるものです。

自分用の、小回りの利くタスク管理を行うにはどうすればいいのでしょうか――すでに述べたとおり、プレーンテキストをエディタで扱うようにすれば、テキストをどう書きどう回すかの部分で工夫ができます。

PTTM≒タスクリストの運用

PTTM ではどのようにタスク管理を行うでしょうか。

言ってしまうと、タスクリストの運用を工夫するだけです。工夫として用いるのが属性です。

タスクリストの中で最も有名なのは **TODO リスト** であり、これはタスクとして「名前」と「状態」の 2 つのみがサポートされたものですが、すぐに破綻することでも知られています。例を挙げつつ、属性を用いて工夫するという点を見ていきましょう。

TODO リストの例を一つ挙げます。

ごみ捨て
新しい冷蔵庫を調べる
□家電量販店
通販
□上司が前買い替えた言ってたので話聞く
コバエうるさいので対処したい

2 つほどタスクが終わっています。たぶん冷蔵庫を調べるタスクの一環として、家電量販店と上司に聞く分は終わったということでしょう。この程度ならすぐ思い出せるかもしれませんが、仕事にせよ私生活にせよ、タスク管理を続けていると、このリストはあっという間もぐちゃぐちゃになりま

す。早ければ明日を待たずに破綻するでしょう。

そうならないためには、属性を上手く導入して、もっと上手く管理できるようにします。たとえば実行日属性を導入して、さらに毎日その日のタスクリストだけ完了させたらOK（デイリータスクリスト）とのルールも入れたとすると、以下のようになります。

■今日： 2024/07/29
ごみ捨て
新しい冷蔵庫を調べる
□家電量販店
□上司が前買い替えた言ってたので話聞く
通販

■2024/07/30

■2024/07/31
コバエうるさいので対処したい

■2024/08/01

コバエ対策が二日後に先送りされたことや、今日ごみ捨てをしたかったのにまだしてないことなどがわかります。通販については、新しい冷蔵庫を通販で調べたいのでしょう。おそらく退社後に調べられるでしょう。今日やるのが面倒なら、7/30（明日に先送り）や7/31（明後日に先送り）に記載を移すことで先送りできます。ごみ捨てについても、また忘れてしまわないように何らかの対策をした方がいいでしょう。これも今日やりたくないなら、たとえば明日 7/30 のところに「ごみ捨て忘れないための対策を考える」などと書いておけば、いったんは忘れずに済みます――と、このように時間軸を導入しただけでもだいぶ融通がきくようになりました。

もちろんこれは完璧には程遠いです。たとえば明日 7/30 になったら、どのように書き換えればいいのでしょうか。おそらく以下のようにするでしょうが、

■今日： 2024/07/30
ごみ捨て
新しい冷蔵庫を調べる
通販

■2024/07/31
コバエうるさいので対処したい

■2024/08/01

手作業でいちいち行うのは面倒くさいですよね。この手間をどうやって薄めるかは腕の見せ所です。我慢して受け入れるのもアリだったりします（し、日常生活の家事や仕事の雑務が泥臭い

のと同様、タスク管理も泥臭いのである程度は受け入れて慣れた方が有利ではあります）。

また、7/29 に終わった「□家電量販店」タスクと「□上司が前買い替えた言ってたので話聞く」タスクの扱いはどうすればいいでしょうか。消せばいいのでしょうか、それとも別の場所に記録として残しておきましょうか。後者の場合、どんなファイル名で残しましょうか。たとえば普段使うファイルは tasks.txt にして、過去の記録は tasks_old.txt とでもしておいてこっちに書いておけば、とりあえずはなんとかなるでしょう。以下に例を示します。

```
[task.txt]
■今日： 2024/07/30
ごみ捨て
新しい冷蔵庫を調べる
通販

■2024/07/31
コバエうるさいので対処したい

■2024/08/01

[tasks_old.txt]
■2024/07/29
□家電量販店
□上司が前買い替えた言ってたので話聞く
```

これはファイル自体を分けており、情報管理（道具の項で述べた 4:）の話です。おすすめとしては「まずは単一のファイルで行う」と書きましたが、過去行ったタスクという「記録」は、あっても邪魔なので別ファイルに移した方がいいのです。

さらに言えば、今日は今日のタスクに集中するべきであって、明日以降のタスクもあまり目に入れたくない、という人もいるかもしれません。その場合は tasks.txt を today.txt、tomorrow.txt に分けて、today.txt には常に今日のタスクが並ぶようにするのもアリかもしれません。ついでに tasks_old.txt も yesterday.txt に変更すると、次のようになります。

```
[today.txt]
■今日： 2024/07/30
ごみ捨て
新しい冷蔵庫を調べる
通販
```

[tomorrow.txt]

■2024/07/31

コバエうるさいので対処したい

■2024/08/01

[yesterday.txt]

■2024/07/29

家電量販店

上司が前買い替えた言ってたので話聞く

この 3 ファイル構成は今日、明日以降、昨日以前（記録）を示しているのでわかりやすそうです。しかしファイルが分かれているので、いちいち行ったり来たりするのが面倒くさそうです。じゃあ、おすすめのとおり、1 ファイルでまとめるとしたらどうでしょうか。色んな記法がありますが、たとえば以下のように線で区切ってみましょう。

■================

■今日

■================

■2024/07/30

ごみ捨て

新しい冷蔵庫を調べる

通販

●================

●明日以降

●================

■2024/07/31

コバエうるさいので対処したい

■2024/08/01

★================

★昨日以前

★================

■2024/07/27

・・・

■2024/07/28

・・・

■2024/07/29

□家電量販店

□上司が前買い替えた言ってたので話聞く

いかがでしょうか。1 ファイル内なので（エディタによるカーソル操作の技術にもよりますが）操作自体は楽になったはずです。慣れるまでは見づらいかもしれません。おそらくスクロールも頻発するでしょう。上記の例では些細な工夫をしており、明日以降のタスクを見たければ「●」で検索、昨日以前の記録を見たければ「★」で検索することで、すぐ飛べるようになっています（エ

ディタの検索機能を使います）。

もう少しお付き合いください。行数がやたら長いのを何とかできれば、もっと使いやすくなると考えたとしましょう。次のように 1 行 1 タスクで書いてみましょう。

```
2024/07/30 ごみ捨て
2024/07/30 新しい冷蔵庫を調べる
2024/07/30 通販

2024/07/31 コバエうるさいので対処したい
2024/08/01

2024/07/29 □家電量販店
2024/07/29 □上司が前買い替えた言ってたので話聞く
```

ずいぶんとスッキリしましたね。一見すると良さそうですが、タスクごとに日付がついているため、先送りが非常に面倒くさいです。いちいち `2024/07/30` のような日付を手で書かないといけません。逆に、これ以前の例で見てきたやり方は、タスク自体を別の行に移すだけで日付を変更できていたので便利だったのです。折衷案として、ヘルパーの Text Expansion を使うと、たとえば `d[[` と打って `2024/07/30` が入力される、のようなことができるのでほんの少し楽できます。

例はこのくらいにしておきましょう。

属性とファイルの分け方を工夫することで自分なりに運用していく、とのエッセンスを感じていただけましたら幸いです。これが PTTM です。

冒頭でも述べたとおり、PTTM も所詮はタスクリストを運用するだけです。ただし属性とファイルの分け方を工夫することで、何とか上手くいくように模索します。正解はありません。その人にあったやり方というものがあります。別の言い方をすると、あなたに合ったやり方はあなたにしかわからず、したがって自分で模索していくこととなります。模索という営みに耐えられない方は、PTTM からは回れ右しましょう。

先人の実践例

参考までに、PTTM を実践されていると思しき方の記事やページをいくつか紹介しておきます。PTTM からして筆者の造語であるため、事例を探すのは容易ではないですが、`プレーンテキスト タスク管理` などで検索すると少しは見つかります。

- なにもかもシンプルに。私がいまだにプレーンテキストを愛用している理由｜ライフハッカー・ジャパン
- 予定管理をテキストエディタで作る方法をAIに教えてもらった｜毎日AI生活

- 生産性を高めるプレーンテキストファイルの使い方10 | ライフハッカー・ジャパン
- プレーンテキスト最強に同意 | トドの日記2.0
- 様々なTODOアプリやタスク管理方法を試行した結果最終的にプレーンテキストに行き着いた話 - みんからきりまで
- テキストファイルで タスク管理 兼 備忘録 を実現する - j3iiifn's blog
- どのツールもしっくりこない人へ「Todo.txt」によるタスクマネジメントのススメ：3分LifeHacking - ITmedia エンタープライズ
- Todo.txt - wkpmm

眺めてみると本当に人それぞれだなとわかります。todo.txt というフレームワークを用いたものもあれば、- と x を使うシンプルなものもありますし、Markdown に頼っているものもありますね。

おわりに

基礎編は以上です。これだけでも自分なりに PTTM を始めることができるはずです。

とはいえ、PTTM には先人の知恵が色々とあります。次節では「設計」と題して、ご自身の PTTM を考える上で役立つ「よく知られたテクニックや考え方」を紹介していきます。

PTTM の設計

自分なりの PTTM を設計する際に役立つヒントを紹介します。

原則

PTTM の管理範囲を決める

Ans: PTTM としてどこまでをタスク管理するかを定めます。

生活や仕事のすべてを管理する必要はありません（もちろんしても OK です）。たとえば複数持つ仕事のうち、A案件だけを PTTM で管理してみようといったスモールスタートも可能です。

管理範囲を決めておかないと迷いが生じるので、さっさと決めてしまうことをおすすめします。選択肢は事実上二つで、特定の対象だけを管理するか、生活のすべてを管理するかの二択です。筆者としては後者をおすすめします。というのも、個人タスク管理では、たいてい複数の役

割や文脈をまたいだ検討が必要となるからです。仕事中にプライベートのタスクを少し微調整したい（逆も然り）、なんてこともよくあります。もちろん（特に）仕事とプライベートは、仕事側が機密的であるため安直にいっしょくたにはできないかもしれませんが、それでもなるべくいっしょくたに扱うことを目指してみてください。少し脱線しますが、ワークライフバランスにおいても、近年では仕事と生活をしっかり分けるのはむしろ難しくて、両者の境界はもっと曖昧だよね、もっと融通利かせた生き方ができるといいよねとする考え方が出てきています（ワークインライフやワークアズライフ）が、タスク管理も同じです。境界が曖昧なので、なるべく全部を管理してしまった方が融通が利きます。おそらく大半の労働者は（組織の一員として）機密性が要求されますので、仕事用とプライベート用の二つの PTTM を回すことになるでしょう。

動線化する

Ans: PTTM で使うプレーンテキストファイルは、毎日頻繁に使う場所にします。

動線については前の章で解説しましたが、毎日頻繁に通る場所を指します。このような場所があれば頻繁に目に入るので、忘れることなく対処やメンテナンスができます。あるいは忘れていても、見れば思い出せます。

PTTM ではテキスト（として書いたタスク）を頻繁に読み書きするため、動線化は必須です。RSAFサイクルを取り入れたいです。面倒くさいと考えるのではなく、面倒と感じないほど手になじませたいのです。正直言うと、この動線化が定着するかどうかが PTTM の成功の分かれ目とさえ考えます。

誰にも見せない

Ans: PTTM の中身はプライベートなものであるため、誰にも見せてはいけません。

個人タスク管理が形骸化する主因の一つは「恥ずかしいから」です。誰かに見られたり、見せてなくても見られる可能性があったりすると、心理的にセーブがかかって **一部のタスクを書かなくなります**。たとえばアダルトな営みに関するタスク、セクシャルマイノリティに関する手術、障害に関する通院や調査、現職や家庭には伏せておきたい転職活動などは、慣れていないと本人自身も恥ずかしくてタスク化（つまりタスクとしてテキストにおこす）できないものです。書くタスクと書かないタスクがあればあるほど、タスク管理は形骸化します。何を書くか書かないかの判断に認知コストを費やすのと、人間は元来面倒くさがりですから書かない側の自分に寄ってしまってサボってしまうためです。

この恥ずかしさのリスクを潰せている状態を **管理的安全性** と呼ばせてください。管理的安全性は極限まで確保したいです。誰かに見せない、見られないようにするのはもちろん、潜在的に見られる可能性も極力潰してください。そのような環境が無いのなら、何とかしてつくってください。特にありがちなのは、職場など複数人で過ごすことが当たり前の環境において PTTM を始める場合で、同僚などから「何してるの？」的なツッコミがよく入ります。これでは管理的安全性などあったものではありません。初期のうちは適当にごまかして、さっさと慣れてもらうのが良い

でしょう。慣れてもらって「そういうキャラ」と定着すれば、もう深入りはされなくなりいたずらに覗き込まれたり注目されたりすることも減るでしょう。

PTTM の管理対象を決める

Ans: PTTM はテキストを書くため割と何でも扱えますが、何を扱うか（というより**何を扱わないか**）は決めた方が良いです。

PTTM はテキストを書くため、その気になればタスクに限らず割と何でも管理できますが、向いていないこともあります。たとえば予定はカレンダーで管理した方がやりやすそうですし、だらだらと日記や思考を書いてもいたずらに時間を使うばかりで（タスク管理としては）あまり意味がないどころか情報量的に圧迫されてしまって読み返しづらくなります。読み返しがなくなると形骸化につながります。要するに何でも扱おうとすると破綻します。

なんだか上手くいかないという場合は、まずは何でも扱いすぎてないかを疑ってください。PTTM はプレーンテキストゆえに紙面上は無限に書けますが、これを使う人間側の認知能力が限られているので、自分のタスク管理に本当に必要なものをシンプルに扱うことを心がけてください。

PTTM で扱うべきではないものについては後述します。

振り返りを行う

Ans: PTTM はテキストの読み書きとメンテナンスを行うものなので、振り返りの習慣が身についていると強いです。

PTTM では残タスクの確認、明日以降のタスクの確認、終わったタスクの反映（完了にする）、その他タスクリストの軽微な修正――このような確認やメンテナンスを頻繁に行います。特に日をまたいだタイミングではメンテナンスの量も増えがちですし、週や月といった単位で俯瞰して反省したり今後を検討したりといったことも必要でしょう。振り返りの章でも DWMY、Day/Week/Month/Yearの頻度が存在することを述べました。定期的に振り返りを行って、メンテナンスにあてることをおすすめします。というより、振り返りの形で確保しないと中々メンテナンスできず、メンテナンスできないと形骸化に繋がります。

部屋やデスクも散らかりすぎると収拾がつかなくなるため、たまには掃除するわけですが、PTTM でも必要です。PTTM として普段使っているテキストを健全に保つためのメンテナンスを、こまめに行う必要があるのです。その機会を捻出するのが振り返りです。

詳細は振り返りの章を参照してください。

3 秒の壁を突破する

Ans: 新しいタスクを追加したい場合、3 秒以内に追加することを目指してください。

今現在忙しいときに、今すぐというわけではないが後でやっておきたい or やらねばならないタスクが発生したとします。どうすればいいでしょうか。PTTM を行っている場合、正解は「そのタスクをささっとメモしておく」です。メモしておけば、あとで見返したときに処理できるからです。私たちは記憶能力を過大評価しがちで「いやまあ忘れないでしょ」とメモするのを怠けがちですが、それで忘れないなら苦労はしません。忘れます。忘れないためには、その場でメモを完了させることが必要です（※1）。

筆者としては、一つの目安として 3 秒を掲げています。3 秒以内にメモを取れる（メモの完了ではなく開始、つまり書き始めるまでの話です）ことを目指してください。これができるくらい素早くメモが取れるなら、あなたの PTTM は手に馴染んでいます。逆に、3 秒以内にできないとしたら、まだ馴染んでいません。馴染んでいない手法やツールを使い続けるのは苦行なので、そのうち形骸化してしまうかもしれません。タスク管理は、特に PTTM はちょっとしたサボりが肥大化していって形骸化に繋がるのですが、この「さっさとメモしておく」場面は、サボりやすい場面の筆頭です。これを凌げるかどうかは PTTM 定着の目安であり、凌ぐためには 3 秒くらいでササッとこなせるくらいの軽さが要ると思います。これ以上かかってしまうと「面倒くさい」が勝ってしまい、サボります。

秒数にこだわりはなく、人によっては 5 秒でも 10 秒でも良いですが、ニュアンスとしてはできるだけ早く、それも「ササッと」のレベルで手間も感じず、息するように行えるレベルだと強調したいです。それくらい手に馴染んでようやく定着するものだと感じるからです。そういうわけで、3 秒の壁はぜひ壊してみてください。壊すための工夫は惜しまないでください。

- ※
  - 1 この素早さが最も必要なのは、ナレッジワーカーやクリエイターと呼ばれるような創造的な人達だと筆者は考えます。アイデアをメモするためです。アイデアはいつ湧くかわかりませんし、湧いたときにメモして捉えなければ未来永劫忘れます。アイデアの重要性は言うまでもないと思いますが、メモができないとその重要なものを逃してしまうのです。メモについてはメモの章もあるので参照してください。

扱わないもの

オルタスクは扱わない

Ans: オルタスク は直接扱わず、別のツールで扱います。

第 3 部では「タスクのようでタスクでないもの」としてオルタスクを紹介しました。コンテナ、イベント、モットー、メモ、タスクソースがあります。これらを PTTM で扱おうとしても分が悪いため、各章で述べたような専用のやり方やツールを使いましょう。

ただし PTTM にて限定的に扱えることはあります。コンテナは見出し表記などで区切る形で実現できます。■2024/07/30 のように日単位で時系列的に区切る例はすでに示しました。また、以下のようにインデント（字下げ）を用いれば階層的な表現もできます。いくつか例を示し

ます。

■引っ越しエリア（2スペースでインデント）
引っ越し
  引越し先の検討
    北海道を調べる
    地元を調べる
    都内で家賃安いエリアをもう一度調べたい
  全体スケジュール
    梱包や粗大ごみとか
    自転車要らないので捨てたい
  冷蔵庫買い替えておきたい
  （まだ洗い出し途中。。。）

■引っ越しエリア（2スペースでインデント ＋ 先頭に`・`もつけた）
・引っ越し
  ・引越し先の検討
    ・北海道を調べる
    ・地元を調べる
    ・都内で家賃安いエリアをもう一度調べたい
  ・全体スケジュール
    ・梱包や粗大ごみとか
    ・自転車要らないので捨てたい
  ・冷蔵庫買い替えておきたい
  ・（まだ洗い出し途中。。。）

■引っ越しエリア（Markdownのリスト表記）
- 引っ越し
  - 引越し先の検討
    - 北海道を調べる
    - 地元を調べる
    - 都内で家賃安いエリアをもう一度調べたい
  - 全体スケジュール
  - 梱包や粗大ごみとか
    - 自転車要らないので捨てたい
    - 冷蔵庫買い替えておきたい
  - （まだ洗い出し途中。。。）

またメモについては、むしろ普段 PTTM を行っているファイルに直接書くのが最も自然です。しかしメモが溜まると処理が苦しくなるので、振り返りを日単位以上の高頻度に行って、溜めずに処理していく（別のツールに移すなりタスクとして書き直すなり）べきです。PTTM が使えない場面（外出先でスマホしかなく PC が使えない等）であれば、別の手段でメモしましょう。後で

忘れないように、PTTM 側に「スマホに書いたメモを回収する」のようなタスクを入れておくとさらにグッドです。

ルーチンタスクは扱わない

Ans: ルーチンタスク についても、別のツールで管理します。

定期的に行うタスクを本書ではルーチンタスクと呼んでいますが、このルーチンタスクも PTTM で管理するのは厳しいです。たとえば毎週火曜日と金曜日に可燃ごみのゴミ捨てを行いたい場合、毎日、一日の最初（あるいは前日の最後）に火曜日もしくは金曜日であることを確認して、もしそうなら「可燃ごみ」タスクを追加する（あるいは常に追加した後、火金以外なら削除する）作業が必要です。非常に面倒くさいでしょうし、ルーチンタスクは他にも多数あります。やってられません。ルーチンタスクは、専用のツールで管理しましょう。

概念

デイリータスクリストを導入する

Ans: デイリータスクリスト（今日やることリスト）はぜひとも導入したいです。

戦略の章にて Dailer を取り上げましたが、タスクリストを日単位で区切る営みはぜひとも導入したいです。たとえば今日が 2024/07/31 なら ■2024/07/31 のようにして今日のエリアを設けて、今日はここに書いてあるタスクだけを終わらせたら OK とするのです。明日になったら ■2024/08/01 を今日エリアにします。こうすることで「今日はここまでやればいい」とのメリハリがつきます。人は終わりを設けてメリハリをつけないとガス欠しちゃいますので、この考え方はぜひともおすすめしたいです。

「曖昧な日付感」を導入する

Ans: 2024/07/31、2024/08/01 のような分け方よりも「今日」「明日以降」の方がシンプルです。

日単位でタスクを区切る際、どのように分けるかに悩まれるかもしれませんが、主に 2 つのやり方があります。

- 2024/07/31、2024/08/01、2024/08/02 など日付ごとに分ける
- 今日、明日以降、来週以降など意味的に分ける

日付ごとに分ける場合、見やすいですがメンテナンスが大変です。まだ書いてない日付の区切り（たとえば ■2024/08/25 のような文字列）を書く手間もありますし、おそらく「今日」は一番上に表示させたいですから、毎日今日に相当する部分を一番上に移動させる手間も生じます。

意味的に分ける場合、メンテナンスはある程度端折れますが、正確な日付（実行日）がわからず後回しにされがち――つまりはタスクが溜まりがちです。

可能なら日付ベースをおすすめしますが、メンテナンスが大変で挫折してしまっては元も子もありません。メンテナンスを軽くしたい場合は、意味ベースが良いでしょう。日付の扱いが曖昧ということで、タイトルは「曖昧な日付感」としています。

以下に 2 つのやり方で書いたものをそれぞれ挙げてみます（中身は ChatGPT で生成）。比較してみてください。

日付ベース

■2024/07/31
- プレゼン準備
- 開発環境設定
- 進捗報告

■2024/08/01
- ユーザーテスト
- バグ修正
- 仕様書更新
- チーム会議
- 新機能のアイデア出し

■2024/08/02
- プロジェクト計画見直し
- チームブレインストーミング
- プレゼン発表
- クライアントのフィードバック確認
- バグ修正

意味ベース（曖昧な日付感）

■今日
- プレゼン準備
- 開発環境設定
- 進捗報告

■明日以降
- ユーザーテスト
- バグ修正
- 仕様書更新
- チーム会議
- 新機能のアイデア出し

- プロジェクト計画見直し
- チームブレインストーミング
- プレゼン発表
- クライアントのフィードバック確認
- バグ修正

日付ベースの場合、2024/07/31 のように日単位で区切る他に、週で区切ったり月単位で区切ったりすることもできます。日単位よりはメンテナンスが楽になります。日報ならぬ週報や月報のイメージです。その分、いつどのタスクを行うかという日レベルの正確性も損なわれます。

意味ベースの場合、空行や区切り線などを工夫すれば「時期的な広がり」を持たせることができます。たとえば以下は空行が多いほど離れていることを示したものです。

■明日以降
- ユーザーテスト
- バグ修正

- 仕様書更新
- チーム会議

- 新機能のアイデア出し
- プロジェクト計画見直し
- チームブレインストーミング

- プレゼン発表

- クライアントのフィードバック確認
- バグ修正

1行離れていると1日分、2行離れていると2日分のように対応させてもいいですし、1行だと明日中/2行だと今週/3行以上は直近じゃなくても大丈夫を意味する、などにしても良いでしょう。何ならもっと雑に、感覚的に扱ってもいいです。空行だと縦長になるのが嫌であれば、以下のように区切り線の数（|を使いましたが-や=でも構いません）で表すこともできます。

■明日以降
- ユーザーテスト
- バグ修正
|
- 仕様書更新
- チーム会議
| |
- 新機能のアイデア出し
- プロジェクト計画見直し
- チームブレインストーミング
| | |
- プレゼン発表
|
- クライアントのフィードバック確認
- バグ修正

フォーマットを統一する

Ans: PTTM において見やすさは重要で、その見やすさを確保するのがフォーマットです。

PTTM はプレーンテキスト――つまりは文字情報ばかりを扱うため、動画や絵や図表と比べると見づらいです。仮に読書好きや活字中毒者がいたとしても、「自分が書いた、タスクに関するテキスト」は味気なくてつまらないため、やはり見づらく感じます。この見づらいテキストと毎日毎日向き合うことになるのですが、ここで挫けてしまう方も見受けられます。少しでも見づらさを減らして負担を減らしていくべきです。

そのために行えるのがフォーマットの統一です。正解はありませんが、こんなレイアウトで書こう、こんな記法で書こう、というのを自分なりに決めて、守るようにします。すでにここまで見てきた例でも、日単位のエリアは ■2024/07/31 のように ■ を使ってきましたし、一つのタスクは一行で書いてますし、タスクを表す場合に-や・をつけたりもしています。こんな感じで、どういうときにどんな風に書くかを統一するのです。

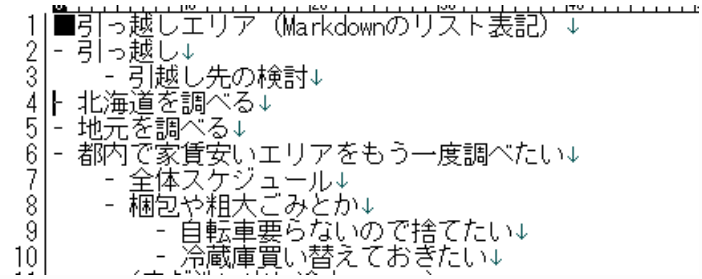
統一したフォーマットに従うことは、最初は苦しいかもしれませんが、そのうち慣れてきます。目が慣れてくると文字が滑らず、素早く読めるようになります。PTTM を続けたいなら絶対にやっておきたい投資です。

ちなみにフォーマットは自分で一から考える必要はありません。Markdown のような既存の文法に頼るのもいいですし、先人の実践例など他の人が使われているものをパクってもいいです。それで済むならしめたものです。人によっては、いまいち納得がいかなくて、自分で模索したくなることがあります。その場合はとことん試行錯誤しましょう。模索していくうちに愛着が湧いてきます。この、自分でつくりあげていくときに湧いてくる愛着は、挫折しがちな PTTM のモチベーションを支えてくれます。

スペースは半角で統一する

Ans: スペースで空白を開ける場面は多いですが、半角スペースで統一したいです。

箇条書きでタスクを階層的に並べたり、空白を入れて見た目を微調整したり、とスペースを使う場面は多いです。このスペース、半角スペース「 」と全角スペース「　」がありますが、前者の半角スペースを常に使うことをおすすめします。全角スペースは百害あって一利なし――は言い過ぎですが、デジタルの世界では何かとトラブルになりがちなものの一つです。プレーンテキストを扱う際も例外ではありません。少なくとも検索時にスペースの可能性が二種類あると検索しづらいですし、エディタの機能でインデントをまとめて上げ下げする（以下例）際も全角だと行われない、など地味に不便を被ります。



indent

全角分のスペースが欲しいなら半角スペースを複数回使えばいいですし、小説など特殊な場合に例外的に入力したくなったらShiftキーを押しながらスペースキーを押すことで全角になります（Google日本語入力の場合）。

常に半角スペースを使うには設定が必要です。詳しくはIMEの設定画面を探してください。検索してもすぐに見つかると思います。ちなみにMacの標準のIMEではかんたんにはできないようです。

まとめ

- PTTM
  - プレーンテキストによるタスク管理
  - アナログで紙とペンで自由自在に書くように、デジタルでも自由自在に書いてタスク管理したい
  - そのためにはデジタルのお作法「プレーンテキスト」を踏まえる必要がある
- 使う道具
  - IME、エディタ、ファイル形式・記法、情報管理（≒ファイル管理）、ヘルパーの5つ
  - まずは標準のIME、テキストエディタ or IDE、.txt、単一ファイルがおすすめ
  - ヘルパーは最悪なくてもいいが、あればあるだけ便利になる
- PTTMの設計
  - 本章の原則や先人の事例を参考にして、自分なりに模索する
  - 模索という営みからは逃れられない
  - メンテナンスからも逃れられない
    - プレーンテキストの読み書きはずっと続く
    - 息するようにこなせるようになりたい
    - なれたら定着するし、なれないとおそらく挫折する

=== Chapter-24 文芸的タスク管理

＝＝＝

最近タスク管理において頭角を現しつつある概念が一つあります。あえて名付けるなら文芸的タスク管理とでも呼べるでしょう。文芸的――つまりは文章を書く営みをベースとしたタスク管理です。

本章では文芸的タスク管理なる概念を筆者なりに整理します。

文芸的タスク管理の概要

文芸的タスク管理

文芸的タスク管理（Literate Task Management, 以下 LTMと略します）とは、文芸的に行うタスク管理を指します。

文芸とは「文章を書く営み」と考えてください。ノートやメモといった言葉がよく似合うものです。芸術性は必要ありません。つまりLTMとは、文章を書く営みを通じたタスク管理を指します。日記でも日誌でも備忘録でも何でも良いですが、恒常的に**自分用の**文章を書くという前提があって、その中でタスクも扱うのです。もちろん、タスク管理にはタスクリストなどタスクをノイズ無しに集めた**ビュー（見せ方）**が必要ですし、前章で述べたようにデジタルで文章と付き合うならプレーンテキストの理解も要りますが、こういったものを上手く取り入れます。そうすることで文章を書く営みを大事にしつつタスク管理も行える、というバランスを手に入れることができます。

メリットは文脈の尊重

LTMのメリットは**文脈の尊重**です。文脈についてはコンテキストの章で詳しく取り上げましたが、文章を書くことで人生や生活や仕事といった文脈がクリアになります。これを読みつつ、書き足したり書き換えたりしつつ、タスクについて考えることで、より文脈を尊重したタスクを扱えます。

目的はQoLの向上

LTMの目的は**QoLの向上**です。自分の文脈を踏まえて行動することは、純粋に心地が良いことです。

必ずしも生産性や成長に繋がらないことには注意してください。生産性を増やしたり、より成長したりしたいなら、文脈なんて無視して必要なタスクを気合を入れて消化するのみです。そうではなく、自分の文脈を尊重することを第一にするのがLTMです。その結果、向上するのは何

かというと QoL に他なりません。ただし、文脈を尊重した方が（長い目で見れば）より早く上手くいくことも多いので、実は生産性や成長を求めている場面でも有効です。とはいえ、文章を読み書きする営みにはある程度の余裕が必要ですから、やはり目的としては QoL の向上、くらいに捉えておくのが良いでしょう。特に LTM は慌ただしい場面の打開には向いていません。

ひとりで行うもの、かつ人を選ぶ

LTM は個人タスク管理の手段です。つまりひとりで行います。プロジェクトタスク管理やパートナータスク管理など、複数人で行うタスク管理の手法としては想定していません。本章でもあくまで個人タスク管理としての LTM を扱います。

また、LTM には向き不向きがあります。少なくとも文章を読み書きする営みを継続的に行えることと、読み書きする余裕があること（たとえば 1 日 1 時間以上を恒常的に確保できるか）、そしてアナログではなくデジタルツールを使うことの三つが必要です。この三つができない人は向いていません。ただし言語化能力や読み書きのスピードは気にしなくてもいいでしょう。タスク管理は本質的に個人的なものです。自分なりに文芸ができればそれでいいのです。

文芸的●●のルーツ

文芸的、のニュアンスを知るために軽くルーツをたどります。

文芸的プログラミング

一番有名で、かつ起源ともなっているのが文芸的プログラミングでしょう。

元はドナルド・クヌースが提唱した概念ですが、発想としてはドキュメント内にコードを書くようなプログラミングスタイルです。プログラミングは通常、プログラムコードを専用で記述し、それをコンピュータが読み取って実行します。コードを読むのは大変ですし、設計思想や意図などを残しておかないと後で改造できませんので、別途解説としてドキュメントも書くのですが、これはコードとは別に自然言語で書きます。あるいはコード内のコメント構文で残します。いずれにせよ「コード」と「ドキュメント（自然言語で書かれた詳しい解説）」を結びつけるのはかんたんではありませんでした。特に両者は整合性も持たせねばなりませんが、コードを更新した後に解説も更新するのは二重管理的であり骨が折れます。文芸的プログラミングは、この件について新たな選択肢を提案しており、「いやいや、だったらドキュメントを中心にして、この中にコードも書いちゃえばいいのでは」と言っているわけです。

現在よく知られている文芸的プログラミングは Jupyter Notebook だと思います。これはデータ分析で使われるもので、ノートブックと呼ばれるファイルに作業手順やメモ、実験したいプログラムコード、その実行結果などをすべて同梱できるものです。作業手順やメモというドキュメントの中に、実験用のコードと実験結果も入っているわけですね。実験という限定的な用途ではありますが、文芸的プログラミング的と言えます。

文芸的タスク管理の可能性

文芸的タスク管理の概念がいつ生まれたかは定かではありませんが、以下 2 つが見つかりました。

一つは 2020/06 の文芸的タスクマネジメント（文芸的タスク管理）のススメ - ari's worldです。記事では『ふりかえりも、計画づくりも、タスクばらし、と実行を分けずにドキュメントに記載していきます』とあり、流れとしては、思いついたことを何でも書く → 適当なタイミングで計画立て → タスクにばらしつつ実行、となっています。文芸的プログラミングから着想は得ていますが、文章にタスクを埋め込むというよりも、文章ベースで何でも書いて膨らませてそこからタスクを取り出していくアプローチとなっています。これはどちらかと言えば次章で解説する探索的タスク管理に近いです。

次が 2021/12 のプロジェクトリストを使わないプロジェクトリスト管理と文芸的タスク管理 | R-styleです。まさに文芸的タスク管理と言える考え方だと筆者は思いました。記事では『「タスク」を処理するその前に、それを生み出している「文脈」を扱えるようにしておきたい』とあります。行動としては『Todo.mdには、まず文章を書き、そこからタスクを起こすようにしている』と文章を書く営みをベースにしつつ、『Dropbox直下のフォルダ/ファイルをすべて走査し、そこにTodo.txtないしTodo.mdファイルが含まれているならば、その内容を抜粋して表示するスクリプトを書いた』とビューをつくる工夫もしています。やはり文芸的プログラミングから着想を得ています。

ノートの必要性

LTM ではノートが必要不可欠です。

たとえば以下は LTM ではありません。

- X、LINE、Slack/Discord/Teams など短文の投稿とタイムライン表示から成る SNS やチャットシステム
- はてなブログなどオンラインのブログシステム
- アットウィキ、Pukiwiki、Mediawiki など従来の Wiki システム
- GitHub Issues など BTS
- Notion など多機能ノートツール

まず LTM を行うためには、文章をストレスフリーに書けなくてはなりません。誰かに見せるための文章を、重たい腰を上げて書くのでは成立しません。前章で書いた3秒の壁は言い過ぎにしても、息するように苦なく文章を書けるだけの軽さが必要です。**世に存在する大部分の情報共有・コミュニケーションシステムはここを満たしていません** し、多機能ノートツールも軽さの面で足りません。できれば前章のとおり「プレーンテキストを扱う」「オフラインのツール」が望ましいですが、後述するように一部オンラインツールでもできなくはないです。

次にまとまった文章を扱えなくてはなりません。一度に 1000 文字くらい書いたり、一日に

10000 文字書いたりすることもありますし、100 文字以内の短文、たとえば単語の羅列を一気に 100 個行うかもしれません。こういった場合でも問題なく動作してほしいですが、SNS やチャットシステムはこれを満たしません（あるいは満たせますが不便です）。ここを満たすには、複数行のテキストを扱えるポテンシャルを持つ「ノート」的なシステムが要ります。短文ベースの SNS やチャットでは足りません。

文章を書くと聞いて、上述したツールを思い浮かべた方も多いと思います。しかしそれでは LTM には足りないのです。十分な軽さを備え、かつノートのポテンシャルを持つツールが必要なのです。

LTM の準備

LTM を始めるために必要な手段や考え方を整理していきます。なお、LTM におけるタスク管理は次節で扱うこととし、本節ではそのための基礎（ノートの使い方や作り方）を解説します。

LTM を回せるツール

LTM にはノートが必要と書きましたが、では、どのようなノートツールがあればいいでしょうか。

3つのアプローチ

2024 年現在では、以下 3 つのアプローチがあると思います。

- 1: オフラインでプレーンテキストエディタ
- 2: ネットワーク型ノートツール
  - 例: Scrapbox、Obsidian、その他 Roam Research や Logseq など
  - オフラインだったりオンラインだったりします
  - 戦略の章でも第三世代的なノートツールとして取り上げています
- 3: アウトライナー
  - 例: WorkFlowy、Dynalist
  - ※厳密にはオフラインのアウトライナーもありますが本書では割愛します

1 のプレーンテキストエディタは前章にて解説したとおり、プレーンテキストをテキストエディタまたは IDE で扱います。オフラインで動作するため軽快ですが、ほぼテキストを編集するだけですので高級な機能が無く、欲しい機能は基本的に自分でつくるなり工夫するなりしなくてはなりません。DIY 的と言えるでしょう。その気になれば、2 や 3 のポテンシャルを持たせることもできなくはありません（※1）。

2 のネットワーク型ノートツールは近年台頭してきた手段です。最大の特徴はノートとノートをリンクでつなぐことです。たとえばページAにて `[[B]]` と書くと、A → B へのリンクが繋がれます。ペー

ジ B を見ると、メタ情報として「A から接続されています」と表示されたりもします（バックリンク）し、文章中のリンクはもちろんクリックしてジャンプできます。ウェブページはリンクを辿って別のページに辿れますが、それと同じことを手元の、個人のノートでも行えるようにする（そのリンクも自分で紡いでいく）のです。こうすることでノート全体がネットワーク的となり、リンクを辿って探索したりリンクしている・されている周辺の情報を表示したり、といった形で情報を探しやすくなります。

ツールとしてはオンラインとオフラインがあります。Scrapbox はオンラインです。Obsidian や Logseq はオフラインです。ただしオフラインのツールもオンラインの機能（特にクラウドへの保存やデータ同期）を持っているか、工夫して付与させることができます。

3 のアウトライナーは、いささか特殊ですが、すべての情報を箇条書きで表現します。階層的に分類したり、並び替えたり、指定階層以下を折りたたんだりなど、階層化した箇条書きを柔軟に扱うことができます。LTM として出来なくもないという選択肢にすぎず、ネットワークをつくる能力が弱いのが難点です。本章ではこれ以降扱いません。

- ※
  - 1 たとえば筆者は愛用の秀丸エディタをベースとして Hidemaru SCB や houtliner を自作しました。2024 年現在、houtliner は使っていませんが、Hidemaru SCB は愛用しています。一応公開はしていますが、自分用であり丁寧な解説もつくっていませんのでご容赦ください。あくまで参考として挙げただけです。

おすすめのツール

すでにお使いのツールがあれば、それでやってみるのが良いと思います。

そうではない場合のおすすめとしては、オンラインなら Scrapbox、オフラインならプレーンテキストエディタです。まずネットワーク型ノートツールとしては Scrapbox が圧倒的だと考えています。筆者のヘビーユーザーの一人ですが、他の追従を許しません。ただしオンラインゆえの制約もあります――文章を大量に書きたい場合や通信環境が強くない場合、あるいは仕事で使いたいがクラウドツールの許可を得づらい場合などです。このようなときは、オフラインの手段としてプレーンテキストエディタが使えます。

余談1: 無限キャンバス vs ネットワーク型ノート

ネットワーク型ノートの対抗馬として **無限キャンバス** が挙げられることがあります。これは Miro に代表されるようなホワイトボードツールのことで、多人数によるコラボレーション用途を想像しがちですが、この無限の二次元空間をひとりで使えば情報管理手段や仕事場としても優秀なのです。

無限キャンバスは、実は文芸的営みと似ています。単一のワークスペース内に「文脈」を――要は関係する情報を何でも全部入れてしまう点が同じなのです。従来のファイル管理やデータベースでは、特に 1000 ファイルなど分量が多いと破綻していましたが、この二つのやり方なら

秩序を保てます。ネットワーク型ノートでは言語（テキスト）を重視してネットワーク構造をつくることで、無限キャンバスではテキストに限らず多様なコンテンツを空間上に配置していくことで保ちます。筆者の知る限り、大量の文脈を管理できるパラダイムはこの二つだけだと思います。

さて、LTM は無限キャンバスで行えるでしょうか。また向いているでしょうか。

結論を言うと、ネットワーク型ノートよりは向いていないと考えられます。まずここまで述べたとおり、タスク管理は言語を扱うものであり、個人タスク管理は自分のタスクを扱います、つまりは自分で言語化したものを扱うためノートツールに分があります。無限キャンバスは、他者の情報を貼り付けたり、グラフィカルにレイアウトしたりといった用途では強くて、言語というよりは視覚的・空間的な営みになります。情報管理や意思決定、あるいは創造のための準備や作業場としては良いですが、タスク管理とは相性が良くありません。本書で強調してきたとおり、個人タスク管理では言語と向き合わねばなりません。

無限キャンバスでもできないことはないと思いますが、結局ノートに頼ることになります。Ink & Switch の Embark という研究では（旅行計画の文脈ですが）無限キャンバスを使っていても結局メモを書くし、それを見て行動すると考察されています。メモとは文脈に他なりません。結局言語で文脈を残す（しかない）のです。ならば、テキストの管理に強いノートツールに分があります。

余談2: ノートの欠点を補う

ノートのデメリットはビュー（見せ方）が弱いことです。LTM においても、ノートのどこかに書いたタスクをどうやって俯瞰するかは主要なトピックです。

肝心なのは**ビューが弱いからといってノートを使うのをやめない**ことです。文芸的な営みにはノートが最適です。でもノートはビューが弱い――だったらノートを諦めずビューを補えばいいだけです。2024 年現在では、まだ LTM におけるビューは不十分であるため（自作なり工夫なりできなければ）不満が出ます。それでもノートから離れないでください。**ノートから離れた時点では LTM は終わります**。特にビューを好む人が多く、無骨で言語的で情報密度の高いノートの世界は敬遠されがちですが、それでも（LTM をしたいのなら）ノートから離れてはいけないのです。

ビューの補完については、個人的には風向きが変わっていると感じます。特に生成 AI が大きいです。というのも、生成 AI はテキストを扱うのに長けており、現状でも要約をつくったり変換や翻訳を行ったりは可能だからです。そう遠くない将来、AI にノートを食わせて問いを投げかけるだけでタスクリストを教えてくれたり、もっと高度な、それこそ秘書のような柔軟な振る舞いをさせることも可能になるかもしれません。限定的にはすでに可能です（※1）し、タスク管理ではありませんが一部のノートツールや、それこそ前述の Miro でも AI 対話機能はすでにサポートしています。ChatGPT でもテキストの指示やデータから図表をつくることができます――というわけで、そう遠くないうちに、生成 AI を活用した、革新的なビューの補完方法も出てくるんじゃないか、それをタスク管理に応用したソリューションやツールも出てくるんじゃないかと筆者は楽観視

しています。

- ※
  - 1 課題を言えば、精度に難があることです。特にタスク管理として運用するとなると、絶対にタスクのヌケモレは避けたいわけですが、現状ではまだその域に達していません。たとえば一週間分のノートを与えて「今日以降存在するすべてのタスクを時系列的に並べて」と指示しても平気でヌケモレが生じます。生成 AI は確率的なアルゴリズムであり、もらった指示の続きとして最もありえそうな文面を学習データに基づいて生成しているだけです。特定の場所に書かれた特定の意味合いを持つ情報をすべて確実に抽出する、といった作業には構造的に向いていません（たとえ指示を工夫しても精度が多少上がるだけでそれなりに漏れます）。これでは怖くて使えません。もちろん、将来的にどうなるかはわかりませんし、筆者が未熟かつ無知なだけで、現状でもすでにタスク管理用途でも実用できるものを開発している例もあるかもしれません。

LTM のスタイル

ノートツールを用いて、どのように LTM を進めていけばいいのでしょうか。やり方に正解は無いのですが、よく見られるものを 3 つにまとめました。

- 1 ダイアリーベース
- 2 プロジェクトベース
- 3 ネットワークベース

先に紹介するほどかんたんですが、LTM としては弱いです。つまり LTM の理想は 3 のネットワークベースです。

では個別に見ていきましょう。

ダイアリーベース

ダイアリーベースは日単位でテキストを書くというシンプルなやり方です。まさに日記です。シンプルゆえに始めやすいですが、LTM としては機能しづらく、正直言って前章のPTTM（プレーンテキストによるタスク管理）を脱せません。所詮は日単位のテキスト群にすぎないからです。LTM としてタスクを認識したり、あとで洗い出して把握したり、あるいは単にタスクの文脈を読み直したりするためには、**自分が書いたテキストをひたすら読み返す** こととなります。職業プログラマーはコードを書く時間よりも自分や人が書いたコードを読む時間の方が多いと言いますし、作家も原稿を書く時間よりも推敲のために読み直して修正する時間の方が長いと聞きますが、同様にダイアリーベースの LTM 実践者でも、読む時間の方が長くなります。ですが自分が書いたテキストを読み返すのは退屈ですし、PTTM でも述べたとおり全面的な手作業でタスク管理を頑張る（特に日々メンテナンスする）のも大変です。ダイアリーベースでは LTM には足りないのです。

プロジェクトベース

次にプロジェクトベースですが、これはタスクをプロジェクトという単位で捉えて、1-プロジェクト 1-ノートで書くやり方です。ノートという単位はファイルでもページでも、あるいは■プロジェクト1のように区切ったエリアでも構いませんが、とにかくプロジェクトごとにエリアを分けて、そのプロジェクトに関する内容を全部そこに書き込むのです。たとえば引っ越しタスクが必要なら引っ越し.txt なり■引っ越しなりをつくって、そこに書き込むのです。こうすることで「プロジェクトAに関する情報はノートAにある」という風に対応付けられます。たとえるならノートという名の作業机をプロジェクトごとに用意します。デジタルなら空間的制約もありませんからいくらでもつくれます。その気になれば 10 はもちろん、100 でも 200 でもつくれます。

一見すると上手く行きそうですが、たいていの人にはまだ難しいでしょう。プロジェクトベースは本質的には **継続的な整理整頓をテキストベースで行え** に等しいです。つまり整理整頓が得意な人でないと成立しません。現実でも一つの机が散らかるように、おそらく未整理.txt みたいな場所に何でも書き込んで、テキストが増えすぎて破綻するでしょう。仮に物理的な整理整頓が得意な人でも、プレーンテキストをきちんと扱えるスキルや素養が必要ですから（これがないと操作の手間がいちいち大きくて面倒くさくなる → 形骸化する）上手くいくとは限りません。他にも「プロジェクト」という単位をどう捉えればいいのかとか、プロジェクトをまたいだ事項――たとえばプロジェクト A にも B にも絡むメモをどこに書けばいいのかといった問題も出てきます。まさか A にも B にもコピーするわけにはいきませんし、2 つならまだしも 3 つだとしたらどうでしょう。面倒くさすぎて形骸化します。

ネットワークベース

このあたりの問題点を比較的上手く解決するためには、新しいパラダイムが必要でした。それがネットワークベースです。これは一つの話題を一つの単位に書いて、かつ話題と話題をリンクで繋ぐことでネットワーク状の構造をつくるやり方です。プロジェクトベースでは 1-プロジェクト 1-ノートでしたが、ネットワークベースでは 1-話題 1-ノートです。話題（トピック）はタスクやタスクを表現するプロジェクトに限らず、何でも良くて、それこそ「炊き込みご飯ってめっちゃ美味いんだな.txt」をつくっても良いのです。これを筆者は **トピック指向** と呼んでいます。話題 A に関することをノート A に書くのです。話題の単位は大きくても小さくても構いませんし、タスクなどわかりやすい単位である必要さえありません。この性質上、ノートの数は数百では済まず、数千、場合によっては数万以上もありえます。もちろんデジタルだからこそできることです。筆者が思うに、このパラダイムを確立したのは Scrapbox だと思います。正直これを体感したければ Scrapbox を重用するのが最も確実です。

このトピック指向を実現するためには、特殊なツールが必要です。それがすでに紹介したネットワーク型ノートツールとなります。

ネットワークはプロジェクトとダイアリーを包含する

ネットワークベースの良い点は、プロジェクトベースとダイアリーベースも包含できることです。単にプロジェクト単位でノートをつくればプロジェクトベースになりますし、もちろん日単位でノートをつくればダイアリーベースにもできます。プロジェクトにせよ、ダイアリーにせよ、歴史があって私達にも馴染みやすいため使わない手はありません。むしろ積極的に使います。

ただ、ネットワークベースですので、それだけではないのです。たとえば 2024/08/05 の日記ノートに「炊き込みご飯ってめっちゃ美味いんだな」と書いて、これをリンク化すれば、「炊き込みご飯ってめっちゃ美味いんだな」ノートがつくられます。炊き込みご飯に関する気付きや思いを書き溜めることができるでしょう。ここからさらに「レシピサイト調べたいな」とか「実家は定食屋だし炊き込みご飯屋とかビジネスとしてアリなのでは」と話題が広がるかもしれません。これらもまたノート化して、さらに書き込んで――と、このようにリンクをつくってネットワークを広げていくことができます。

ただのダイアリーやプロジェクトでは到達できない高みがあります。人間の脳、交通網やインターネット、人間関係などネットワークは実は非常に馴染みのある構造であり、これを個人のノートにも取り入れることで圧倒的な書きやすさと読みやすさ（辿りやすさ）を実現できます。第二の脳と呼んでもいいくらいの、濃厚なノートをつくることが現実的に可能なのです。そして、ここまで時代が追いついたことで、ようやく LTM も可能となったのです。

□ LTM ≒ ネットワーク

ここまでを踏まえると、LTM の推奨ツールはネットワーク型エディタ、推奨スタイルはネットワークベース、となります。以降ではこの前提で進みます。

つまり以降では LTM としてネットワーク型エディタかつネットワークベースなスタイルを用いることを前提として解説します。

ホームをどうつくるか

通常私たちには自宅（住み込み等も含む）があり、生活の拠点になっています。仕事においてもオフィスの自席や作業場といった拠点があります。少し前にノマドワーカーという言葉が流行りましたし、今もリモートワークにより拠点を持たない生き方も可能となっていますが、そういった生き方でも短期的な拠点はつくるはずです。

LTM も同じで、まずは拠点をつくらねばなりません。これを **ホーム** と呼ばせてください。このホームのやり方にいくつかスタイルがあり、よく見られるものを紹介します。以下の 4 つです。

- 1 ホームノート
- 2 ホームビュー
- 動的なホーム
    - 3 日付ノート

    - 4 トランク

オフラインで手作業が多い場合は 1 のホームノート、3 の日付ノート、4 のトランクが良いでしょう。どれが合っているかは人によります。どれも運用難度は大差ないので、ひととおり試すのも良いでしょう。Scrapbox などオンラインで、かつビューも用意されている場合は 2 のホームビューが使えます。

続いて各詳細を見ていきましょう。

※ちなみにツールによってはノートのことを「ページ」と呼ぶことがあります。1つの情報単位を何と呼ぶかはツール次第です。Scrapbox ではページです。**本書では「ノート」を使います。**

ホームノート

唯一の玄関口に相当するノートを一つつくります。これを **ホームノート** と呼びます。

ホームノートは玄関であり、あなたが LTM として書いてきたノートのすべてはここから辿れるようになっています。何をどう書くかはすべてあなたが決めます。散らかった机のように長々と書き散らしても良いですし、案件Aに関する記述を「案件Aノート」をつくってそっちに移しつつホームノートからは案件Aノートへのリンクを一行だけ書く、でも良いでしょう。

構造的にはおそらく以下の 3 つが生まれるはずです。

- 1 ホームノート上で書き散らしているもの
- 2 ある程度まとめてはいるが、ホームノートに直接書いているもの
- 3 リンク集
    - ホームノートに書いてないものは別ノートに存在しますが、それだと辿れないのでホームノートからリンクを張ります。この性質上、ホームノートにはリンクが並んでリンク集になります

リンク集についてはホームノートに直接書く必要はありません。たとえば「もう見てない」という名前のノートをつくって、これをリンク集にすればいいでしょう。ホームノートからは「もう見てない」ノートに1行だけリンクがあり、「もう見てない」ノートには実際にもう見ていないノート達のリンクが何百と並んでいる――なんてこともありえます。この何百のリンクをホームノートに置くと邪魔なので、「もう見てない」という名前のノートを一段つくってそこに入れるわけです。

要は玄関口を自分で上手くメンテナンスしなさい、というやり方ですので、自分で整理整頓できるならこれで十分です。できない場合はビューに頼ったホームビューや、日単位で強制的に移動させられる日付ノートを使うと良いでしょう。

ホームビュー

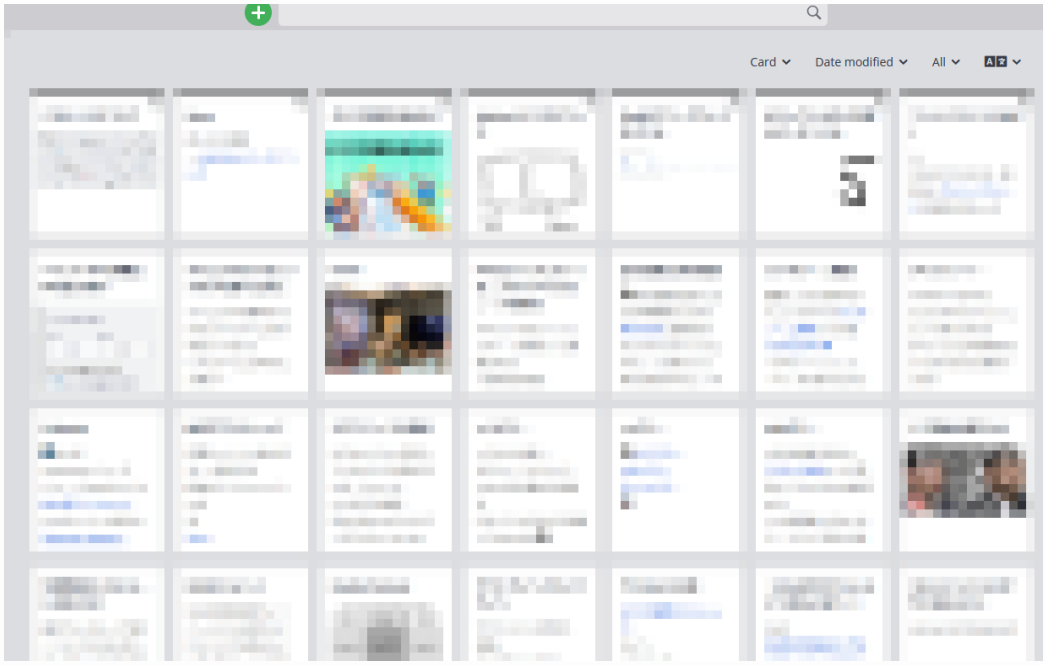
自分がつくったノート達を上手く見せるビューをホームにすること、あるいはしたものを **ホーム**

ビュー と呼びます。

ホームビューでは、おそらくノートをリストなりカードなりで一覧表示したり、検索ボックスを出したりソートやフィルタリング用の部品を置いたり、あるいは新しくノートをつくる手段が整っていたりするでしょう。ビューですので、ここで直接ノートを書くことはできません。かわりにノートの管理、特に俯瞰が行えます。

もちろん、このようなビューはただのノートツールではつくれないため、ビューを持つツールを使う必要があります（自分でプログラミングしてつくることもできますが本書では割愛）。それもただのビューではなく、ホームとして使えるだけのポテンシャルが必要です。事実上ビューとして相当優れている必要があります。

筆者がよく知るビューは Scrapbox です。以下は筆者が持つプロジェクトの例です（私的なコンテンツなのでぼかしています）。



これはプロジェクトというワークスペース内にページというノートをつくる構造ですが、このプロジェクトがビューとなっており、カード形式で一覧表示します。ここから読みたいカードを開いたり、＋ボタン（上部の緑色のボタン）で新しくカードをつくったり、検索やソートやフィルタリングもできます。よく使うカードはピン留めして最上部に固定しておけば開きやすいです――と、ホームビューを使いたいならこれくらいのビューが必要でしょう。

日付ノート

日単位で日付名のノートをつくり、今日は「今日の日付名のノート」をホームとするやり方を 日付ノート と呼ばせてください。

たとえば今日が 2024/08/06 なら「2024-08-06」という名前のノートがあって、そこがホームになります。明日は「2024-08-07」ノートがホームになるでしょう。戦略の章で述べたデイリーエリアの概念そのものです。

日付ノートは、ホテルでたとえるなら毎日部屋を変えるようなものです。部屋を移る手間はかかりますが、毎日新しい部屋を使うことができます。ホームノートでは散らかってしまうという場合は、日付ノートをホームにするのが良いでしょう。

ここで一つジレンマが起きます。よく使うリンク集や毎日高頻度にメンテナンスするテキスト（典型例はデイリータスクリスト）といったものはほぼ必ず登場しますが、これらは常に手元に置いておきたいです。日付ノートがホームだと、毎日移動させることになってしまいます。これは手間ですし、これらが長大になってくると日付ノート自体も長くなってしまい使いづらいです。じゃあこれらを、たとえば「倉庫」という名前のノートをつくってそっちに退避させて、日付ノートでは「倉庫」ノートへのリンクを一行だけ書くことにしようかと考えますが、これだと倉庫の中身を見る度に毎回リンク先を開かねばなりません。それは大変だからと、じゃあ日付ノートと「倉庫」ノートを両方とも（別ウィンドウや別タブの形で）常に表示させておけばいいかと考えますが、これはこれで日付ノートと「倉庫」ノートを行ったり来たりすることになって **タスクが漏れます**。タスクリストとしてちゃんと書いてたはずなのに、なぜかスルーしてしまった、ということが起こり得ます。二つ以上に分散しちゃうと、注意も分散してしまって忘迷怠が起こりやすくなるからです。なので **ホームとして常に表示し、常に見るノートは一つだけに留めたい** ところです（※1）。

このように毎日使うテキストを日付ノートで扱うのは意外と難しいのです。面倒くささを減らして、かつ忘迷怠も最小限にするバランスは各自で探ることになります。

実用的にはハイブリッドに使うのが良いと思います。たとえばホームノートとハイブリッドにする場合、ホームはホームノートとしておき、ホームノートには当月分の日付ノートへのリンク集をつくるのです。一番の上を今日の日付ノートへのリンクにすれば、今日の分は一番上のリンクからアクセスすればいいと単純化できます。また Scrapbox では今日の日付ページをピン留めすることで、ホームビューをホームとしつつも今日の日付ノートにもすぐにアクセスできる形を実現できます（ホームビューと日付ノートのハイブリッド）――つまり日付ノートを唯一のホームにするよりも、何か別のホームをつくった上で、そこから日付ノートにもアクセスしやすくしておくという形にするのです。その上で日単位でメリハリをつけたい部分は日付ノート側で扱うようにするのです。そういう意味では、上述したバランスとは「日付ノートに何を入れるか（どこまで入れるか）」のバランスとも言えます。

- ※
  - 1 ただしディスプレイが大きいとか複数のディスプレイがある等、表示領域が潤沢な場合は例外です。日付ノートも倉庫ノートもどちらも常に表示できます。

トランク

トランク（**Trunk**）とは直訳すると「幹」で、幹から多数の枝葉が伸びているイメージです。トラ

ンクとは「多数のリンクを伸ばしたノート」を指します。そういう意味ではホームノートやリンク集は最も太いトランクですし、日付ノートも日単位で色々リンクを張るという意味ではそこそこ太いトランクですが、何もこれらに限らなくてもいいのです。

仮に今日から LTM を始めるとして、よくわからないのでとりあえず「テスト」という名前のノートをつくったとします。ここで日々タスクリストを書いたり、メモを書いたり、思考や検討や調査の結果も書いたりして過ごしていくとしましょう。何もしないとテキストが肥大化するばかりですが、ネットワークですので別途ノートをつくってそこへのリンクを張る、という形で逃がしても良いわけです。たとえば引っ越しネタは「引っ越し」ノートに、転職ネタは「転職したい」ノートに、日付ノートは「2024-08-06」ノートや「2024-08-07」ノートに、といった形でどんどん逃がしていきます。もちろん逃がしただけだと後で辿れないので、「テスト」ノートからはリンクを張っています。このとき「テスト」ノートはトランク的と言えます。「テスト」ノートで過ごせば過ごすほど、多くのリンク（枝）が増えていくでしょう。

しばらくして転職が完了し、引っ越しもできたとします。生活が一新されたようなものですが、「テスト」ノートはそのままです。すでにここはごちゃごちゃしているでしょうから、思い切ってホームを変えましょう。仮に大阪から岡山に引っ越したとして、ノート名は何でもいいですが「岡山新生活」とでもしておきましょうか。「岡山新生活」ノートをつくり、「テスト」ノートから必要なテキストをコピペしてきます。以降はこの「岡山新生活」ノートがホームです。今後はここからリンクが生えていくでしょう。可能ならあとで辿れるように、「テスト」ノートに跡地の旨と「岡山新生活」へのリンクを書いておくと優しいですね。

さて、さらに月日が経ちまして、岡山での新生活も落ち着いてきました。長期休暇も取れそうです。旅行に行きたくなってきました。「2024 旅行」ノートをつくりましょう。旅行に関する検討は（「岡山新生活」ノートではなく）こちらでやることになるでしょう。つまり「2024 旅行」ノートもそれなりのトランクにはなると思われます。このようにホームに限らず、必要に応じてトランクをつくるのもアリです。

と、しばらく例を膨らませてみましたが、何が言いたいかというと、**トランクの使い方は定住的でなくても良い** ということです。遊牧民のようにホームを転々とするやり方でも構わないのです。ただし、後々辿りたくなったときに困ってしまうので、可能ならリンクをちゃんとつけて辿れるようにしておきましょう。立つ鳥跡を濁さずと言いますが、遊牧的にトランクを切り替える場合はリンクくらいは残しましょう。別の言い方をすると、別れるときには別れの挨拶や送迎会といった儀式をしますが、遊牧的な LTM においては「リンクをつける」ことが儀式に相当すると考えてください。

まとめ

LTM の準備と題して、以下の点を述べてきました。

- ネットワーク型エディタを使うこと
- ネットワークベースを使うこと
    - ダイアリーベースやプロジェクトベースも使えます

- ホームをつくること
  - ホームノート、日付ノート、トランクの 3 つ
  - ビューが備わっているなら、そこをホームとしたホームビューもアリ

ですが、ここまで述べてきたのはノートの話です。肝心のタスク管理の話はまだ出ていませんので、次節から詳しく見ていきます。

LTM におけるタスク管理

ここまでで LTM はノートを使うこと、ネットワーク的につくること、特にホームをつくって広げていくことを強調してきました。

いよいよタスク管理の部分に入っていきます。ネットワーク的なノートの世界において、どのようにタスクを読み書きしていけばいいのでしょうか――その考え方とやり方を詳しく見ていきます。

ホームをつくる

とにかくまずはホームをつくります。

迷うなら「ホーム」でも「デスクトップ」でも「てすと」でも何でも良いので、何か一つノートをつくってそこをホームにしてください。ホームは後で変えることもできますが、ホームなる概念が定着しないことには始まりません。

加えて、数秒以内に記入開始できる状態にしたいです。3秒の壁はタスクを追加する時の話ですが、ここでは「ノートに何か書き込む時」の話をしています。どこに書き込むかは複数考えられますが、もっともシンプルなのはホームです。つまりホームへの書き込みを数秒以内に開始できる状態にしたいです。この素早さを達成するには、基本的には「常に開いておく」か、少なくともボタンやショートカットキー一発で起動できることが必要でしょう。開始までの早さは本当に重要ですので、0.x 秒でも縮める努力をしてください。開始までが遅いと書く腰が上がりにくく、LTM の挫折に引きずり込まれてしまいます。X など SNS をチラ見している人は多いと思いますが、イメージとしてはそれと同じです。おそらく即座にチラ見できるようにしているはずです。それを「ホームに何か書き込む」ことに関しても行いたいのです。

タスクに関する情報は 3 種類ある

LTM において、タスクに関する情報が 3 種類ほど存在することを押さえてください。

- メモ
- タスク

    - タスクビュー
    - タスクノート
- 文脈

メモとは、あとで処理する事柄を素早く一時的に残しておくことです。メモの章でも述べたとおり、できるだけ早く書ける素早さが重要となります。もちろん、書けたとしても後で拾えないと意味がないので、あとできちんと処理できるよう工夫もします。

タスクとは、やることです。Must でも Should でも Want でも何でもいいので、「やる！」と決まったもの、あるいは決めたいものを指します。

タスクはビューで俯瞰できねばなりません。自動でタスクを拾ってビューにするか、自分の手作業でリスト化するなどしてビューにするか、やり方は色々あります。また、どこに書くかについても、GTD のネクストアクションのように「あらゆるタスクを一箇所に集める」やり方（※1）もあれば、ノートごとにタスクリストエリアをつくってそこに集める（つまり複数に分散する）やり方もあります。いずれにせよ、複数のタスクを、俯瞰するために集める必要があるわけです。これを **タスクビュー** と呼びます。

もう一つ、タスクはそれ自体が色々な情報を持ちます。属性を含め 3A の話はすでにしました。ただし LTM は文芸的な世界ですから、これら属性なり関係なり機能といったものを全部テキストで表現することになります。また、文脈を尊重して QoL を上げると書いたように、開始時間だとか締切だとか分類だとかやったやらなかったとか進捗だとかいった、そういった「管理」的なことはしたくないわけです。もっと伸び伸びしたいのです。そのためには――と考えたときに、前述したトピック指向が出てきます。タスク T があるとするなら、T というノートをつくってしまい、T については全部そこに書き込めばいいのです。ついでに、T と関係ありそうなノートがあればリンクも張ります。こうすることで、T というノートに（その中身 or リンクで繋がった接続という形で）T に関する情報、つまりは文脈が集まるのです。**というより、集めるのです**。と、少し長くなりましたが、この T のように、タスクそれ自体に関するノートを **タスクノート** と呼びます。

つまり **LTM ではタスクノートの形で各タスクの文脈をノート的・ネットワーク的につくりつつ、タスク管理もちゃんとやるためにタスクビューも適宜つくって俯瞰もする、という戦略を取る** のです。この両輪から攻める程度でタスク管理を行えるようにしたいのです。従来のようなザ・管理的なやり方はしません。それはただのタスク管理であって文芸的タスク管理ではないからです。

さて、最後に文脈ですが、これはメモでもタスクでもない、すべての情報を指します。早い話、ノートに書いたことはすべて文脈です。この中から色んな発想や情報やタスクや意思決定が生まれます。指示されたことに黙って従うわけでもなく、正解がなくて動けないわけでもなく、LTM として書いて貯めてきた文脈の力を借りて、自分なりに意思決定して前に進むのです。十分な文脈が集まっていれば、たいていは自然な結論が出ます（何もわからないのでとりあえずやってみよう等も含めて）。そして、LTM を行えば文脈は集まります。ノートにあれこれ書き込んでいくからです。第二の脳と書きましたが、LTM では頭の中で考えてタスクリストを書いて終わりではありません。考えたことを全部（でなくてもいいですが）ノートに書き込み、その書き込みをメン

テナンスしていくのです。ノートを育てるのです。**育てたノートは文脈の宝庫になります。そうでなくとも文脈を育てる育成場として機能します。**

- ※
    - 1 ここで GTD のネクストアクションを持ち出したのは、タスクを一箇所に集める例としてわかりやすいと思ったからです。GTD ではインボックスやプロジェクトといった形でワークフローに基づいて「気になること」をフィルターしていき、最終的に「具体的に行動可能な小さなアクション」が出てきます。これを集めたのがネクストアクションです。仮に 20 くらいプロジェクトがあって、1 プロジェクトあたり 5 個のアクションがあるとしたら、単純計算で 100 個のネクストアクションが集まります。実際は途中で別のフィルターに行ったり（たとえば「Someday（いつかやる）」や「Waiting（連絡待ち）」など）、収集ステップで大量に気になることをインプットしたりするので 100 より増えたり減ったりします。ともかく、ネクストアクションには次にやればいいことが集まっているわけで、理想的には、ネクストアクションから一つ選んで行動 → また一つ選んで行動、のような過ごし方になるはずです。ネクストアクションという場所一つだけを相手にすればいいのです。「一箇所に集める」は、このニュアンスです。

エリアとマーク

メモとタスクの二つは、後で処理しなければならないものです。メモは早く読み返さないと（何を書いているか）忘れてしまいますし、タスクは忘迷怠を起こさずにこなさなければなりません。

では、LTM においてメモやタスクを「後で処理する」ためには、どうすればいいでしょうか。

本質は至ってシンプルで、後で改めて読み返すしかありません。となれば問題となるのは読み返しのコストです。読み返すための労力をできるだけ減らせばいいのです。何も工夫しないなら極論すべてのノートをを一から読み返せばいいだけですが、現実的ではありません。LTM ではノート数は数百、数千、下手すると万も超えるのです。

読み返しの労力を減らす戦略も、二つしかありません。

- エリア
    - あとで読み返すエリアを決めておき、そこに書いたものを後で読み返す
- マーク
    - 読み返す部分にマークをつけておき、あとでまとめて検索・表示する
    - つまり「マークがついたものを表示する」ビューをつくる

本当はもう一つ、生成 AI に全部読ませてリマインドしてもらうやり方もあるのですが、余談で述べたとおり精度の面で実用的ではないため割愛します。

エリア

まずエリアですが、これは全面的に手作業で行うものです。特定のノートまたは特定のノートの特定の場所を「後で読み返す場所（エリア）」と定めて、そこに書き込んでおき、別のタイミングでそこを読み返すのです。ホームはエリアの一つですが、エリアというにはしばしば大きくなりすぎるため意味がありません。

エリアはもっと小さいものです。たとえばホームに「デイリータスクリスト」を書く領域をつくって、その日やるタスクを全部突っ込んで、かつ高頻度に読み返すようにします。これはデイリータスクリストというエリアをつくっていると言えます。もちろん、やり方はこれだけではなく、「デイリータスクリスト」ノートを別途つくってもいいですし、「プロジェクトA」ノートと「プロジェクトB」ノートをつくってそれぞれにデイリータスクリストエリアをつくってもいいかもしれません。

重要なのは書き込みのしやすさ、そして **後でちゃんと読み返せるかどうか** です。ホームであれば毎日高頻度に目にするはずですから、何も工夫せずとも読み返せるでしょう。その分、慣れすぎてしまって「読んだつもりなのに普通にスルーしてしまう」恐れもあります。別にノートをつくって丸ごとエリアにすれば、慣れすぎは防げますが、今度はアクセスがしづらくなります。このあたりのバランスは動線の章でも述べました（ネグレクテッド）。また、カレンダーやリマインドを使って「このノートのこのエリアをチェックしろ！」と自分に教えるよう仕込むこともできます。GTD のレビューのように定期的なイベントにしてもいいでしょう、たとえば「読み返すエリアの一覧」をつくっておいて、週次レビューで全部読み返す、などです。

エリアは手作業的で、非常に面倒くさく思えますが、LTM では意外と機能します。そもそも LTM では恒常的にノートに書き込んだり、リンクを張ったり、新しいノートをつくったり、といったことをしているので土地勘や地理感のように定着してきます。目的の場所にも高々数ホップ（リンクを1つ辿ることをホップといいます）で到達できますし、どのノートからどのリンクを辿ればいいかも定着しているゆえに「わかる」ので迷いません。そもそも定着するまでが長い可能性はありますが、意外と手間は少ないのです。

マーク

タグがついたものを一覧表示する機能はお馴染みだと思いますが、マークとはまさにこれのことです。後で読み返したい文章（特に特定の行）に何らかの印（マーク）をつけておいて、あとでその印を検索して「印がついている部分」だけを抽出します。

手段としては、最もかんたんなのはノートツールが持つ検索機能を使うことです。エディタの場合は「全文検索」とか「Grep」と呼ばれていることもあります。これら検索機能は特定のキーワードにマッチする部分を表示します。つまりキーワードとしてマークを指定することで、マークのついた部分を表示できます。

さて、残る問題はマークをどうするかです。昔は `@task` など記号（この例ではアットマーク）つきの文字を書いたり、●task や ■task のように記号を付けるのが主流でしたが、現代では絵文字も使えます。絵文字は視覚的に目につきやすいのがメリットで、検索による一覧表示というよりは普段ノートを読み書きしているときに目に入りやすいのが便利です。タスクを表す絵文字

というと難しいですが、自分が分かればいいので□（ノート）、□（チェック）、□（ランナー）、□（ピン）、□（バクダン）など何でも構いません。可能なら終了を示す□は別に決めておいて、未終了を示すマークが欲しいです。□タスク1が完了したときは□□未タスク1のように重ねて書けば完了したことはわかります。

マークによる検索結果はビューと呼ぶことができます。タスクにのみマークをつけることを徹底しているのなら、検索結果は事実上タスクビューとなります。しかし、ビューを見るためにはいちいち検索を行わねばならず、この手間が命取りになります。手間がかかると怠けてしまい、怠惰が続くと形骸化まっしぐらです。したがって **検索結果を開く手間はできるだけ抑えてください。** ショートカット的な機能があるならそれで一発で呼び出せるようにしたり、ウェブアプリ型のノートツールであれば検索結果のページをブックマークしておいたり、もちろん自分で検索処理や検索結果を表示するスクリプトなどをつくっても良いでしょう。エディタを使っている場合、エディタの拡張機能を駆使することでも工夫できそうです。とにかく、マークを使う場合はビューをいかに手間なくつくれるかが重要ですので、この点は妥協しないでください。

さらに高度なやり方として、マーク以外にもタスクの属性をつけておくことが挙げられます。たとえばhownでは`[2002-10-22]@ タスク`という形でタスクを書きます（@の他にも!や+や-を使い分けます）が、これは日付系属性をつけています。そして、あとでビューをつくるときにこの日付と現在日を比べて、より近いものを一覧の上位に出すとか、逆に下位に沈ませるといった工夫をします。詳細は割愛しますが、現在日ベースで優先順位を自然に制御できるのです。もちろん、この書き方を読み込んでちゃんと処理するようhownがつくられてるから動作するわけで、自分でこれと同じことがしたいのなら自分でつくるしかありません。hownは文芸的タスク管理の上手い例と言えるでしょう。

LTM でカバーできない部分の対処

LTM は文芸的な営みであり、目的も QoL の向上です。要は文章を読み書きして自己満足感や自己肯定感を高めましょう、ついでにタスク管理もある程度行えるようにしましょう、というだけにすぎません。タスク管理としてあらゆることを行えるだけのポテンシャルなどないのです。LTM ではカバーしきれない部分がどうしても出てきます。これを **アンカバード（Uncovered）** と呼びます。

アンカバードの例と対処を以下にまとめます。

- ルーチンタスクとオルタスク
  - 詳細はプレーンテキストの章を参照
- タスクや文脈を他者に見せること
  - これもプレーンテキストの章で書きました
  - LTM は自分のみが読み書きすることを想定しており、そのままでは見せられないはずです
  - 他人にも見える場所での LTM もできなくはないですが、人目があると伸び伸びと書けなくて破綻しやすいため推奨しません

- 属性に基づいた細かい管理や高機能なビュー
  - 通常のタスク管理ツールを使ってください
- 締切を厳格に守らねばならない、厳しめのタスク管理
  - 通常のタスク管理ツールを使ってください
  - 必要ならカレンダーやリマインダーに頼ったり、イベント（強制力の働く場）への参加や設定も検討してください → オルタスクとしてで扱っています

とはいえ、アンカバードのやり方ばかりに頼っていては LTM などできません。なるべく LTM で完結できるように工夫して、どうしても面倒くさい部分だけは諦めて別のやり方でやるくらいが良いでしょう。

料理や DIY と似たようなものです。世の中にすでに良いものがあるから自分でつくるのはやめよう、としてしまっては何もできません。そんなことは百も承知で、それでも QoL を上げるためにあえて自分でつくってみるのです。実際楽しかったりもします。LTM も同じで、タスク管理ツールで行えることを、あえて文芸的に、自分で、ノートを使ってやってみるのです。とはいえ何でもかんでも再現していては大変ですから、調味料は市販のものを使うとか、ドリルは電動のものを使おうなど面倒なところは端折ります。LTM も同じで、アンカバードは面倒くさいので既存の手段に頼るのがベターです。

文脈の捉え方とつくりかた

捉え方

LTM にはメモとタスクと文脈があって、メモとタスク以外はすべて文脈だと書きました。ノートに書いたことはすべて文脈になるとも書きました。そもそも文脈とは何でしょうか。

LTM の目的は QoL の向上と書きましたが、タスク管理ではあるのでタスクを扱うことで向上させます。どのように扱うかというと、文芸的に行います――つまりは文章を読み書きします。通常、タスク管理は「やらないといけないもの」が降ってきて四の五の言わずに対応するしかない、ただリソースにも限りがあるから優先順位とタイミングは考えないといけない、といったニュアンスを持ちますが、LTM はそうではなく、日頃扱っている文章からタスクを決めます。**自分主導で行うタスク管理** であり、ヒントとして自分が書いてきた文章を使うのです。

文脈とはヒントです。

それゆえ正解はありません。仮に航海日誌のように淡々と事実と出来事だけを書くとしたら、それは客観的記録を文脈にしていると言えますし、感情をだらだら長々と書くとしたら主観を文脈にしていると言えます。書籍や SNS から有益な概念や金言を集めて考察するのもアリですし、誰にも通じないような深い思索や哲学におぼれてもいいでしょう。とにかく、書いたことは何でも文脈になりますし、LTM ではそれら文脈からタスクを導きます。

タスクを導くという点もポイントです。書いて終わりではなく、タスクという行動を伴う必要がありま

す。なぜなら、行動しなければ何も変わらないからです。行動の起こし方には色々ありますが、LTM ではノートに書いて考え、書いたそれらを育てるやり方を選びます。書いて、読み返して、考えて、反映して、を繰り返して積み重ねていくことで納得の行く結論や意思決定を出すのです。正しい、正しくないと言っているのではなく、あくまで LTM ではそうしますという話です。

小難しく書きましたが、まとめると以下のとおりです。

- LTM では、
  - ノートに書いたこと（文脈）をベースに考えを広げていく
  - 行動を伴わせたいので、どこかで行動――すなわちタスクを導く必要はある
- 何を書くか、つまり文脈は何かという問いに正解はない

つくりかた

では文脈はどのようにつくっていけばいいのでしょうか。言い換えると何を書いていけばいいのでしょうか。

ポイントは三つあると思います。

一つは自分が書きたいことを書くことです。LTM は今日も、明日も、ずっと読み書きを続けていく営みですから、そもそも読み書きが続かないと話になりません。嫌いなことやどうでもいいことはそもそも続きません。書きたいと思えるようなことを積極的に取り入れるのがまずは大事です。日記でもいいですし、趣味の話を全開にしても構いません。物申したいことや愚痴があるならそれを書いてもいいでしょう。とにかく、書くことが続きそうな題材を遠慮なく選んでください。

二つ目は「深堀りできるネタ」を捉えることです。最低でも現時点で見えているタスク――直近やらないといけないことから、そのうちやりたいことやできればやっておきたいこと、あるいはやりたくないけどやるべきだと思っていることまで、タスクになりうるものなら何でもです。趣味で深堀りしたいものがあればそれを含めても構いません。深堀りできれば何でも良いです。振り返りの章で取り上げたトリガーリストも便利です（深堀りできそうなネタを貯めておく）し、SNS やニュースや書籍を日頃からキャッチアップするのもよく使われる手です。発信者の方はご存知だと思いますが、意識としては「ネタ探しのアンテナを張っておく」感じです。ビビっと来たこと、特に色々深堀りしたそうなことを見つけたら、ぜひともメモして貯めたいです。あるいは別の言い方をするなら、業務中でも雑談をすることがあると思いますし、雑談であっても話を深堀りすることがあると思います。そのおかげで仕事に繋がることもよくあると思います。LTM も似たようなもので、要は自分に直接関係するタスクのみを扱うのではなく、**扱いたいものは何でも扱ってみましょう** ということです。そうすることで LTM の書き方にも慣れていきますし、自分の向き不向きや興味好奇や調子動機も見えてきます。こういうネタなら深堀りできそう、しやすそうという「美味しいネタ」が見えてきます。人は自分なりの軸がないと自律的に動けませんが、これら美味しいネタがまさに軸になるのです。

もう一つ、頭の中の棚卸しも適宜行うと良いでしょう。GTDでは収集ステップと呼ばれていますが、1〜2 時間などたっぷり時間を確保した上で、思いついたことを何でも書くのです。書いたも

のを見ていると、さらに色々思いつくのでそれも書きます。そのうち関連や本質が見えてくるので、整理もしていきます。こうすることで色々見えてきます。脳内で考えるだけでは堂々巡りにしかなりませんが、書いたものをベースにしていくことで（疲れはしますが）進んでいけるのです。ブレスト（ブレインストーミング）、フリーライティング、KJ法など、この手の営みは多数存在しますが、手法は重要ではありません。たっぷり時間を確保して、書いたことをベースに考えを広げて・深めていければそれでいいのです。自分のための営みですから正解もありません。自分なりに存分に楽しめればそれでいいです。LTM の場合、これを行う用のノートをつくって書き殴るだけで済みます。ノートの名前は「ブレスト1」でも「2024/08/11」でも「2024/08/11 07:25:47」でも「8/11 フリーライティングするぜ」でも何でもいいです。何ならノートさえつくらずに、ホーム上で始めてもいいです。とにかく、LTM であれば、書くことは気軽にできるのですから、さっさと始めてしまえばいいのです。このような営みを探索と呼びます。深堀りできるネタを、脳内に潜って探しに行くイメージです。

三つ目は自問自答です。一つ目や二つ目により、書くことは集まりますが、そこにどんな風に自分なりに書き足していくかがまだ足りません。自分で書かないと、ただの収集や陳列で終わってしまいます。自分の言葉で書くためには、自問自答が最適です。たとえば以下のような問いをします。

- 自分はどう思うか
- 行動に繋げるとしたら何ができそうか
- （嫉妬や羨望が湧き上がったとして）なぜそう感じてしまうのか
- 建前としてはどうするべきか、一方で本心ではどう考えているか
- 自分なりに面白いと感じる部分はどこか
- 自分が続きをやるとしたら、どうやるか etc

最初はどんな問いを選べばいいか混乱すると思いますが、頑張って続けてみてください。最初のうちはトリガーリストをつくって、それを見ながらでもいいでしょう。次第に「自分が好む問い」が見えてきて、息するように書き足していけるようになります。たとえば筆者の場合、「凄えと言われそうな上手いことを言いたい」という物申したさ、「新しい仕事術や概念をつくりたい」というものづくりの欲求、あとは自分のためになる学びや知識を吸収するために「自分の言葉で表現したい」とする気持ちなどが強いです。問いの形で表現するとしたら、「皆に凄えと思われる見解やツッコミは何か言えないか？」「ここからどんな新しい概念を導けるだろう？」「これを理解するために自分だったらどう表現する？」となるでしょうか。もちろん、これは私の好みであって、これが正解というわけではありません。

ライティングスタック

私たちは通常、頭の中に原本があります。頭の中の原本を言語化して人に伝えたり何かを書いたりします。これを筆者は **イメージスタック（Image Stack）** と呼んでいます。頭の中のイメージを積み重ねていく感じです。

一方、LTM では異なる考え方に則ります。ノートに書いたことを原本にするのです。頭の中

は、ノートに書いたことを読み込んであれこれ考えたりするための作業場にすぎません。原本はあくまでもノートにあります。ということは、日常的にノートを読み書きして育てていくことになります。仮に自分の人生哲学が 3 つあるとするなら、それは頭の中に留めるのではなく、ノートとして書いておくべきです。そして哲学が変わったのなら、その旨もノートに反映します。ノートが原本です。これを **ライティングスタック（Writing Stack）** と呼んでいます。

LTM が定着するかどうかは、このライティングスタックの考え方に染まれるかどうかにかかっていると筆者は思います。もちろん、人間なのでイメージスタックを完全に脱することはできませんが、それでも（特に LTM として扱う事項については）ライティングスタックに寄せたいです。これくらい考え方を変えないと、LTM という読み書きの営みは続きません。作家にとっては書いた文章が原本ですし、プログラマーにとっては書いたコードが原本ですが、同様に LTM を行う者にとって書いたノートが原本です。このライティングスタックの考え方を採用するからこそ、書くことができるのです。

特にタスクについては、なるべくライティングスタックを採用してください。タスク管理の主な恩恵は「脳内ではなくタスク管理ツール側に保存しているから脳内がスッキリする」ですが、LTM も例外ではありません。LTM を正しく運用できていると、然るべきノートを然るべきタイミングで見ることで思い出せる・切り替えられるようになり、頭の中にあれこれタスクを保持しておく必要がなくなります。その分、ノートの読み書きや思考に集中できます。慌ただしいシチュエーションには向いていませんが、このようなあり方は自律的で創造的な生き方であり、QoL を高めます。

切り出し

有益と思われる文章や単語は積極的に切り出してください。**切り出し** とは、指定した文章（単語でも良い）をノート名にして別ノートとして独立させることです。

たとえば、ちょっと前節の文章を引用してみますが、

> LTM が定着するかどうかは、このライティングスタックの考え方に染まれるかどうかにかかっていると筆者は思います。もちろん、人間なのでイメージスタックを完全に脱することはできませんが、それでも（特に LTM として扱う事項については）ライティングスタックに寄せたいです。これくらい考え方を変えないと、LTM という読み書きの営みは続きません。作家にとっては書いた文章が原本ですし、プログラマーにとっては書いたコードが原本ですが、同様に LTM を行う者にとって書いたノートが原本です。このライティングスタックの考え方を採用するからこそ、書くことができるのです。

ライティングスタックの概念を勉強し、定着させたいなら、何かしら切り出しを行うのが良いです。切り出し方は色々ありますが、筆者なら「LTM を行う者にとって書いたノートが原本」を選びます。これはつまり「LTM を行う者にとって書いたノートが原本」という名前のノートを新たにつくるということです。

実際の動き方としては、以下のように該当部分をリンクにします。Scrapbox だと`[ ]`、

Obsidian だと[[ ]]で囲むことで行えます。他のノートツールを使っている場合はその記法に従ってください。

> ……（中略）……プログラマーにとっては書いたコードが原本ですが、同様に[LTM を行う者にとって書いたノートが原本]です。このライティングスタックの考え方を採用するからこそ、書くことができるのです。

これでリンク化されるので、リンクをクリックして「LTM を行う者にとって書いたノートが原本」ノートを開きます。自分なりの学びを書いたり、リンク元ノートへのリンクを記載したりなどすれば良いでしょう。

こうすることで、以後「LTM を行う者にとって書いたノートが原本」という名前のノートが使えるようになります。部品化とも言えます。たとえば別のノートで LTM の勉強をしているときに「あ、これノートの方を原本にするってことだよな」と気づいたら、この「LTM を行う者にとって書いたノートが原本」ノートへのリンクを書き足すことができます。「LTM を行う者にとって書いたノートが原本」という知識が使われたのです。これを続けていくと、「LTM を行う者にとって書いたノートが原本」ノート側にも「どのページから使われたのか」が貯まっていきます。使われる機会が多いほど自分にとって重要そうだとわかりますし、自発的に再度読み返すことで新たな発見があるかもしれません。

作家は良いと思ったことわざや格言を貯めておいて、文中で使うと思います。プログラマーも、便利なスニペット（コード片）は使い回しますし、関数やモジュールやライブラリといった形で部品化もします。同様に、**LTM では文章を何でも部品化できます**。ノートにはノート名（タイトル）と本文があるので、詳細は本文に書いておいて、本質だけをノート名に含めればいいのです。この部品化のやり方は色々ありますが、最も頻出するのが切り出しなのです。

この切り出しは、タスクを扱う場面でもよく使います。わからない用語があったときにその用語名で切り出して、辞書や検索などで勉強したことを書き足しておくとか、Aさんのチャットメッセージを引用した上で、ここについては私はこう思うんだけどなぁということで該当部分を切り出してから自分の考えを書いてみるなど。ひと手間はかかりますが、切り出してちゃんと（あるいは雑でも良い）扱ってみることで文脈が強化されます。もっと言えば納得感を導きやすくなります。少なくとも自分はこう考えたのだと胸を張れますし、ノートとして残しておけば後々目にして再利用なり触発なりも起きるかもしれません。

無論、すべてについていちいち切り出すわけにはいきませんが、少しずつでも積み重ねていくことで、後々変わってきます。勉強も練習も地道に少しずつ重ねていきますが、LTM でも同様です。大げさに言えば、切り出しが文脈をつくります。切り出したノートは、言わばあなたの選択であり、あなたの文脈です。あなたなりの文脈を、切り出しによって日々積み重ねていくのです。

微修正とエイリアス

ライティングスタックや切り出しの話をしてきました。文章なら何でも部品化することができ、部品

を使う（リンクを張る）ことでノート間の繋がりが出来てきてネットワークが育ちます。ノートという原本が育っていくのです。まさに第二の脳です。

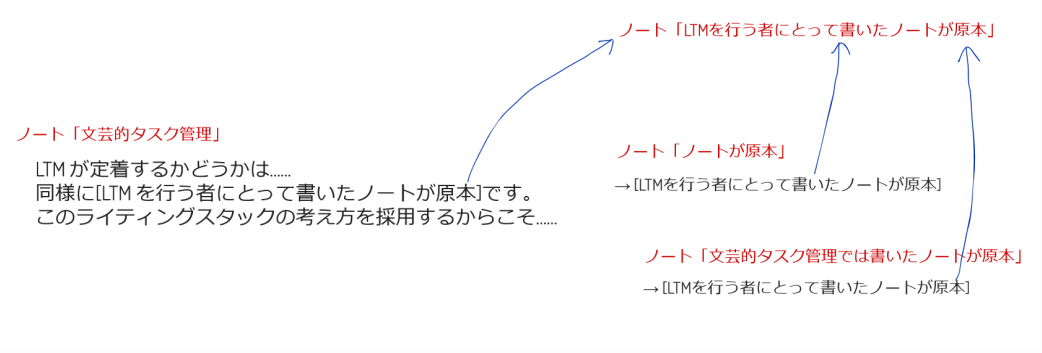
さて、このときに重要な原則が一つあります。ノート名は少しでもわかりやすく、あるいはキャッチーにするということです。リンクはノート名がベースとなっていますから、ノート名をどう書くか次第で扱いやすさ（見やすさや言及のしやすさ）が変わってきます。扱いやすさが低いと再読解や再利用の機会も失われて、忘れてしまいますので、明示的に確保していきたいのです。

まず自分にとってわかりやすい名前にします。前節では「LTM を行う者にとって書いたノートが原本」ノートをつくりましたが、この表現が気に入らなければ修正するべきです。以下にいくつか例を挙げます。

- 修正前
  - 「LTM を行う者にとって書いたノートが原本」
- 修正後の例
  - 「LTM を行う者にとっては、書いたノートが原本」
  - 「LTM では書いたノートが原本」
  - 「文芸的タスク管理では書いたノートが原本」
  - 「書いたノートが原本」
  - 「ノートが原本」

正解はないですし、原意の尊重や厳密性が重要とも限りません。自分にとってわかりやすいかどうかがすべてです。もちろん原意を尊重する方がやりやすいならそれでも構いません。

次にエイリアス（別名）も便利なので積極的に使いたいです。たとえば「LTM を行う者にとって書いたノートが原本」という名前を尊重しつつも、自分としては「ノートが原本」と「文芸的タスク管理では書いたノートが原本」が使いやすいなぁと思ったら、全部つくればいいのです。



こうすれば「ノートが原本」ノートからでも、あるいは「文芸的タスク管理では書いたノートが原本」ノートからでも「LTM を行う者にとって書いたノートが原本」ノートにたどり着けます。これはあくまでも一例で、「LTM を行う者にとって書いたノートが原本」にリンクを集めるようにしていますが、もちろん別のノートに集めても構いません。

エイリアスを使う場合は、1-ソース n-エイリアスを心がけてください。エイリアスを集める先（ソー

ス）はただ一つであるべきです。また、無闇にエイリアスを増やしてもあとで混乱するだけですので、「どうしても二つとも残したい」とか「なぜか知らないけど二択で迷う」など、自然とｎパターンに分岐してしまう（n=2 が多い）ときに限ってエイリアスを使うくらいがバランスが良いです。

アイキャッチとエキサイトメント

LTM を続けていくとノートがたくさん貯まります。リンクもたくさん集まります。そして、それらはテキストとしてずらりと並んでいます。いわば活字の海となっています。慣れた者であっても、これらを読み返すのは中々にハードです。そもそもすべてのノートを読み返したりメンテナンスしたりすることは非現実的でしょう。

ノートと継続的に向き合うためのモチベーション、なるものを何とかして保つ必要があります。ここで使えるのがアイキャッチとエキサイトメントです。

アイキャッチ とは、ブログ記事に差し込まれる画像やサムネ――動画のサムネイルなどが有名ですが、目を引く要素を指します。LTM においてはキャッチーなノート名、画像、GIF、動画などが該当します。要は活字の海に視覚的なアクセントを持たせることで、ノート全体を賑わせます。扱うコンテンツはタスク管理に関係しなくても構いません。筆者の場合、SNS でバズったポストの画像や VTuber の動画、あるいは自分で撮影したゲーム動画やダンス動画などを掲載しています。一見するとタスク管理には関係ないですが、これらのおかげでテンションが上がって、ノートを読み返すモチベーションも増えるため総合的にはプラスになります。特に自分のコンテンツを、まるで自慢するかのように展示することは自己肯定感を高めるメリットもあります（人を選びます）。

アイキャッチを行えるノートツールは非常に限られています。オフラインでエディタを使っている場合は厳しいでしょう。本章では Scrapbox を推していますが、動画像の挿入やサムネイル表示に対応した Saas 型のツールを事実上使うことになります――と書くと、動画像といった非言語要素が重要に思えますが、実はそれだけではありません。キャッチーなノート名をつけることも意外と重要です。LTM はノートであり、ノートであるからにはテキストは避けられませんので、テキスト自体でアイキャッチを確保することもサボらない方が良いでしょう。SNS やニュースでもキャッチーな見出しはよく見られますが、あのイメージです。自分にとって自然な表現や、自分を鼓舞する表現を追求していきましょう。ただし、他人の言葉を展示するだけだと「自分の」文脈が育まれないため要注意です。他人のコンテンツを保存するばかりですと、LTM というより、ただのブックマーク展示場となってしまいます。

次に **エキサイトメント** ですが、これは作業興奮を引き起こしやすくする構成にすることです。より具体的に言えば、書きかけのノートを優先的に表示することです。これも Scrapbox がお手本で、Date Modified という「更新した順にページ（ノート）を並べる」機能があったり、特定のページをピン留めして常に最上位に表示したりできます。こういった機能を使えば、やりかけのノート――いわば未完成の、中途半端なノートが先に表示されるので、手を付けやすくなるのです。Scrapbox のようなビューを持たない手段を使っている場合は、ビューを自分でつくってください。たとえばリンクを並べて自分なりの即席リンク集をつくってください。面倒ではありますが、

この一手間を加えていかないとエキサイトメントは確保できません。

エキサイトメントは広く確保すれば良いというものでもありません。やりかけのノートがたくさん並んでいると、それはそれで判断に迷ってしまって消耗します。おすすめはピン留めのイメージで常に固定しておくノート（静的なエキサイトメント）と、Date Modified のように動的に並び替えられるビュー（動的なエキサイトメント）を使い分けることです。前者、静的なエキサイトメントは、自分のキャパシティを守りたいです。筆者は 3〜7 個以内に抑えています。後者、動的なエキサイトメントは定期的に、または不定期に眺めて、目についたノートに手を加える形で雑に向き合います。筆者は週に一度の振り返りを除けば、基本的に不定期に適当に眺めて、目についたノートをいじくっています。筆者にとっては、これくらいが最適なバランスと思われます。これ以上個数を広げたり、眺める頻度を増やしたりしても、かえって苦痛になります。**ノートの読み書きには正解がなく、正解に向けて進めばいいという単純な世界には落とし込めません**。正解がない分、毎回大量の判断や思考を必要とするので、思っている以上に疲れますし、判断に迷うと書いた通り、実際はその疲労を警戒して「そもそも手がつかなくなる」「なぜか腰が上がらない」ことがよく起きます。これを防ぐためには、静的と動的の両面に対して、自分に合った広さを想定することが肝心なのです。サステナビリティと言うと大げさですが、末永く続けるつもりで、無理のない範囲でやりましょう。

おわりに

ネットワーク的なノートの世界において、どのようにタスクを読み書きしていけばいいかを紹介しました。

本説で強調したのは、タスク管理そのもののやり方や考え方というよりも、ノートのつくりかた——特に文脈をどう紡いでいくかという点です。「タスク管理じゃなくてノートテイキング（ノート取り）の話じゃないか」と思われたかもしれませんが、そのとおりです。**文芸的タスク管理は一応タスク管理といいつつも、本質はノートテイキングです**。ノートをつくることで文脈が固まり、文脈が固まればタスクの自ずも固まってきます。さらに言えば、固まったタスクは、LTM ではなく通常のタスク管理ツールで管理した方が上手くいくことが多いです。

これを私は **見えてきたらさっさと行動する** と捉えています。タスク管理にはタスクを決める段階と、決めたタスクをひたすら消化する段階がありますが、LTM が扱っているのは前者なのです。前者を、QoL を高めながら楽しくこなすために、自分の自分による自分のためだけのノートをつくるのです。冒頭でも LTM のメリットは文脈の尊重、目的は QoL の向上と書きましたが、後者の消化を重視する人には LTM は向いていません。

しかし、完全に向いてないかというと、そうでもなくて、文脈を尊重するからこそ消化も上手くしていけることもあります。スタンスの章でもコンテキスト（文脈）を重視するコンテキストンを取り上げました。とはいえ、慌ただしい状況下で瞬発的に次々とこなさないといけない場面ではさすがに役に立ちません。

LTM の意義は、前者の「タスクを決める段階」にフォーカスした手法であることです。元々タス

ク管理と呼ばれるものは後者、やると決まったタスクをいかに消化するかにフォーカスするものばかりでした。しかし現代は我々の目も肥え、水準も上がり、自律的に生きやすくなりました。こういう時代では何をどうやるかを自分で決めることもできます。いきなり決めるのは難しいですが、継続的に学びと考えを続けることで見えてきます。頭の中だけで続けるのは難しいのでノートに書きます。従来の手法でこれを行うのは困難でしたが、ネットワーク型ノートツールが出てきたことで可能になったのです。

まとめ

概要と考え方:

- LTM（文芸的タスク管理）とは、文章を読み書きする営みをベースとしながらタスク管理も行うというもの
  - メモ、タスク、文脈を扱う
    - メモは一時的に素早く書いておき、あとでちゃんと処理するもの
    - タスクはタスクの個別詳細を書くタスクノートと、複数のタスクを俯瞰するタスクビューの両輪から扱う
    - 文脈とはノートに書いたことのすべて。何を書き、メンテナンスしていくかは自分次第だが、それがそのまま自分という文脈になる
  - 日々文脈を自分なりにノートに紡いでいき、「見えてきたら」タスク化して行動する。見えるまでの調査・思考・検討などの段階でノートを使うのが LTM
- LTM はライティングスタックに則る
  - 原本は頭の中ではなくノートにある。ノートが第二の脳になる
- タスクが見えてきたら、あるいはノートで扱いづらい事柄（アンカバード）は、素直に別のツールに頼るのが良い
  - しかしノートが定着しているのなら、適切なノートにタスクリストを書いておくくらいの手間でも意外と回る
  - そうは言っても所詮はノートであり、できることには限度がある（アンカバード）ので、別のツールに頼ることも忘れない
  - 極論、予定が多い人がカレンダーを使うことは避けられない、ここもノートで全部カバーしようとするのはやりすぎ

目的やポテンシャル:

- LTM の目的は QoL の向上
  - 自分のノートを読み書きすることは純粋に楽しいし、自分と深く向き合うので満足感も高くなる
  - ストイックに生産性や成長を追い求めるというよりは、QoL を高める営み
- LTM ではノートを読み書きするための恒常的な余裕が必要
  - 最低でも1日1時間、ノートを読み書きする余裕が恒常的に欲しい

    - 慌ただしい場面との相性は悪い

手段:

- LTM ではネットワーク型ノートツールを使う
    - Scrapbox のようなオンラインで出来の良いツールか、Obsidian などオフラインでも動作できるエディタか
    - 肝心なのはノートとしてザクザク読み書きできるポテンシャルと、`[ ]`や`[[ ]]`といった記法でリンク化してノート同士を繋いでネットワーク化できる世界観
- タスク管理というよりもネットワーク型ノートのノート術が重要となる
    - 本章でもいくつかのテクニックを紹介している
    - 例: ホーム、日付ノート、トランク、エリアとマーク、切り出し、エイリアス等

=== Chapter-25 探索的タスク管理 ===

タスク管理は「指向的な」営みと言えます。計画、設計、目標、手順など色んな言い方がありますが、あらかじめ基準をつくってからそのとおりに動きます。そのとおりに動くことが前提となっているため、動けているかどうかの観測つまりは管理が必要です――至極当たり前に聞こえますが、これは考え方の一つにすぎません。限界や不向きも色々とあります。

本章では、もう一つの考え方となる「探索的な」営みにフォーカスします。基準に頼らず、好き勝手に動いてみることで視界を広げていきます。基準をつくる必要がなく管理も要らないため、自由に楽しく進めていけます。とはいえ何も無いと迷いますから、小さな基準を動的につくって羅針盤にします。指向的な考え方から見るといいかげんで適当に見えますが、これが有効となる場面も多々あります。特に指向的な考え方に必要な基準が無い、もしくはつくれない状況下で、そのヒントを集める際に重宝します。つまり、まず探索して、ある程度見えてから指向に切り替えるというハイブリッドです（このハイブリッド戦略はタスクソースの章でも少し述べました）。

本章では探索的タスク管理と題して、「指向的な営み」の前段階で使えそうな「探索的な営み」のやり方と考え方を整理します。なお、**前章の文芸的タスク管理を理解していることを前提とします。**つまり言語的な読み書きが多発します。

探索的タスク管理の概要

探索的タスク管理

探索的タスク管理（Exploratory Task Management、以下 ETM と略します）とは、探索的なタスク管理を指します。

探索的 = No ABCD

探索的とは No ABCD を満たすことです。以下の ABCD が無いという意味です。

- Assign
  - アサインしない
  - 「やらないといけないこと」を正式に決めて、そこに誰かや自分を割り当てる、ということをしないという意味です
- Ball
  - ボールを持たない・渡さない
  - そもそもボールは使いません
- Consensus
  - 合意を取らない
  - 各自が好き勝手に好きなことをやればいいと考えます、合意はいちいち取りません
- Deadline
  - 締切をつくらない
  - 締切に追われるといったことも発生しません

指向的な営みが排除されているとも言えます。正解もなければ基準もなく、どう動くかは自分次第です。自分の意思で、自分なりに動いていくことになります。これが探索です。

少人数を想定するが個人でも可能

ETM は少人数（2〜5人程度）で行えるポテンシャルがあり、本章でも少人数で行うことを想定しています。前述の ABCD に他者とのコミュニケーション要素が含まれているのはそのためです。

しかし ETM は個人で行うこともできますので、適宜読み替えてください。

ETM はパラダイムシフト

ETM は **探索的（Exploratory）なアプローチ** を使います。これは従来の考え方とは大きく異なるものです。この点を説明するために、まず従来の話から詳しくしていきます。

従来は **指向的（Directional）なアプローチ** であり、冒頭でも述べたとおり、先に基準をつくってからそれに従うあり方でした。このアプローチは直感的でわかりやすく、管理もしやすいため、ビジネスから個人まで幅広く使われます。あまりに当たり前すぎるため名前すらついていません。しかし探索的な考え方を述べるにあたっては名前がないと不便ですから、指向的と名付け

ました（レトロニムと呼ばれます）。

指向的なアプローチには構造的な欠陥が二つあります。

一つは「あそび」がないことです。一度決めた基準に従うということは、基準以外を許さないことでもあります。余裕、余暇、余地といったものもありません。もっとも現実的には基準を「目安」程度に捉えつつ融通を利かせることも多いですが、それでもたかが知れています。基準へのとらわれからは逃れられません。基準を必要とする文脈では重宝しますが、そうではない場合は過剰なのです。なのに指向的なアプローチしかやり方を知らないから無理やりにでも基準を定めようとします。そしてそれが仕事だと、そういうものだと正当化します。特に現在は管理上の理由で都合が良かったり、その管理方法が会社のルールのレベルで組み込まれていたりするため抗いづらい――どころか抗う発想すら持てないという事情もあります。

あそびがないと何がまずいかというと、まず融通が利かないことです。仮に試せそうなやり方が 20 あるとしても、指向的なアプローチだとせいぜい数個しか試せません。20 を全部、は難しいにしても、もっとじっくり色々試してみようじゃないか、といったことができないのです。これは各々が自分の多様性を発揮して主体的に行動しづらいとも言えます。結局、誰かの監督下・指揮命令下で言われるがままに動く、という単純なモデルになってしまいます。前進させるには向いていますし、管理者が有能なら融通を利かせた制御もできますが、そんな有能な人はそうはいません。通常は管理者がボトルネックになり、融通も大して利きません。極端に言えばワンマンか、ワンマンに近しいあり方にしかなりません。

そして、仮に有能なワンマンだったとしても、従う側は楽しくないのです。特に管理過多（マイクロマネジメント）は最悪で、恐怖政治などムチをちらつかせるか、収入や立場など原始的な報酬というアメを与えるか、あるいは宗教信者的に信仰で盲信させる（筆者は「コエ」と呼んでます）かしないと続きません。人によってはそれでもストレスや過労で倒れます。管理過多というと、監視やルールばかりを想像しがちですが、それらがなくても単に忙しすぎる場合も同様です。実際に忙しすぎる人は十中八九ムチ・アメ・コエに依存していると思います。それを正当化します、たとえば資本主義だ、お金は大事だ、稼げるだけ稼いで何が悪い、稼げてない負け犬は黙っておれ――と、少し話がそれましたが、要はニンジンありきになってしまいます。

もう一つは「試行錯誤のペースが遅い」ことです。指向的なアプローチでは基準をしっかりとつくりがちですし、一度つくった基準はなかなか変えられません。特に基準をつくったり管理したりする側と、基準に従って動く側とが分かれているがゆえに、前者のフィードバックを後者に伝えづらいというコミュニケーションコストの高さもあります。複数人で協調したり、ひとりでも中長期的に望む場合はたしかに基準が大事（でないと活動がブレて収拾がつきません）ですが、大事にするがゆえに試行錯誤のペースが遅くなります。基準を上手くつくれるなら問題ないのですが、状況によってはそうもいかないことがあります。特に昨今は VUCA であり、先が見えない中で新しい取り組みをすることも少なくないでしょう。いきなり上手い基準などつくれるはずがありません。実際そうであるからこそ、リーンスタートアップやアジャイル開発など仮説検証のスピードを挙げる手法が登場したのだと思います。ちなみに、これらの手法もまだ指向的なアプローチの域を出ていません

まとめると、指向的なアプローチには以下の欠陥があります（※1）。

- あそびがないこと
  - 融通が利かない。つくった基準やそれを管理する管理者の存在がボトルネックとなるため
  - ニンジンが必要になる。ワンマン的なあり方は楽しくないし、ストレスフルであるため
- 試行錯誤のペースが遅いこと
  - 基準をつくる側とそれに従って動く側とで分かれているがゆえに、コミュニケーションコストがかかりがち
  - 先が見えない場面ではそもそも不適切
  - 仮説検証のペースを挙げる手法はすでに存在するが、これらもまだ指向的なアプローチの域

指向的アプローチそのものの必要性を否定しているわけではないことに注意してください。指向的なアプローチは（あえて言葉にするまでもなく）非常に有益ですし、これだけでも人生はやっていけます。それでも上記のような限界があるため、別の道具をつくろうとしているのです。それが **探索的なアプローチ** です。

特に本書はタスク管理の本ですから、タスク管理と絡めて整理します。それが ETM です。あそびをつくり、試行錯誤のペースも早められるようなタスク管理のあり方を導きます。

- ※
  - 1 付随的なものなので挙げていませんが、欠陥はもう一つあります。「管理コストがかかる」ことです。全部で 100 のリソースが使えたとしても、管理のために 10〜20 くらいは使ってしまいます。マネージャーの役割は言わずと知られていますが、これは管理コストがそれなりにかかるがゆえに専任者を置く必要があるからです。そして、マネージャーがいるということは、その分のコミュニケーションコストもかかってしまいます。実質敵に半分の 50 も使ええないことは決して珍しくありません。探索的なアプローチを取ると、この管理コストを（もちろんコミュニケーションコストも含めて）なくせます。マネージャーも要りません。100 使えるとしたら、90 くらい使えます。それでも多少の支えとやり取りは必要ですので 100 をフルで、というわけにはいきませんが。

メリットは自由と楽しさ

ETM のメリットは自由と楽しさです。

従うべき基準が無く、各自が好きなように探索するため自由です。この自由がパワーを生みます。

特に自由だと楽しいです。探索的なアプローチを行うと、私達がいかに基準に縛られているかがよくわかります。この感覚は創造的なもので、特に物語をつくっている営みと似ていると思います。逆を言えば、このような自由な営みに楽しさを感じられない人には合いません（おそらく指向

的なアプローチでガンガン基準を決めてそれに従って、と猛進するのが合っているでしょう）。楽しさ、というと何とも幼稚に聞こえるかもしれませんが、楽しさは重要です（※1）。本書でも散々述べてきたように、私達は怠け者です。タスク管理に頼らねばならないほど、あるいは頼ることすらできないほど怠ける生き物です。だからこそ楽しさは貴重なモチベーションになります。子供は意味もなくあちこちをうろついたり、おもちゃをいじったりしますが、そのような根源的な楽しさが探索にはあります。

自由が生み出すパワーとして、もう一つ典型的なものがあります。納得感です。自由には責任が伴うと言いますが、まさにそのとおりで、何をどの順番でどれだけ深堀りするかも全部自分で判断して行うことになります。判断の連続であり、実は非常に疲れる（だからこそ ETM としてここを和らげるテクニックが重要）ものですが、ここには「すべて自分がやってきた」という確かな事実があります。当事者感と言い換えてもいいでしょう。自分で携わっているからこそ、たとえ結果が優れなくても、醜くても、納得感が得られます。

他にも様々なパワーを生み出すかもしれませんが、主なものは上記のとおり、楽しさと納得感だと思います。指向的なアプローチの欠陥だった「あそびのなさ」をカバーできています。

一方、もう一つの欠陥である「試行錯誤のペース」が上がることについては、メリットとしては掲げてません。ともすると上がらないかもしれません。試行錯誤は、仮説検証も含めて本質的に難しい営みだからです。ETM で「上がります」と言えるほど単純ではないのです。とはいえ、ETM により自由に楽しく取り組むことによって、初動のペースを上げることはできると考えます。指向的なアプローチの制約を取っ払って頭手足を動かすことは、とてつもなくエネルギッシュでエキサイトです。

- ※
  - 1「楽しさ」がわかりづらければ「面白さ」でも構いません。楽しさドリブンや面白さドリブンという言い方も見られますが、まさにそのとおりです。通常このような生き方は個人プレイでなければできませんし、個人でもどうやればいいかわからなかったりするものですが、ETM は一つの解を与えます。

目的は視界を広げること

ETM の目的は単純明快で、視界を広げることです。

正解も無く、事例もなく、ともすると何がわからないのかさえわからないような場面は現代ではあるあるだと思います。ETM を使うと、この暗中を自分なりに照らすことができます。「自分なりに」なので正解とは限りませんが、それでもまがいなりには見えているわけです。見えていれば行動できます。それこそ、いつものように指向的なアプローチで攻めることもできます（し、見えたのならそうして一つずつ進めていくのが望ましい）。

ただ、指向的なアプローチだけではこのような場面に対抗できませんでした。だからこそ ETM を使うのです。まずは ETM で探索して視界を広げるのです。

適用シーンは Unknown Unknown

ETM が適しているのは「何もわからない」と言えるような場面です。制約が強すぎて何もできないというよりは、情報不足により何も思いつかないとか、逆に制約が無さすぎてどうとでもできてしまうとかいったことです。特に「何がわからないのかわからない（Unknown Unknown）」はあるあるだと思いますが、まさに ETM に適しています。

ETM により自由に探索することによって、まがいなりにも視界が広がります。視界が広がれば、見えてくるものも増えてきて、さらに色々思いついたり気づけたりします。そうしていけば、そのうち「大体見えてくる」状態になります。ここまで来たらしめたもので、あとは従来どおり指向的なアプローチとタスク管理でどうとでもなります。

言葉にすると当たり前に聞こえますが、この最初の、探索して視界を広げる部分が難しいのです。ETM は、この部分を担う方法論の一つです。

ETM の歴史

筆者による提唱のため、歴史はありません。

参考にしているものは多数あります。特に文芸的タスク管理でも用いるノートテイキング（ノート取り）の話と、発想法に代表されるような知的生産（※1）の話を重点的に取り入れています。もちろん ETM もタスク管理ではあるのでタスク管理の話も入れてますが、前述のとおり指向的なアプローチではないため、「管理」の仕方も変わってきます。

- ※
  - 1 知的生産という言葉は多義語だと思いますが、特に書籍『知的生産の技術』をベースにしてます。引用すると『かんたんにいえば、知的生産というのは、頭をはたらかせて、なにかあたらしいことがら――情報――を、ひとにわかるかたちで提出すること』です。筆者としては、自分なりに新しい情報（知識や見解）をつくって、かつ他者にもわかるように整理することと捉えてます。自分なりで良いこと、自分にとっては新しいこと、そして他者が理解できるよう言語化することがポイントです。

ダンジョンや地形を探索するイメージ

では ETM の考え方とやり方はどのようなものでしょうか。どのようにタスク管理を行っていくのでしょうか。

考え方としては、ダンジョンや地形の探索とアイテム集めに似ています。広大で不確かなそれらを自分なりに探索し、あとで迷わないようにマッピングもしながら少しずつ広げていきます。できれば完全制覇したいところですが、どれだけ広いかなんてわからないですし、疲労やモチベーションや実力（HP や攻撃力や魔法や装備といったステータス）の問題もありますし、さらにたと

えを飛躍させると「ダンジョンや地形自体も変わることがある」ため、きりがありません。どこかで切り上げます。集めたアイテムが戦利品となります。これらを使って、次の（指向的なアプローチが通用する）ダンジョンや地形で勝負するのです。

どこまでやるかは自分次第ですが、ポイントが二つあります。まず冒頭でも述べたとおり No ABCD を心がけます。探索でも自分なりの管理は要りますが、ABCD を使うほどかしこまった管理はしません。次に、なるべく 100% を出し切って「現時点の自分ではこれ以上はない」と胸を張れることを目指します――というと過激に聞こえますが、「まあこんなものか」を自分なりに言語化して示せれば良いというイメージです。

次にやり方ですが、どこをどう探索するかの作戦決め、マッピングの仕方、自分自身のステータスの把握と、疲労やモチベーションやその他リソースなどの制御と監視といったことを行います。ETM も Task Management であり管理ですが、管理としてそういったことをするわけです。道具としては基本的にはノートを使って、言語で書きます（図や絵ではなく）。書いたものを読み返して膨らませたり、要らないとわかったのでバッサリ捨てて新しく書き始めたりもします。この点は文芸的タスク管理と同様です。

ETM ことはじめ1 ～情報単位を知る～

ここからは「ことはじめ」と題して、ETM を始めるのに必要なやり方と考え方を解説していきます。ここ part1 から始まり、part4 まであります。また、ヒント集として part5 も用意しています。

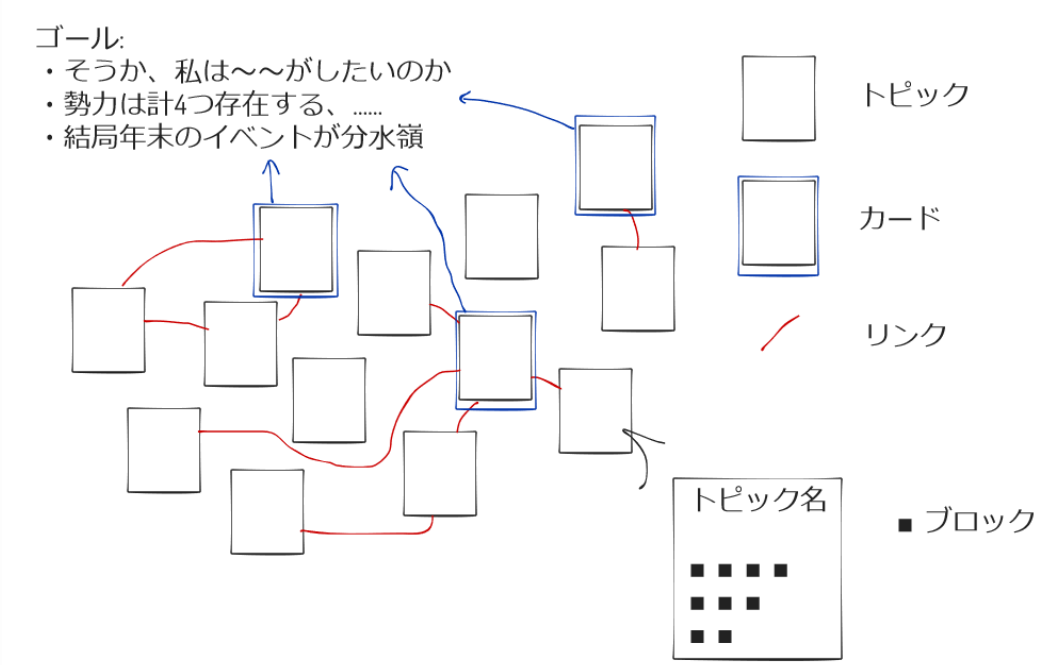
part1 では ETM において扱う情報単位について解説します。

最初にまとめ

登場する単位:

- ゴール
- カード
- トピック
- ブロック

各単位の関係:



解説:

- ETM ではゴールを導くためにカードを集めます
  - ゴールとは「見えてきた」もの
  - カードとはゴールを導くための「役に立つトピック」
- ETM ではトピックをつくっていきます
  - トピックとは「一つの話題を表現した単位」
  - トピックは一つのノートで表現される
  - どう表現するかに正解はない
- トピックはブロックから構成されます
  - ブロックとは情報を何らかの観点でまとめた塊
  - トピックは n のブロックから構成される
    - 論理的とは限らない
    - ブロックからトピックを導く（ボトムアップ）こともあれば、トピックからブロックを洗い出していく（トップダウン）こともある

手札（カード）を増やす

ETM の目的は、次のように言い換えることもできます。

> 「見えてきたら」を導くための手札（カード）を拡充する。

カードとは一つの知識、見解、意思決定、想像や仮説等のことです。これらを集めて、貯めていきます。情報源から知識を理解して持ってくる（インプット）だけで済むこともありますが、たい

ていは足りません。それをどのように使うかとか、足りない部分をどう補うかとか、そのままでは使いづらいので少し組み合わせて新しい知識にする、あるいは解釈が一意ではないがいったんこのように解釈するなど、自分なりの加工が必要となります。

カードの粒度（どこまでまとめたものを一枚のカードとするか）や揃える枚数に正解はありませんが、粒度は細かめ、枚数は数百枚以上になることは珍しくありません。タスク管理でもタスクの分解・細分化が重要だったり、一つ一つのタスクは具体的で実行可能なものが望ましいとはよく言われますが、ニュアンスとしては同じです。カードも一枚一枚は小さめで、わかりやすい方が良いです。

別の言い方をすると「パーツを揃える」「ボトムアップに積み上げていく」とも言えます。コンピュータに詳しい方は、Linux の KISS 原則を思い浮かべてください。小さくて汎用的なツールを組み合わせて使う、とする哲学です。違うとすれば、カードは汎用的になるとは限らないことです。非常に狭い特殊解を述べたカード、もよくあります。

ゴールとは「見えてきたこと」

カードを集めて何をするかというと、何かを導きたいのです。この何かを **ゴール** と呼びます。

ゴールは前もって見えているものではありません。また、仮説として設定しておくものでもありません。カードを集めていくうちに **自ずと見えてくる** ものです。あるいは執念でひねりだすものです。指向的なアプローチが染み付いていると、この感覚はわかりづらいかもしれませんが、そういうものだと捉えてください。

これは言い方を変えれば **使うカードが違えば見えてくるゴールも違ってくる** とも言えますし、何なら **同じカードでもあって人によって見えてくるゴールが違う** とも言えますし、さらには **同じカードと同じ人であっても見えてくるゴールが変わる** とも言えます。一方で、カードや人が違っても大体同じゴールが見えてくることももちろんあります。良くも悪くもゴールは不定なのです。

また、ゴールは単一とは限りません。30 のカードからたった 1 つのゴールを導くこともできれば、6 つのゴールが出てくることもあります。

カードとは「役に立つトピック」

ETM では、実際にはトピックという単位で情報をつくっていきます。トピックについては後述します。

カードとは役に立つトピックのことです。たとえばトピックが 100 個あったとすると、カードとして使えるのは 20 個だったり 30 個だったり、あるいは 5 個だったりします。何をカードとみなすかに正解はありません。

トピックとブロック

ETM において実際につくっていく単位をトピックといいます。また、トピックはブロックから構成されます。ここは ETM の根幹となる概念なので詳しく解説します。

トピック

まずは **トピック** です。前章で述べたトピック指向に相当し、一つの話題を指します。

またトピックごとにノートを分けます。仮にトピックが 20 個あるなら、ノート（実際の単位はファイルだったりページだったりと使うツール次第です）も 20 個あるはずです。一つのノートに 20 個のトピックを全部書き殴っている、みたいなことはしません。その状態ではトピックとは言えません。1-トピック 1-ノートになるようにちゃんとつくっていく必要があります。ただし、色んな情報を詰め込んだ雑多なノートは、それはそれでトピックになります。ただしカードとしては使いづらいでしょう。最終的にはゴールを導くためのカードを増やしたいのですから、カードとして使いやすくするためにも **一つのトピックで一つの話題を表現する** という単位感を意識したいです。これがトピック指向という考え方です。

トピックの例を挙げます。たとえば、とあるベンチャー企業で IT エンジニアを新たに数名雇いたいとして、そもそも IT エンジニアってなんだっけ、うちはどんなエンジニアが欲しいんだっけとなった場合を考えましょう。まずは自分達なりに IT エンジニアとは何か、を考えたいです。このとき、以下はすべてトピックとなります。

- 1「IT エンジニア」
- 2「IT エンジニアとは」
- 3「IT エンジニア = システム、ソフトウェア、ネットワーク、インフラ、サイトリライアビリティ」
- 4「IT エンジニア ≒ システム、ソフトウェア、ネットワーク、インフラ、サイトリライアビリティ」

1 や 2 は良いトピックとは言えません。ただの用語であり、カードとしては使いづらいです。おそらく「IT エンジニア」ノートや「IT エンジニアとは」ノートを開くと、中に定義や意見が色々と書き込まれているでしょう。散らかった机のように散乱している可能性が高いと思います。あるいは、単に用語の定義に留めるくらいならアリです。

3 や 4 は理解を端的に記述しており、良いトピックと言えます。4 の方はニアリーイコール（だいたい同じ）を使っていて、厳密性を持たせていないこともわかります。

実際には両方混ざることも多いです。たとえば「IT エンジニア」ノートをつくり、その中から切り出しを行って「IT エンジニア ≒ システム、ソフトウェア、ネットワーク、インフラ、サイトリライアビリティ」ノートをつくる、との流れはよくあります。抽象的なトピックもあれば具体的なトピックもありますし、作業としてたくさん書き殴っている未整理の汚いトピックもあれば、そこから導かれた本質をシンプルに表現した綺麗なトピックもあります。

ブロック

次に **ブロック** ですが、これは一つのトピック内に存在する一構成要素です。包含関係としては 1-トピック N-ブロックとなります。つまり一つのトピックは単一または複数のブロックを含みます。

トピックとブロックの関係は特にわかりづらいので、いくつか補足します。

- **論理的であるとは限りません**
  - あるトピックが 4 個のブロックを持っていたとして、そのトピックはその 4 個のブロックから論理的に導かれているとは限りません。各ブロックは使ったかもしれないし、使わなかったかもしれないのです
  - 統一的な使い方もありません。同じ 4 個のブロックでも、A さんと B さんとでは違うトピックを導くかもしれません
- **ボトムアップとトップダウンの両方がありえます**
  - ボトムアップ: n 個のブロックからトピックを導く
  - トップダウン: トピックから n 個のブロックを導く
  - ETM ではどちらもありえます。カードからゴールを導く部分などボトムアップはわかりやすいですが、トップダウンもありえます。トップダウンとしてよくあるのはお題です。トピックとして何らかのお題を設定し、そこから思いつくことを書き込んでいきます。アイデア出しなど発散的な文脈でもよく登場します

さて、ブロックですが、単にトピックの中身を指すものではありません。トピックに単に情報が書き込まれた段階では、ブロックがあるとは言えないのです。ブロックとは塊であり、特定の観点でまとめられた情報群を指します。まとめ方は雑でも構いませんが、とにかくまとめます。最終的には「トピックは n 個のブロックから構成されている」という構図にしたいのです。

ETM では部品化（パーツ）を意識しています。ゴールを導く際にもカードという部品を使いますし、普段情報を読み書きするときもトピックという部品単位で行います。そして、トピックもまたブロックという単位で構成されているのです。このように部品化をイメージすることで情報を扱いやすくします。たとえば部品と部品を組み合わせるとか、この部品は要らないから捨てるといったことができるようになります。

また、トピックも同様、ブロックのつくりかたにも正解はありません。とはいえ、比較的使いやすいつくりかたはいくつかありますので後で紹介します。

ブロックについても例を挙げてみましょう。先の例を使うと、おそらく「IT エンジニア = システム、ソフトウェア、ネットワーク、インフラ、サイトリライアビリティ」トピックには以下のようなブロックがあるでしょう。

- 辞書から抜粋した定義
- Aさんの第一印象
- Bさんの第一印象
- Aさん vs Bさんの議論のログ

- エンジニアの例を皆で洗い出したもの（システム、ソフトウェアなど多数並ぶ）
- システムエンジニアとは
- スタッフエンジニアとは etc

このようなブロックを色々書いた上で、IT エンジニアとはシステムエンジニア、ソフトウェアエンジニア、ネットワークエンジニア、インフラエンジニア、SRE（サイトリライアビリティエンジニア）の 5 種類があるのだな、などと判断するわけです。判断した結果がこのトピックであり、ノートで表現すると「IT エンジニア = システム、ソフトウェア、ネットワーク、インフラ、サイトリライアビリティ」ノートになるわけです。もちろん判断の仕方次第で如何様にも変わります。たとえば、この例ではスタッフエンジニアという役割を含めていません。もちろん含めることもできますし、両取りすることもできます――「IT エンジニア」ノートから「従来の IT エンジニア」ノートと「最近の IT エンジニア」ノートをつくり、後者の定義にスタッフエンジニアを含める、といったことです（ちなみにスタッフエンジニアは 2023 年頃から提唱され始めた比較的新しめの役割）。これもあくまで一例でであり、やりようは他にも無数にあります。

カードは「特に役に立つトピック」

カードとトピックは同義ですが、カードは使えること――特に繰り返し使えたり皆が使えたりすることを重視しています。

トピック自体は使えるものから使えないものまでたくさんありますが、それら全部を手札としたところで収拾がつきません。それこそゴミがたくさんあっても仕方ないわけです。なので、ちゃんと使えるもの（カード）を揃えます。

ETM ことはじめ2 ～トピックの育て方を知る～

part1 ではトピックの概念を整理しました。ここ part2 では、トピックをどのように育てていくかを扱います。

広げる、詰める、導く

トピックに対して行えることは、大まかには以下の 3 つです。

- 広げる(Spread)
- 詰める(Detailize)
- 導く(Summarize and Distill)

ETM ではこれらの操作を行き来しながらトピックを育てていきます。このとき、複数のトピックに

またがることも珍しくありません。

典型的な営みを一つ挙げます。たとえば「引っ越し」トピックで引っ越しの検討をしている（詰めている）ときに、ふと本棚が圧迫されている話になって、電子書籍を本格的に導入するべきか迷っていることを思い出したとします。引っ越し自体の検討はきりがいいとして、タイミングも良かったとします、ならちょうどいいですね、考えてみましょう……というわけで「電子書籍を導入する？」トピックをつくって、思いつくことを書き込みます（広げる）。幸いにも急ぐ必要はないため、何日もかけて、のんびりやりました。二週間ほど経ちましたが、ダラダラ続けても仕方がないので、そろそろ結論を出したい考え、まとめに入りました（導く）。自分の本心は何なのか、何がしたいのか、という観点でこれまでのノートを読み返し、気付いたことも追記して考察を続けました。その結果、「そうか、図書館や作家の書斎みたいに広々とした空間で読書を楽しみたいんだ」と気付きます（導く）。まだ違和感があるので、さらに粘って、最終的には「私にとって狭いことは非常にストレスフルらしい」とわかりました（導く）。ついでに、この観点で今回の引っ越しを見直すために、「引っ越し」トピックにて思いつくことを書き殴ってみます（広げる）――

いかがでしょうか。必要に応じてトピックをつくったり移ったりしつつ、広げてみたり、逆に詰めてみたり、あるいはある程度集まったので導いてみたり、と行き来していることがわかります。自由奔放に見えますが、本質は広げる・詰める・導くの 3 つです。

続いて各操作の詳細を見ていきます。

広げる

広げる（Spread） とは発散です。思いつくことや洗い出せることを出していきます。たとえば、お題に対して思いつくことを書いてみる等です。より具体的には、引っ越ししたい場合や転職したい場合などに、お題を「引っ越し」とか「2024/08/20 前々から転職したいと思ってるので広げてみる」などと設定して、思いつくことを書いてみたりします。お題は明確である必要はありません。「なんかモヤモヤする」と題して、思いつくことを書いてもいいのです。アイデア出し、フリーライティング、ひとりブレストなど、この手の営みや手法は多数存在しますし、大喜利や連想ゲームなど娯楽として行われることもあります。

ETM ではトピックをベースとするので、実際には「引っ越し」ノートや「なんかモヤモヤする」ノートをつくることになります（お題名のノートをつくる）。あるいは、どのノートで行っても構いませんが、あとでトピック化（最低でもブロック化）します。より実践に即して言えば、ノートツールの都合でノート名が被ることがあるので「引っ越し1」や「引っ越し 2024/08/20」など番号や日付をつけて区別させることもよくあります。名前を考えるのが面倒な場合は「untitled」「無題」とか「あ」とか「aaaaaa」とかでも構いません。あるいは「メモ1」「メモ2」など雑に書く用のノートをあらかじめつくっておいてそれを開く、でも良いでしょう。

詰める

詰める（Detailize）とは拡充です。わからないことや不明なことを調べたり、自分なりに検討してみたり、色んな人の声を集めてみたり、あるいはリンク集など情報を集めたり動線を整えたりといった作業も含みます。中身を詰めて埋めていくイメージです。英語の Detailize は造語ですが、Detail（詳細）をつくっていくニュアンスを込めています。

何をどう詰めるかについては後述します。

詰めるものは事実と主観に大別できます。この 2 つはなるべく区別した方が良いでしょう。特に ETM を複数人で行う場合、**事実と主観のバランス感覚**が人や組織によって違うためにこじれやすいです。どちらを、どれだけ、どのように使うかに正解はありません（※1）し、各々が模索するしかないことですが、それでも区別ができていなければ使い分けることもできなくなります。区別の仕方についても後述します。

- ※
  - 1 事実ばかりだと意思の無い学者になります。主観ばかりだと根拠のない評論家になります。状況により、どちらかへの偏重が重要となることこそありますが、どちらにどれだけ頼ればいいかには通常正解はありません（というより際限なく追求できてきりがない）。その都度バランスを模索することになります。とはいえ、現実的には ETM を行う個人や小集団の価値観に左右され、また扱うトピックもその価値観に合った方が重視される傾向があると思います。

導く

導くとは、広げたものと詰めたものを見て、何かを導き出すことです。

要約と蒸留があります。

要約（Summerize）は、今ある部品から忠実に導きます。部品は改変せずに全部使います（使えない部品を意図的に使わないのはアリ）。導出の整合性はピンキリで、論理的だったり、ストーリーテイストで情動的だったり、あるいは考察や私見をたくさん交えて主観的にすることもできます。いずれにせよ自分なりにもっともらしくつくれれば OK ですが、ともかく部品は忠実に全部使います。

蒸留（Distill）は、部品をヒントとみなすスタンスです。部品を全部使わなくてもいいですし、そのまま使わず適当にアレンジしたり一部だけ切り取って使っても構いません。導くときに新しい思いつきや主観を入れても OK です。良く言えば大胆、悪く言えば誠意の無いスタンスです。このスタンスを筆者は**ダシにする**と呼んでいます。元ネタは知的生産の技術で、以下に抜粋します。

> いわば本をダシにして、自分のかってなかんがえを開発し、そだててゆくというやりかたである。……このやりかたなら、読書はひとつの創造的行為となる

本をダシにするとありますが、ETM では本に限りません。トピックも、ブロックも、部品化されてい

ない文章も、すべてダシにすることができます。それこそ他者の意見や作品をダシにすることもできますし、やります。

どちらも一長一短です。要約は堅実で荒れにくいですが、ETM のメリットである自由と楽しさを発揮しづらいです。創造性を発揮しづらいとも言えます。一方、蒸留は自由で楽しく創造的ですが、非常に失礼な行為となることもあり、使い方を間違えれば人間関係に亀裂が入ります。また（特に個人の場合は）視野狭窄に陥る恐れがあります。

それでも、できれば蒸留に頼りたいです。蒸留による飛躍――特に肯定的な急進を **ブレイクスルー** と呼びますが、このブレイクスルーこそが ETM の醍醐味だからです。究極的に言えば、**ETM とはブレイクスルーを味わうための創造的な営み** です。そもそも要約で済むなら指向的なアプローチで事足ります。それでは通用しない場面や段階があるからこそ、探索的なアプローチを――ETM を採用するのです。このような状況では正解は存在せず、自分がどう意思決定するか次第で無限の未来が広がっています。このときに頼れるのは自分の主観に他なりません。主観をフル活用して、あらゆるものをダシにして本質をひねり出すのです。そうすることでしか見えてこないものがあります。仮に見つからなかったとしても、それはそれで構いません（ひねり出したという経験から納得感とひねりだせなかったという事実がある）。

と、小難しく書きましたが、要は創造的な解決を狙っています。そのために ETM ではトピックという形で言語的に部品を扱うのです。No ABCD を掲げ、自由を謳うなど制約が少なく設計されているのもそのためです。ETM は実は創造的な営みなのです。

トピックを用いた知的生産にすぎない

広げる、詰める、導くとまとめましたが、このような営みは既に知られており、言及の仕方も様々です。

広げるについてはブレインストーミングはよく知られていますし、個人で文章を書くやり方ならフリーライティングもあります。またGTDでも、収集ステップとして頭の中にある「気になること」を出していきますが、これも発散の一種です。娯楽としても大喜利やマジカルバナナなど連想ゲームは多数知られています。

詰めるについては、これ単独というよりも広げる行為とセットで扱われることが多いでしょう。発散と収束という言い方はすでにビジネス用語ですし、アウトライナーの文脈ではTak. による「シェイク」の概念もあります。科学でも演繹法と帰納法がありますし、より一般的にはボトムアップとトップダウンともいいます。共通しているのは広げる行為と詰める行為の行き来です。両者を行き来することで色んな考えや気付きが得られ、思考の隙をなくしていけます。

導くについては、最終的な結論を出す部分です。仕事の文脈では意思決定と呼ばれて、素早く行うとの前提を含むことが多いですが、プレゼンでサマリーをつくったり、物語のあらすじを考えたりなど意外と身近です。専門的になりますが、ソフトウェア開発の一手法であるドメイン駆動開発は、用語集と設計モデルを用いた顧客との対話を通じてコアドメイン（ビジネスの資産

となる概念的な本質）を導こうとします。これは蒸留と呼ばれており、ETM の用語もここから取ってきています。

仮にこのような営みを知的生産と呼ぶとして、知的生産自体は別段珍しくも何ともありません。この 3 つをすべてこなしてしまう手法として KJ 法がありますが、これは 50 年以上前から存在しているようです。

ETM も知的生産の一手法にすぎません。文芸的タスク管理をベースにして、探索的にタスク管理を行うために、トピックだとかブロックだとかいった概念を導入して再構成したにすぎないのです。ですので、知的生産の営みをご存知の方は、それらを思い浮かべながら読み進めていただくと理解しやすいでしょう。逆にご存知ない方は、上述した既存の手法から勉強・試行してみることでも理解が進むと思います。

ETM ことはじめ3 ～タスク管理の仕方を知る～

part1 にてトピックの概念を、part2 にてトピックの育て方を整理しました。ここ part3 ではタスク管理の話に入ります。ETM におけるタスク管理とは何か、またどのような考え方とやり方に従うかを扱います。

ETM におけるタスク管理

ETM におけるタスク管理は以下から成ります。

- タスク管理以前の管理
  - 1 ロードの管理（直近何を重視するか）
  - 2 ターゲットの管理（直近何のトピックを重点的にいじるか）
  - 3 リソースの管理（体力ややる気と上手く付き合う）
  - 4 コンテキストの管理（あとで再開しやすいよう日頃から整理する）
- タスク管理
  - 5 タスクの管理（文芸的タスク管理）

実はタスク管理以前の管理がメインです。ETM は探索的なアプローチであるため、探索を上手くやるための管理への比重が重くなります。それが 1～4、ロードとターゲットとリソースとコンテキストです。無理やりタスク管理という言葉を使うのなら、「終了条件の見えない曖昧なタスク」の管理と言えます。自分が何をどれだけやるかという自己制御に力点を置きます。

探索を行っていると次第にタスクも見えてきます。あるいは、見えなくとも「いったんこれとこれをやる」と決める（タスク化する）こともできます。このような時にようやく 5 のタスク管理が登場しま

す。やり方や考え方としては文芸的タスク管理に従います。詳細はそちらを見ていただくとして、ここでもかんたんにまとめておくと、次のとおりです。

- ホームを基点として日々ノートを育てていきます
- タスクについてはタスクの詳細を書くタスクノートと、複数のタスクを俯瞰するタスクビューを使い分けます
- 文芸的タスク管理ではカバー出来ない部分（アンカバード）については、素直に既存のツールに頼ります

別の言い方をすると、1〜4 では「やってもやらなくてもいい」ことを扱います。やるやらないの判断やどこまでやるかの判断が頻繁に必要であり、単に忘迷怠を減らして一つずつこなせばいいだけではないため従来のタスク管理は使えません。探索を上手くやるための管理が必要であり、それを ETM ではロード、ターゲット、リソース、コンテキストという形で整理しています。次に、探索を通じて、ある程度見えてきたら「やらなきゃいけないこと」や「やると決めたこと」などタスクという言葉が従来持つ意味合いのものが増えてきますので、ここで従来のタスク管理を使います。手段としてはノートに頼っていますので、5 のとおり文芸的タスク管理にて行います。

以降では各管理を詳しく見ていきます。

ロードの管理

ETM では直近何に取り組むかを動的に決めます。たとえば普段の持ち分は 5 個、今週は A,B,C,D,E の 5 個に取り組む、来週は F と G をやりたいのでいったん A と C は保留にする、といったようなことです。これを **ロード（Load）** と呼びます。

ロード

ロードの直訳は荷重、積荷、負荷ですが、ETM においては **直近取り組みたいこと（重視したいこと）として抱えているもの** を指します。

ロードにはいくつか性質があります。

- **ロードキャパシティ**
  - 何個抱えられるかが人によって違います
  - 5 個かもしれませんし、3 個や 7 個かもしれませんが、おそらく一桁に収まるはずです
  - 同じ人でも普段は 5 個だが調子が悪いときは 3 個、など変わります（かつ変えても良いです）
- 粗さ
  - タスクよりもかなり粒度が粗いです
  - 例
    - □「引っ越し」「転職を検討する」「生成 AI との付き合い方を考えねば」
    - □「引越し先業者の選定」「大手 n 社の採用ページを調べる」「ChatGPT

Plus を契約して使ってみる」
    - ロードは様々なタスクを内包しています
        - そういう意味ではタスクソースも似ていますし、目標事項や維持事項はまさにロードにできます
- 不確定性
    - ロードは探索の対象となる事柄であり、通常終わりは見えません、あるいは見えていても変更が可能です
    - どこまでやるか、どこで終わらせるかは自分自身で意思決定します
    - ここで終了・ここで中止と決定すれば終わりですし、決定しない限りは続きます

ニュアンスとしては「常に背負っている荷物」です。終わらない限りはずっと背負い続けたままですし、そんなにたくさん持つこともできません。人間ひとりが持てる荷物などたかが知れています。また、常時持っているので毎日使う機会があります（ので使いやすい＝取り組みやすい）、もちろん毎日必ず使わなければならないわけではありません。

一桁の原則

ロードキャパシティですが、身軽に過ごして融通を利かせたいなら少なくするべきです。逆にどうしても重視したいものが多数あるなら、できるだけ積みたいところです。それでも限度があり、一桁（9個以内）には収まるはずです。これを **一桁の原則** と呼びます。

筆者の体感では 9 個でも多すぎます。半分の 5 個でもちょうど良いか、まだ多いくらいです。もし一桁の原則を超えられるというなら、それは単にロードの捕捉が下手なだけでしょう。たとえば、やることが明確だったり比較的すぐに終わったりする「タスク」をロードとして掲げているかもしれません。あるいは十分な実力があれば（常人にとってはロードでも）タスクのように処理できますが、これは稀ですし、それができるなら ETM は要りません。それができないからこそ ETM にて探索的にアプローチするのです。

ロードキャパシティの可視化

複数人で行う場合、メンバーのロードはお互いに見えた方が良いでしょう。ロードノートやロードページのような場所をつくってもいいですし、ホームに載せてもいいです。ただ人に知らせたくないプライベートなロードもありますので、そこは伏せても構いません。たとえば「そろそろ結婚したいのでマッチングアプリや婚活をガチっている」という場合、正直にロードとして見せなくても、「私用」の一言で OK です。内容を伏せていてもロードが存在することはわかります。逆にロードとして計上する必要がないのであれば、いちいい書く必要はありません。

このとおり、重要なのは **ロードキャパシティの可視化** です。ETM では他者を管理しないので、可視化したところで何らかの干渉を行うわけではありませんが、可視化できていれば皆の様子や全体の状況がわかり、自分の行動にも反映できます。また誰が何を重視しているかを見ることは刺激にもなります。皆のロードキャパシティは、いわば空気と呼べるものです。

ターゲットの管理

ETM ではトピックをつくっていくのでした。トピックは多数つくられます。何十、何百は当たり前ですし、中長期的に続ければ何千や何万にも増えていきます。これらにすべて均等に取り組んだり、真摯に深堀りしたりするのは不可能です。選別が必要です。

トピックを 3T モデルで選別する

選別の手法として **3T モデル** があります。これはトピックを Topic, Target, Task の 3 状態で捉え、より重視したいものを Topic → Target → Task と昇格させていきます。

各状態は次のようになっています。

- 1: Topic
  - ただのトピック
  - 誰でも好き勝手に読み書きできる
  - やってもやらなくてもいい
- 2: Target
  - 「当事者」が存在するトピック
  - 当事者は責任を負い、自らリードして進める。意思決定権も持つ
  - 当事者以外は「助言者」であり、助言は可能だが意思決定権は無い
- 3: Task
  - 当事者が存在し、かつ当事者への「催促」も許されたトピック

トピックはデフォルトでは 1 の Topic です。これはただのトピックであり、誰がどのように読み書きしようが自由です。スルーも可能ですし、「興味ないからいいや」と投げることもできます。自由です。もちろん熱心に取り組んでも構いません。

Topic のうち、より重視したいものは Target にします。トピックに対して誰かが **当事者** を名乗り出ると、そのトピックは Target になります。Target は当事者のリードで進めていきます。意思決定権も当事者が持ちます。当事者はリードの責任を負うかわりに裁量を得るのです。当事者以外は全員助言者（※1）となり、意思決定権はありません。

Target のうち、催促したいものは Task にします。ETM では管理は行わず、これはつまり **通知も飛ばさないことを意味します（なので会議を開催したりメンションを飛ばしたりすることも想定していません）** が、一方であまり悠長にされると困る Target というものが出てきます。そんなときに Task に昇格させます。Task 状態のトピックでは、当事者に催促を行うことが許可されます。**催促とは忘迷怠のフォローを行うこと** です。といっても通知は飛ばせないので、新しくノートをつくったり、ホームなど動線上で行ったりします。「～～の件どうなってる？」とか「3 日間音沙汰無いけどどした？」といったことです。その名のとおり、早くしてほしい旨をつっつくわけです。

この 3T モデルは非常に重要、かつ中身も濃いので、以降ではもう少し詳しく補足していきます。

- ※
  - 1 この部分はティール組織本の助言者モデルを参考にしています

Target は表明

トピックに対して当事者だと名乗り出ることは表明にも等しい行為です。「私は今からこのトピックを少し重視しますよ」と皆に告げるようなものですし、Target が出たということは各人も「ああ、この人がこのトピックをこれから重視するんだな」と理解します。

しかし約束ではありません。何度も言うように ETM では強制や管理はしません。ただ重視することを表明するだけです。

当事者キャパシティとロードキャパシティ

ETM を実践する各人は 2 つのキャパシティを持ちます。

- 前述したロードキャパシティ
- 当事者として抱える Target の数を示した **当事者キャパシティ**

両者は違うことに注意してください。ロードキャパシティは「直近はこういうことを重視します」という大局的な方針（ロード）のキャパシティを示したものですが、当事者キャパシティは「直近はこれこれのトピックを重視します」という局所的、というよりトピックレベルのキャパシティを示したものです。

当事者キャパシティについても一桁の原則は当てはまります。ロードよりも粒度が細かいため、10 以上抱えることも難しくないですが、あまり Target が多すぎても形骸化するだけですので一桁で十分です。だからといって何個抱えているかを意識する必要もありません。多すぎたら形骸化の兆候――Target の割には更新されないトピックが増えてきたとか、Task への昇格が増えてきてなんか慌ただしいなとかいったことが出てくるので、その時に減らせば済みます。

可視化については、当事者キャパシティには必要ありません。トピックをどう読むかは各人に委ねられていますし、盛り上がっているトピックがあるとノートツール上でも目立つため気付けるからです。ただし当事者の新着くらいはわかるようにしておいた方が便利でしょう。たとえばトピック A で当事者になりたい場合に `[当事者]` や `[[当事者]]` のようにリンクすれば、「当事者」ノートからバックリンクの形でトピック A がわかります。

当事者はよく変わる

当事者の席は動的です。

最初 A さんが当事者になったが、実力やらやる気やらが芳しくなくて早々に「やっぱやめます」と諦めて外したとします。このトピックにはもう当事者はいないのかというと、そんなことはなくて、「じゃあ次は私がやってみるわ」と B さんが当事者になってもいいのです。その B さんも一区切りつけて、いい感じに仕上がった後に、さらに上手くやれるからと今度は C さんが当事者になってさらに詰めて――なども可能です。

ダメならば外せばいいだけですし、助言者による助言にも助けられます（ひとりで行う場合は期待できません）。変に敷居を上げず、気軽に付けたり外したりすれば良いのです。

強いて言えば、細かい議論に凝りだすときりがないので、トピックから派生させると良いでしょう。つまり一段落ついたら、そのトピック A は確定とみなして、もう当事者になるのはやめます。かわりに、何か言いたいことやさらに詰めたいことがあるなら、そのトピック A から派生する形で新たなトピック B をつくり、B で当事者になるようにします。そもそも収束が遠い場合は当事者にならず、Topic の状態で議論を重ねた方が良いです。

当事者は数人でも良いが、できれば 1 人で

当事者は複数人も可能ですが、現存する当事者全員の同意がなければ増やせません。1 人目は早い者勝ち、2 人目は 1 人目が同意したら OK、3 人目は 2 人が同意したら OK――という形です。

とはいえ人数が多いと当事者というシステムの意味がなくなるので、通常は 1 人だと考えてください。これを **単一当事者の原則** と呼びます。もし 2 人や 3 人が望ましいと思える場合も、トピックの粒度が粗いことが多いです。トピック A を B, C, D に分解して、それぞれ 1 人ずつ当事者になった方が上手くいきます。

ETM は探索的な営みであり、探索は一人で行うものです。一番わかりやすいのは作家だと思います。作品は作家一人がつくります。分担やレビューこそありますが、複数人で一緒につくることはありません。正解が無い世界なので、いちいち協調していてはきりがないのです。なので一人で責任を持ってつくって、つくったものをチェックして品質を高めます。ETM も同じです。Target には当事者を一人だけ立てて、その人が責任を持って詰めて、その結果を皆に見てもらうのです。もちろん途中で助言を受けることは可能ですし、推奨しますが、あくまでも詰める責任は当事者自身にあります。

もし当事者を 1 人にしづらい場合、構造的な問題が潜んでいます。具体的には **権限委譲がしづらい組織構造や文化になっている** か、**探索が有効となるシチュエーションではない** かのいずれかです。対処が必要ですので、次項で説明します。

単一当事者の原則を阻む問題に対処する

まず権限委譲のしづらさについては、従来ありがちな階層組織だと解消は難しいです。特にマネージャーなど権限を持つ者がボトルネックになります。役割分担や適材適所は構いません

が、メンバー全員が対等であることが必要です。対等とは意思決定のレベルで委譲できることを指します。「必ず私のチェックを通せ」「私がチェックするまで待て」ではなく「わかった、任せるよ」ということです。

これは想像以上に難しく、意思決定者に権限を握らせるのをやめさせるだけではなく、各メンバー側にも自律性と主体性が求められます。自分の意思を出さない人には務まりません。できるかどうかは実質メンバー全員の能力と性格（相性含む）次第です。もし再現性のあるやり方が欲しいなら、助言者モデルの例でも述べたティール組織が現状の解だと思います。

次に探索が有効となるシチュエーションではない件については、まず No ABCD を当てはめられるかが目安です。ETM では No ABCD ができないと話になりません。よくある原因のトップツーは「変えることのできないタイトなスケジュール」と「メンバー全員が継続的に探索を行えないほど少ない予算」ですが、ちゃんと見れば意外と行える余地はあります。たとえば、一見すると変えられないスケジュールがあったとしても、実は単にお偉いさんやその中間の管理職が適当に言っているだけだったりします。特に現代のサラリーマンや経営者はまだまだ指向的なアプローチが根強く、細かいスケジュール管理を課してくる傾向があるので、意識的に抗わねばなりません。このあたりのやり方は次の part4 で述べます。予算についても、1日1時間の時間くらいなら捻出できるでしょうし、ETM を普段のコミュニケーション手段にすれば日常の一部にもできます。

もう一つ、ETM において意識したいのが **トピックはブースターである** という点です。ブースター（**Booster**）とは実際行動に移す際に使える各種情報のことで、行動をブーストさせるとの意味合いがあります。実は ETM で行うのは、実際の行動というよりも **行動可能なところまで情報（考察や議論を経た意思決定も含む）を揃える** 部分です。というのも、いざ行動するとなると現実の様々な制約に阻まれてしまい、探索的なアプローチができないからです。ETM はトピックをこねてゴールを導くものですが、もっと言えばトピックをこねる形でブースターを揃えることでゴールを導く、とも言えます。

つまり行動まで済ませようとするシチュエーションは ETM には合いません。ETM はあくまで行動する直前の、準備や整理の部分までを済ませるものです。この準備や整理を自由に楽しくやろうぜ、と言っているのです。

一つ例を出しましょう。ある大企業で新規事業用に少人数チームを組んで ETM をしているときに、利便性とセキュリティの観点から「全員に社有の iPhone を持たせるべき」とのトピックが出たとします。これは Target として、A さんが当事者となって出した結論です。ETM の範囲は、この結論を出すところまでです。実際に iPhone を入手して、会社の調達体制も整えて、予算や会計もちゃんと反映させて――といった行動はしません（かんたんにできるのならやっても良いが、この文脈では無理でしょう）。全員に社有の iPhone を持たせるべき、というトピックはこれ自体が有益なブースターです。探索的な検討や議論を経た結論だからです。**実際に採用するかどうかは別問題です**。この点も次の part4 で述べますが、実際に行動に移すのは ETM でゴールを見つけた後になります。ETM の最中ではありません。ETM の最中は、あくまでも探索によりブースターを増やすことに専念します。

リソースの管理

本書でも度々顔を出してきた厄介者として「やる気」があります。他にも「体力」はよく知られています。体力というと身体能力をイメージしがちですが、頭脳面でもあります。判断に使う資源は注意資源とも呼ばれ、これが枯渇すると頭が働かなくなりますし、スマホや SNS でも消耗します。他にも社交エネルギーとかソーシャルバッテリーといった言葉もあります。

私たちは電気さえあれば無限に動けるロボットではなく、実際に動くためには何らかの資源（リソース）を使わねばなりません。リソースが残ってないと動けませんし、残っていたとしても、それをどう使うかの判断でもリソースを使いますから実際動ける分は意外と少ないのです。タスク管理というと、タスクをいかに可視化して一つずつこなすかにフォーカスされがちですが、実は、その可視化したタスクにどうリソースを振るかという意味での管理こそが重要だったりします。本書でも全体像を俯瞰したりヒントとして様々なやり方考え方を提示してきました（※1）。このリソース管理の必要性は ETM でも変わりません。

ETM は文芸的タスク管理でもあり、文芸的タスク管理でもこれだけでカバー出来ない部分（アンカバード）は通常のタスク管理に頼ります。なので、ここまで紹介してきたリソース管理あるいはそのためのヒントは ETM でも活用できます。しかし ETM において押さえておくと便利な考え方がありますので、本節にて整理します。

- ※
  - 1 たとえば自分に合った戦略やスタンスを選ぶことができますし、オルタスクとして取り上げたイベントやモットーやタスクソースを上手く扱うことでも助けになります。また応用編で述べた動線やリマインド、習慣、コンテキストなども把握して仕込んだり尊重したりすることでも高められるでしょう。

管理 2.0

ETM ではリソースを管理 2.0 的に管理します。

管理 2.0 とは、理想状態を維持できれば良いという前提のもと、理想状態の因子と計測するという管理のあり方です。従来は成果物の進み具合を進捗（n% 進んでます等）や予実（予定と実績の乖離を見る）によって管理する 管理 1.0 でしたが、そうではなく、あくまで状態を見ます。

元ネタはソフトウェア開発の話ですが Engineering Effectiveness で、出力の計測ではなく入力の計測と表現されています。また、生活の文脈では日々体調を記録するセルフモニタリングが知られており、これも管理 2.0 的と言えます。要は **結果を管理するのではなく、結果を生み出すもとである私たちの状態を計測しましょう** とする潮流です。名前自体は筆者が名付けたものですが、概念自体は目新しくはないと思います。

ETM は探索的な世界であり、管理もしないのでした。管理 1.0 はできません。しかし何もしな

いと忘迷怠が発生しますし、やる気と体力の問題があることは既に述べたとおりです。そこで折衷案として、自分のリソースの状態を計測するのです。日々計測し、記録していくことで、こういう状態だと生産的に過ごせるとか、逆にこういう出来事があると集中しづらくなるといった傾向を見つけます。傾向を把握できたら、あとは理想の状態に近づけるよう働きかけを行うだけです――良い状態の因子を増やしたり、逆に状態を悪くする因子を減らしたりします。

先の Engineering Effectiveness では計測因子の例として「コードレビューにかかる時間」「割り込みの量」「計画外の作業量」が挙げられています。エンジニアリングは創造的な仕事であり、指定されたテーマに、まとまった時間を使って集中して取り組むのが大事です（理想の状態）。レビューや割り込みや計画外といったものは、この理想を削ぐ因子なわけです。これらを計測することで、たとえば「レビューが半日以上続いた日はまともに生産できてない」とか「割り込みが 2 回以内の日は落ち着いて生産できるが、3 回を超えてくるとその後も増えていて平均 6 回ある」とかいったことがわかります。これらの記録を考察すれば改善できます。たとえば割り込みの回数を 2 回以内に制限したり、レビューを行う日を限定したりすれば良いでしょう。

管理 1.0 的に成果物の進捗や予実を管理しているわけではありません。そもそも単純作業でもないのに 1.0 的に管理すること自体に無理があります。かわりに、できるだけ理想状態を維持できればいいと考えます。理想状態の最適化です。そのために理想状態に絡んでそうな因子を計測します。それが管理 2.0 であり、ETM もこのやり方に則ります。

理想状態とは

そもそも理想状態とは何かという議論がありますが、明確な解は用意していません。

ETM は管理無しに自由に行うことを前提としているので、あえて定義するなら「自分の生活と心身を尊重しながらも、普段のパフォーマンスを発揮できること」でしょうか。

筆者は健康であること、ひとりで集中できること、でも退屈と阻害も無くフィードバックも来ること、自分のペースが尊重されること、情報がオープンでありいちいち伺わなくても存在すること、あるいは言えばすぐに出してくれるか、出せないならその理由もちゃんと出してくれること、そして恒常的に余裕があることなどを好みます。会議やイベントなどの拘束は好みませんし、むしろ害悪だと捉えます。しかし全く無いのは寂しいし、望ましくないので、たまには（稀には）欲しいです。一方で、毎日顔を合わせてワイワイしたい人もいるでしょうし、純粋にライフスタイルだけを見ても筆者のように超朝型の人もいれば夜型の人もいますし、ワークライフバランス的にメリハリをつける人もいれば、ワークアズライフ的に区別はせず隙あらば仕事を差し込む人もいます。人によって理想は違います。

しかしながら ETM は探索的なアプローチを取るため、探索を阻害する状態――もっと言えば管理的な営みを肯定または促進するような状態は想定しません。別の言い方をすると、理想状態は個人に閉じており、また完結させるべきです。「必ず週に一度は出社して顔合わせをして遊ぼう」のようなことは人を巻き込んでいる（管理に等しい）ため、ETM における理想状態とは言えません。

計測因子のポイント

実はこれを計測すればいい、と言えるほどの知見はまだありませんが、あたりはついていますのでいくつか紹介します。

計測の方針としては、以下を意識すると無駄な苦労を迂回できるでしょう。

- あまり細かい情報を計測しても意味がないこと
  - それこそライフログのように何でもかんでも記録したところで、後で読みません
  - また自動で記録できる情報にはあまり使い道がありません
- 計測する因子の個数は厳選して、一桁以内（〜9個）に抑えること
  - いまいち傾向が見えてこない場合は、計測する因子を変えます
  - また一桁といっても 9 個は多すぎて形骸化します、まずは数個以内にしたいです
- 計測する因子の選定は遠慮無く突き詰めること
  - もしかすると「A さんと顔を合わせた回数」が因子かもしれません
  - おそらく「これじゃないか」と感じる因子候補があるはずで、これを遠慮なく捉えるところから始まります

次に因子にはポジティブとネガティブがあります。ポジティブな因子は理想に近づけるもの、ネガティブな因子は理想から遠ざけるものです。別の言い方をすると、起きたらプラスになるのがポジティブで、起きたらマイナスになるのがネガティブです。朝のルーチン（食事 → 散歩 → シャワー → 軽くゲームして頭をあたためる）を全部こなせた日に仕事が捗っているというのなら、このルーチンはポジティブです。雨の日にパフォーマンスが出ていないとすれば、雨天はネガティブな因子です。あるいは「朝に日光を浴びる」というポジティブな因子があって、それを満たせてないのかもしれません。

理想状態を維持する戦略は、以下の 3 つがあります。

- 1 ミニマイザー。ネガティブを減らす戦略
- 2 マキシマイザー。ポジティブを増やす戦略
- 3 ハイブリッド。両方。ネガティブを減らすし、ポジティブも増やす

一見すると誰にでもポジティブとネガティブがあるので全員ハイブリッドか、と思いがちですが、意外と偏ります。ミニマイザーの人は、別に良いことなんてなくてもいいからとにかく嫌なことを避けたい、嫌なことさえなければパフォーマンスなんて出せると考えるタイプです。筆者もこちらにあたります。一方、マキシマイザーは、嫌なことが多少あってでも、それ以上に良いことを得たいと考えます。

目安として使えるのは旅行です。海外旅行に行きたい人はマキシマイザー、逆に行きたくない人はミニマイザーだと言えます。割合的にはマキシマイザーの方が多いのでそちらの解説は割愛して、ミニマイザーの心理を解説すると「慣れ親しんでてインフラと治安も整ってる日本から出る意味がわからん」となります。海外旅行を普段の生活水準が乱れるネガティブなイベントだと捉えているわけです。あるいはひとりビジネスホテルのように旅行先でも自分の居心地を重視する

あり方や、転じて「じゃあ単に遠出してホテルで過ごしてみればいいのでは？」となるような思考もミニマイザーでしょう。ここでコンフォートゾーンを出ないのがミニマイザーと思いがちですが、そうではありません。旅行はあくまでも例です。ミニマイザー、マキシマイザーにかかわらず出る人は出ますし、出ない人は出ません。ただ理想の定義と維持の戦略が違うだけです。

ミニマイザーの人はネガティブな因子を計測するのが良いです。マキシマイザーの人はポジティブな因子も計測できます。しかし、パフォーマンスは（特に ETM のように探索的な営みでは）繊細であり、邪魔されないことの方が重要ですので、総合的には **ネガティブな因子を計測する方が重要です**。ポジティブな因子は、見つかったらラッキーくらいの意識でいる方が良いでしょう。

最後に、因子の観点についても取り上げておきます。特にネガティブな因子を捉えるのに使える観点を挙げます。

- 拘束の数
  - 拘束とは指定時間中、場所や時間が束縛されることです（イベントの章を参照）
  - 要は会議やイベントが多いとか、会議でなくとも話し込むことが多いとかいったことです
    - 拘束に慣れている人は前者のうち、特に大きなものや疲れるものだけをカウントすればいいでしょう
- 割り込みの数
  - 割り込みとは何かに集中しているときに割って入られて中断させられることです
  - 仕事や人によっては割り込みに応じて動く世界もありますが、ETM では自分ひとりが自分のペースで取り組むことが大事なので、この因子は重要です
    - 別の言い方をすれば、前者のメンタルモデルで過ごす人は転換を余儀なくされます
  - できる限りゼロを目指したいです、少なくとも「多すぎて計測してられない」状態は論外であり、この状態をチェックするのにも役立ちます（計測できる程度には少ないことが ETM の必要条件）
- メンタルロードの数
  - メンタルロードとは頭の中で高頻度にちらつく不安や不満のことです
  - 一つ一つは大した害がなくとも、意外と理想状態を遠のかせています
  - この個数がある程度以上増えてくると一気に遠のきますので、増加の兆候をウォッチしたいです
- 暴力の有無
  - 暴力の定義は自分次第ですが、物理的な暴力はもちろん、各種ハラスメントや怒鳴ることも含みます
    - 筆者は個人的に「手で殴るのは昭和まで、怒気で殴る（怒鳴る）のは平成まで」との格言を掲げています
    - 別の例として、ソフトウェア開発の文脈では「優秀だけどイヤな奴」をブリリアント・ジャークと呼び、このような者は採用しなかったりクビにしたりする組織もあります
    - またマサカリではなくマシュマロとの考察もあります（マシュマロ派にとってはマサカリ自体が暴力だと解釈されうると思います）

    - この因子が 1 回でもあった場合、理想状態は大きく遠のきがちです
- 体調不良の有無
    - セルフモニタリングとして日々記録している人もいらっしゃるのではないかと思います
    - 例: 低気圧、筋肉痛、花粉、生理
- ネガティブなフィードバックの有無
    - ここでネガティブなフィードバックとは、成果物や意見やプロセスではなく自身の容姿・性格・主義思想に向けられたものを指します
    - わかりやすく言えば人格批判・否定であり、これは 1 回あっただけでも理想状態を遠ざけがちです
        - たとえば表面上は気にしない演技ができても、メンタルロードとしてくすぶり続けたりします

観点の度合いとして「数」と「有無」を挙げましたが、他にも「長さ」があります。計測のしやすさは 有無 > 数 > 長さ です。長さの計測は非常に面倒くさいので、可能ならば数、できれば有無にしたいところです。

計測方法とタイミング

計測方法もタイミングも正解はありません。自分に合ったやり方を選んでください。詳しくは習慣の章を参考にしていただきたいですが、ここでもかんたんにまとめておきます。

方法については、習慣トラッカーを使うか、自分で律儀に記録を取るかの二択となります。場合によってはデジタルツール上のログやデータを見ることでも知れますが、計測結果は日ごとに俯瞰できることが重要なので、それができる程度には加工が必要です（結局記録を取るのと同等の作業が要ります）。個人的には、先に観点を記述しておいて、あとから振り返って当てはまる部分に印をつけるのがシンプルで良いと思います。これを突き詰めたのがまさに習慣トラッカーです。

どこに記録するかについても自由です。ETM で使っているノートでもいいですし、ノートから離れて別のアプリを使ってもいいですし、手帳などアナログでも良いでしょう。

タイミングについては、人によって最適解が分かれます。大まかには一日の終わりに一度だけ振り返って書くか、一日に数回（昼休憩前・退勤前・就寝前など）書くか、あるいは気付いたときに随時書くかの三択です。タイミングを減らせば減らすほど楽はできますが、何が計測されたかを覚えているとは限らず記録の精度が落ちます。逆にタイミングを増やせば精度は増やせますが、純粋に記録を取るのが面倒です。

計測結果を踏まえてどうするか

計測しました。結果が出ました。では、これらを踏まえて何をすれば良いでしょうか。

もちろん理想状態に近づくための対処を考えます。

ポジティブな因子の場合、どうやったら増えるかを考えて取り入れましょう。たとえば「普段はリモートだけど 1on1 でたっぷり雑談した後はなぜかパフォーマンスが良い」とわかったとするなら、たっぷりの雑談がおそらくポジティブな因子ですので、雑談する機会を増やします。個人的にリモート雑談を申し込んでもいいですし、機会を増やすために出社頻度を増やしてみてもいいでしょう。あるいは業務中に雑談をせず、プライベートで確保してもいいかもしれません。

しかしながら、狙って増やせることはあまりありません。ポジティブな因子は突発的なイベントによってもたらされることが多いからです。「起きたらラッキーだよね」と捉えるくらいがちょうどいいと思います。あるいは、狙って機会を増やしに行くこともちろん可能ですが、大きな労力がかかりがちです。なぜなら、後述するネガティブな場合（取り除けばいい）とは違って、新たに獲得する営みになるからです。何を手に入れたり増やしたりすることは、通常捨てることよりもエネルギーも時間もかかります。マキシマイザーの人ならそれでも増やしにいけるでしょうが、ミニマイザーの人は腰が上がらないと思います。腰が上がらない自分を責める必要はありません、そういうものです。

ネガティブな因子の場合、どうやったら減らせるかを考えて取り入れます。発生源が何かしらあるはずで、そこから距離を置いたり、それに直接働きかけたりします。そういったアクティブな対応が嫌な場合は、自分の心持ちを変える（捉え方を変える）ことでも可能ですし、自己啓発的にはよく言われることですが、そんなことができれば苦労はしません。あるいはできるにしても、中長期的に悟っていくものであり、短期的にすぐに身につくものではありません。発生源との対峙は多かれ少なかれ必要です。ここを認めて、愚直に行動できるかどうかに軽減の成否がかかっています。

たとえば仕事でも私生活でも趣味でも何でもいいですが、前述したブリリアントジャーク（優秀だけどイヤな奴）がいて、その人の物言いのせいで有意にパフォーマンスが落ちていることがわかっているとします。この問題は、その人を追放するか、その人を変えるか、自分がその人から離れるかしない限りは終わりません。かんたんに対処できれば苦労はしないですし、こちらが消耗することもよくありますが、それでも何かしら行動しない限りは終わりません。

そういう意味で、**リソースの管理には結局痛みを伴う行動が求められます**。本書でも何度も述べてきた盤外戦ですね。もし、これを行う気がないのだとしたら、管理しても意味はないので管理しなくて良いです。性格的に向いてない人はいます。無理して取り組む必要はありませんし、取り組むことが是というわけでもありません。とはいえ、他人に働きかけるのは無理にしても、自分の行動を変えるだけで対処できるものもあります。仮に X を見すぎているせいでパフォーマンスが出ないのだとしたら、X のアプリやアカウントを消せば済むことです（依存症的な側面もありこれはこれで難しいですが）し、仮に自分で消すことさえできなかったとしても単に「X のせいでパフォーマンスが落ちている」と知れているだけでも違います。

そうです、仮に行動できなかったとしても、リソースの管理にてネガティブな因子を把握しておくだけでも違うのです。把握できていたら、そのうち対処のモチベーションやタイミングが訪れるかもしれません。

コンテキストの管理

ETM ではロードを管理することで直近何をやるかを掲げ、ターゲットを管理――3T モデルに基づいて当事者をに志願することで直近どのトピックを重視するかを掲げます。この二つの管理は取り組む対象を有限化していると言えますが、これだけではまだ不十分です。

ETM は探索的であり、一度の着手では終了しないことがよくあります。トピックは多数あるので、同じトピックばかり着手し続けるとも限りません。ある程度期間が空いてから再度着手するとか、全く別の頭の使い方をしたあとに再開するとかいったことがよくあります。このとき、前回はどこまでやっていたのかとか次は何をするつもりだったのかといった文脈（コンテキスト）を思い出す必要があり、これをコンテキストスイッチングと呼びますが、非常に消耗する営みです。ETM を上手くやれるかどうかは、このコンテキストスイッチングの消耗をいかに抑えるかにかかっています。

つまりコンテキストの管理とはコンテキストスイッチングの対処であり、ロードコンテキスト（頭の中に入れる文脈情報）の管理だと言えます。コンテキストの章ではコンテキストスイッチング・モデルとして土地でたとえる説明をしましたが、本節では論点を小さく絞った上で、実践的なテクニックの紹介に比重を置きます。

本節では以降からロードコンテキストのことをコンテキストと略して書くことにします。

忘れない程度に継続するか、すぐ思い出せるよう整えるか

コンテキストの管理として本節で紹介するのは 2 つです。

- コンテキストメンテナンス
  - 忘れないように高頻度で着手します
- コンテキスト駆動
  - 着手を終わる前に、その時点のコンテキストを残しておくようにします
  - そうすると次、着手するときにその残したものを読むことですぐに思い出せます

以降でそれぞれ詳しく見ていきます。

コンテキストメンテナンス

コンテキストメンテナンス（Context Maintenance） とは、今取り組んでいる事柄のコンテキストを脳内に留めるアプローチで、忘れない程度に高頻度に、その事柄に取り組むことを指します。一流の職人やアスリートは感覚を鈍らせないために毎日欠かさず練習しますが、似たようなものです。忘れてしまうのが問題なら、忘れないよう高頻度に続けてしまえば良いのです。

とはいえ、一日 n 時間以上、のように一流の練習量をつぎ込むわけにはいかないでしょう。そこ

でコンテキストメンテナンスでは「ほんの少しでも着手できたら良い」と考えます。この考え方は習慣の章ですでに紹介した「着手ベース」他、書籍としては『先送り0（ゼロ）』で訴求されています。これらの考え方をさらにアレンジして、コンテキストメンテナンスでは「着手はゼロだけど読み返した」も許容します。コンテキストを維持できれば良いからです。たとえ進捗が無かったとしても、読み返すだけで脳内のコンテキストはある程度保持されます。つまりコンテキストメンテナンスとは **脳内のコンテキストを維持するために、とりあえず触れるだけの行為** と言えます。

もっともわかりやすいのが長大な文書の作成でしょう。プレゼン資料、論文、レポートやブログや書籍など、何回も取り組まねば完成に至れない文章の仕事がありますが、このような仕事はまさにコンテキストの管理が求められるものです。しかし実質、やっていることは「書いたことを読み返す」ことのはずです。読み返していれば脳内のコンテキストは保持されやすいですし、仮に頭が仕事モードではなかったとしても温まってきたりします。触れるだけでもそれなりに効果があるのです。ETM も本質的には言語を用いたノート取りであり、これらの仕事と似たようなものです。

コンテキストメンテナンスの良い点は、取り組みやすいことです。たとえ疲れていて続きに取り組む気力がなかったとしても、軽く触れる程度であればできます。ダラダラ読んでもいいですし、タイマーを 30 秒にセットして読むだけでも OK です。そもそもそうした補助も面倒くさいので、その時その時で行える程度でやればいいでしょう。

コンテキストメンテナンスをやるかやらないかで、後日のコンテキストスイッチングの負担が変わってきます。それはあってもなくてもわからないような軽微な差かもしれませんが、ETM ではたくさんのトピックを扱います。この小さな差が後々聞いてきます。一流を持ち出すつもりはありませんが、この小さな蓄積はコンテキストを脳内に留める訓練の蓄積であり、中長期的に ETM 実践者の地力を引き上げます。

コンテキスト駆動

コンテキストメンテナンスは「忘れない程度に高頻度に触る」でしたが、別のアプローチもあります。それが「忘れても思い出せるようにしておく」であり、具体的に言うと **着手を終わる度にその時点でのコンテキストを書いて残しておきます**。すると、次に着手するときにそこを読むことで、前回のコンテキストを思い出せるのです。これを **コンテキスト駆動仕事（Context Driven Work）**、あるいは単に **コンテキスト駆動** と呼びます。

コンテキスト駆動の論点は以下 3 点です。

- 1 コンテキストはどの単位で書き残すのか
- 2 コンテキストとして何を書き残すのか
- 2 コンテキストをトピック内のどこに書くのか

1 の単位については、トピックです。原理的には 1-トピック 1-コンテキスト が望ましいですが、さすがにやってられません。そもそも作業的でないトピックにはコンテキストもありません（自分がいつ読んだとか読んでどう思ったか等の感想を書くこともできなくはない）。そういうわけで、現実的

には必要に応じて書いてください、となります。

次に 2 の何を書くかですが、正解はありません。未来の自分がまた来て読んだときにすぐ思い出せるような文章であれば何でも構いません。思い出せるなら文章ではなく単語でも良いです。推奨したいのは「次に何をすればいいか」と「なぜこのトピックに取り組んでいるのか」の二点を明らかにすることです。やることややる理由が不明瞭だとモチベーションが出ずに先延ばしにしがちです。特に後者のなぜについては、与えられた状況だけではなく自分の見解もセットで書いてください。「だるいけど必要なのでやる」や「必要らしいけど気に入らないのでやる気ない」といった気持ちが見えていると良いです。というのも、ETM は管理も強制もせず、やるやらないもどこまでやるかも自分が決めることだからです。**意思決定の変遷（特に直近）もコンテキストに含めてください** ということです。

よくあるアンチパターンとして進捗を書く人が多いですが、これも ETM においては望ましくありません。もう一度書きますが、ETM では管理はしません。進捗は管理のための観点であり、ETM では基本的に使うことはありません。あるいは全体像を書いて俯瞰したり、今どこまで終わっているかを可視化してやる気を引き出したりするためには使っても良いですが、少なくともコンテキストとして書くことは避けたいです。進捗という概念は本質的に管理的であり、これをコンテキストの中に含めてしまうと、コンテキスト駆動の営みそのものが管理の色を帯びてしまいます。

最後にコンテキストをトピック内（ノート内）のどこに書くかですが、ここも正解はありません。筆者はノートの行頭が好みですが、進行中の部分――行末でも全 100 行中の 36 行目付近でもどこでも構いません。重要なのは、どこにコンテキストが書いてあるかを迷わずにすぐ探して読めることです。記号や区切り線をつけたり、□（ノート）や□（のうみそ）など絵文字をつけて視覚的に目立たせても良いでしょう。テキストがお好みなら `(context)` のようなタグ文字列を書いて、それを検索してヒットさせることもできます。

同時にはしない

コンテキストを管理する方法としてコンテキストメンテナンスとコンテキスト駆動を紹介しましたが、片方ではなく両方を使った方がより管理が捗ります。

その際、コンテキストメンテナンス中はコンテキストを書かないようにしてください。コンテキストを書いたり更新したりすることは、単に読むことよりもはるかに消耗するからです。消耗するからと腰が上がりにくくなり、コンテキストメンテナンス自体の腰も上がりにくくなってしまいます。

両方使うというのは、たとえばトピック A ではコンテキストメンテナンスを行って、トピック B ではコンテキスト駆動をするといったこと、あるいはトピック A で元々コンテキストメンテナンスしてたけどコンテキスト駆動に切り替えるといったことです。トピック A においてコンテキストメンテナンスとコンテキスト駆動を **両方同時に行うことは意味しません**。なぜ同時がダメかというと、頭の使い方が違うからです。使い方の違う営みを混ぜてしまうと非常に消耗します。

特に「あ、良いこと思いついた」など不意に良い考えが振ってくることがありますが、これをコンテ

キストに含めてしまうと、後々未来の自分が見たときに混乱します。思いついたことがあれば、それはコンテキストとしてではなく、そのトピックのコンテンツとして、コンテキスト以外の部分に書いておきましょう。その場で深堀りできるならしてしまえばいいです（コンテキストメンテナンスのつもりが普通の検討になってますが望ましいことです）し、できないならメモとして残しておくだけです。もちろんメモの場合は、忘れないうちに後で処理することを忘れずに。

タスクの管理

本節冒頭でも解説したため割愛します。

まとめ

- 探索を上手くやるための管理が色々と必要です
- ロードの管理
  - 直近重視すること（ロード）を掲げます
  - 抱える個数は一桁以内にします（一桁の原則）
  - 抱えられる個数は人によって違うので、自ら尊重します（ロードキャパシティ）
  - メンバー全員のロードを可視化しておくと、全体の「空気」が出てきて便利です
- ターゲットの管理
  - 直近重視するトピックを掲げます
  - 3T モデルに従い、Topic → Target → Task と昇格させていきます（当事者の志願）
  - 当事者にもキャパシティがありますが、個人的なものなので可視化は不要です
    - しかしどのトピックに当事者が表れたかはわかるようにしておきたいです
  - 当事者は可能な限り 1 人が良いです（単一当事者の原則）
    - 満たせない場合、構造的な問題が潜んでいます
  - 重要な考え方として、ETM はそもそもブースター（行動時に使う “よく検討された” インプット）を揃えるものです
    - 実際に行動することとは切り離してください。その前の準備や整備までを行います
    - もちろんかんたんに行動できるのならやってしまって構いませんが、No ABCD など探索的な世界は壊さないようにします
- リソースの管理
  - 理想状態に影響を与える因子を計測します（管理2.0）
  - 因子にはポジティブとネガティブがありますが、ネガティブな因子の計測が重要です
  - 計測結果が何をすればいいかを教えてくれます
    - 理想状態を遠ざける or 近づけてくれる因子がわかる → その因子のもとに働きかける
  - 計測結果があるからといって行動できるとは限りませんが、結果があるだけ（因子を特定できているだけ）でも意味はあります
- コンテキストの管理
  - コンテキストスイッチングの負担を減らします

    - 忘れない程度に着手し続けるコンテキストメンテナンスと、忘れてもすぐ思い出せるよう書き込んでおくコンテキスト駆動があります
    - 両方とも取り組むのが望ましいですが、同時に二つを行わないようにします
        - 特にコンテキストメンテナンス中にコンテキスト駆動をしてしまわないように注意してください

ETM ことはじめ4 ～ワークフローを知る～

part1 と part2 ではトピックベースの世界観をお伝えしました。part3 ではどうやって探索的にタスク管理するのか、特に何をどう管理するのかをお話ししました。

ここ part4 では、ETM を実際に始めて運用していくための全体の流れを解説します。

全体の流れ

ETM の流れ自体はシンプルです。

- Step1: セットアップ
    - スコープと期間を決める、参加者を決める、会場をつくる
- Step2: ETM
    - step1 に基づいて ETM を行う
- Step3: ゴール判定
    - step1 の期間が終了したタイミングで振り返りを行い、次のアクションを決める
    - 足りているなら指向的なアプローチに切り替える。足りてないなら ETM を再度 step1 から回す

Step1: セットアップ

ETM を始める前に準備が必要です。

スコープと探索期間

まずはスコープ（ETM にてどんなゴールを導きたいか）と期間（いつまで ETM を行うか）を決めます。

スコープは探索的なアプローチが通用するものである必要があります。No ABCD が通用することに加え、以下の三点を満たしてください。

- 絶対クリアしなければならないミッションなるものがないこと

- 定量的な指標が含まれていないこと
- 達成できなくても存続が可能であること

また、目指すべきゴールはどうせ変わるので暫定で構いません。そういうわけで、実は「何か新規事業をつくる」くらいにラフでも良かったりします。むしろこれだけラフな方が「そもそも新規事業とは」「うちの状況は？」「なんでつくりたいんだっけ？」「今持ってる手札やリソース」など色んな観点を出せます。スコープは最初の方向づけにすぎず。抽象的な方が上手くいきます。ただし、あまり抽象的すぎて何も出ないという場合は、具体側に寄せた何かを暫定でいいので掲げてください。

もう一つ、いつまで ETM を行うかという期間も決めます。「今日から二週間」「8/14 から 1 ヶ月間」といったように連続して取ります。これを **探索期間** と呼びます。探索期間はフルタイムで取るべきであり、週に一度だけ 20% ルールとして探索する、のようなことはあってはなりません。**探索期間中は探索のみ行えるようにし、従来行っていた指向的なアプローチは一切ストップさせます**。この点は非常に重要で、ここができないなら ETM は諦めてください。ETM をやりたいなら、探索期間中のフルコミットを確保しなければなりません。妥協案として「午前中」か「午後以降」のどちらか片方だけフルで取るハーフも可能ですが、探索が阻害されないように注意してください。たとえば午前中に探索、午後は通常どおりとした場合、午前中に探索だけします（No ABCD を保障します）。油断するとすぐに侵食されてなんちゃって探索期間になってしまいます。

探索期間はたとえるなら長期休暇のようなものです。休暇中に一切仕事を入れないように、探索期間中も一切指向的なアプローチを入れません。これは実質的に、ETM を行うメンバー全員に（探索期間中を自律的に過ごせるだけの）自律性を要求します。また、彼らをフォローする側もこの流儀に従わねばなりません。よくあるのが、ETM を行っているメンバー達の管理者が探索期間中に「経過を聞きたいから会議設定よろしく」などとすることです。No ABCD を破ってはなりません。管理者が状況を知りたいなら、自ら ETM の会場に足を運んで、No ABCD の流儀に基づいて情報収集・コミュニケーションをしてください。**探索期間という聖域は絶対に侵攻させてはなりません**。しつこく強調しますが、この点は ETM において最重要とも呼べるほど重要です。

ここで探索期間中は一切指向的なアプローチをしない、と言っているわけ **ではない** ことに注意してください。あくまでも仕事への取り組み方の話です。探索期間中の仕事の過ごし方は No ABCD でなければなりませんが、スコープから外れた仕事や私生活はその限りではありませんし、期間中にスコープ外のタスクをこなすのはもちろん OK です。ただし、ETM は探索期間中はフルコミットすることを前提としていますので、絶対的なリソース（時間やお金）がそもそも足りない、なんてことがないようにしてください。仮に 1 日 6 時間（午前3h + 午後3h）働くとしたら、許容できるのはハーフ（3時間）までです。さらに言えば、ETM は探索的な営みであり、まとまった時間は絶対に必要ですので、**最低3時間は確保してください**。

この時間、つまりは探索期間中の「一日における探索時間」を **探索時間** と呼びます。探索時間は最低 3 時間確保してください。それ以上でも問題ありませんが、各自の事情を反映し

てください。1 日 8 時間働く会社では 8 時間（ハーフなら 4 時間）になるでしょうし、融通の利く組織や個人であれば 3 時間でも 5 時間でも 10 時間でも自由にできるでしょう。

メンバー

ETM を行うメンバーを選別します。

本章で散々述べてきたとおり、特殊な手法ですので適性があります。向いていない人のカバーやフォローは想定していないので、そのような人を含めてはなりません。必要な要素を言語化するのは難しいですが、参考までに 4 つほど挙げます。

- 自分ひとりでも読み書きしていける自律性
- 文芸的タスク管理を行える程度の各種リテラシー
    - 言語化能力やタイピングスピードやツールの使い方など
- 親しい相手でなくともコメントや意見を率直に書き込める何か
    - 感性でも性格でも演技力でもメンタルでも構いません
    - ETM は読み書きの営みであり、親しさの醸成は通常できないか、少なくとも遅いので、できないまま進める力が必須です
- 会話や会議に頼らずに過ごす耐性、またはメンバーを巻き込まず自分で勝手に解消する要領

一つでも欠けていると、おそらく ETM は続かないでしょう。

適性がある場合、できるとかできそうだとすぐにわかります。逆に無いか、わからない場合は無理だと即答したり「いやわからんけどどうなんだろう」と悩んだりします。もし現時点で適性がわからないなら、練習してみることをおすすめします。

筆者がおすすめするエクササイズは **ミュートデイ** です。これは一日間、誰とも一言も喋らず、またメンションなど通知を要する拘束も行わずに過ごすことを指します。ミュートデイができたら、次は **ミュートウィーク** に挑戦してみましょう。こちらは一週間です。どちらも非常に難しく感じるかもしれませんが、本章でここまで述べてきたことがヒントとなっています。この程度もできないようでは ETM は務まりません。ミュートデイやミュートウィークは一日または一週間、と短期であり、中長期に耐えるためのメンテナンスと管理が不要で済む分、まだ易しいのです。

よりステップアップ的にやりたいのであれば、以下の順が良いでしょう。

- 1 ひとりでミュートデイ
- 2 ひとりでミュートウィーク
- 3 メンバー全員でミュートデイ
- 4 メンバー全員でミュートウィーク

ひとりで行う場合は退屈や孤独に勝たねばなりません。複数人で行う場合は会話や会議といった誘惑・衝動に耐えねばなりません。両方とも必要です。ひとりで ETM を行うなら前者だけで問題ありませんが、複数で行う場合はメンバー全員が前者も後者もどちらも備えておかね

ばなりません。前者を備えてないと病みますし、後者を備えてないと備えてない人が足を引っ張って ETM が破綻します。

ミュートデイやミュートウィークは良い練習になりますし、実際にやってみることで課題が山ほど出てきます。つまり一度の実施で可否が確定するものではなく、必要なら繰り返すことで鍛えていけます。

会場

ETM を行うとは、何らかのノートツール――特にネットワーク型エディタを使ってトピックを読み書きしていくことです。ツール上にワークスペースやプロジェクトといった実際の場所（**会場** と呼びます）をつくります。

準備としてツール選定と会場作成が必要です。もう一つ、ツールの習熟も必要ですが割愛します。

ツール選定については文芸的タスク管理の章を参照してください。

会場作成については、ひとりで行うのならホームノートをつくるだけで事足ります。複数人で行う場合は、メンバー全員を招待した会場と権限まわりなどいくつかの設定が必要でしょう。Scrapbox の場合、プロジェクトをつくって、メンバーを招待して、必要なら管理者権限を付与して、としていきます。Scrapbox の導入や練習は拙作の『Scrapboxing』でも紹介していますが、内容が古いと思います。より最新は公式サイトなどを参照してください。また公式のサポート資料やヘルプも重宝します。

Step2: ETM を始める

Step1 にて会場までつくれたら、あとは実際に ETM を始めていくだけです。

会場を正しく準備できているなら、各自が好き勝手にノートを書いていけるはずです。複数人で行う場合、どのように秩序を保つかが論点となりますが、正解は無いため各自模索してください。この部分を解説すると、それだけで本が何冊もできてしまうため割愛します。ETM 以前に、ネットワーク型エディタ自体の作法やベストプラクティスについては Scrapbox まわりの資料やコミュニティが参考になります。上述した拙作でも取り上げています。

ETM に関する部分については、まずは本章でここまで取り上げてきた内容を運用してください。ことはじめとしては part 1 にてトピックベースでノートをつくること、トピックはブロックから成ること、トピックからブロックをつくったり、ブロックからトピックを導いたりすることを、part 2 にてトピックは広げたり詰めたり導いたりできること、part 3 にてロード、ターゲット、リソース、コンテキストといった管理を行うこと（会場に書かず個人的に管理すればいいものもあります）を扱ってきました。その他、細かいテクニックは次の part 5 で扱いますので、適宜ヒントにしてください。

注意点としては、**ゴールはこのステップでつくってください**。ゴールは次のゴール判定ステップでつくるのではなく、ここ Step2 でつくります。というのも、ゴールは最後の最後につくるものではなく、道中で導くものだからです。ゴールもまた仮説検証の対象であり、とりあえず掲げてみては修正して、と何度も書き換えたり増やしたり減らしたりします。

Step3: ゴール判定

探索期間が終わったら ETM も終わりです。ここまで探索してきた内容を踏まえて、どうするかを決める会議を行います。これを **ゴール判定** と呼びます。会議と書きましたが、真面目に打ち合わせをしても良いですし、合宿やパーティーの形でくつろいでも構いません。

時間としては半日以上、一日以内が良いでしょう。短すぎると満足がいくほど話せませんし、逆に長過ぎてもだらけてしまいます。探索はもう終わりましたので、従来どおり指向的なアプローチで、有限時間でばしっと意思決定するべきです。ETM の延長で行えるならそれでも構いません。

ゴール判定で行うのは以下 2 点です。

- ETM を続けるかどうか
- 今後何をするか

まずは ETM をここで終わらせるか、それともまだ続けるのかを決めます。続ける場合はまた Step1 から始めてください。

次に今後何をするかを決めてください。ETM としては方向性や方針を決める程度を想定しており、この時点ではあまり精度の高い計画や約束はしません。そちらはゴール判定が終わった後に、今まで通りのやり方で（つまりは指向的なアプローチで）やってください。ゴール判定自体は ETM の範疇であり、ETM はブースターを拡充するものでした。ゴール判定にて決めることもまたブースターの一つにすぎません。ですのでゴール判定の時点では「現時点ではよくわからない」「これから追々詰めていこうか」のような結論になることもよくあります。それはそれで構いません。ブースターが拡充されているならば、ETM としては成功なのです。

いずれにせよ、上記二点を行うときはゴールを使ってください。正しく ETM を行っているのなら、すでにゴールを導けているはずです。ですので通常は **ゴールというトピックを広げる・詰めることで上記 2 点を導くことになるはずです**。30 分以内で終わることもありますし、何なら会場にゴール判定トピックをつくって、そこで（会議をせずに）やりとりしておしまい、ということも十分ありえます。最後のゴール判定というと大きな出来事にしたくなりがちですが、上手く探索できているならその必要はなく、どうすればいいかは割と明示的です。逆を言うと、ゴール判定で荒れたり長期化したりするとしたら、そもそも ETM が上手く行ってないか、上手く行ってはいるが足りていません。後者の場合は、単に ETM を再開すれば事足ります。

と、このとおり、短時間で終われるし、何なら集まる必要さえありません。なのになぜ半日以

上、一日以内と書いたのでしょうか。これは **人が節目を意識する生き物** だからです。メリハリは大事です。ずっと探索ばかりだと息がつまりますし、探索の頭だと、この先行動していく上では支障が出ます。探索期間はもう終わったのだということを自らに教えねばなりません。そのために節目のイベントを設定して、ある程度長い時間を過ごすのです。しかし上述のとおり、ゴール判定として行うことは（上手く行ってるなら）すぐ済むはずですから、残りの時間は何しようか、となります。これについても 2 点あります。

- 1 打ち上げ
- 2 自由時間

1 の打ち上げは、プロジェクト終了時にもよく見られますが、その認識で合っています。一日以内と書きましたが、この目的であるならば数日以上でも構いません。探索は非常に疲れる営みで、一日で疲れを取って切り替えるのは難しいことがありますから、（探索期間にもよりますがたとえば）数日旅行するくらいしてもいいかなと思います。ただし旅行の場合、すぐに決めることは難しいでしょうから、探索期間の間に旅行トピックでもつくって練っておきましょう。ちなみに、次の part5 でも書きますが、こういう雑談的・遊び的な営みも適宜はさんだ方が ETM は上手くいきます。

2 の自由時間は、会議の場としてはすぐに解散して、各自自由時間を過ごすというものです。このとき、仕事を入れてはいけません。休むのも遊ぶのも自由です、各々が好きに過ごします。要はインターバルを個々に取らせる、というものです。こちらについても数日確保しても良いでしょう。

打ち上げと自由時間のどちらが良いかはメンバー次第です。通常は打ち上げしたいメンバーとひとりで自由に過ごしたいメンバーに分かれますので、どちらもサポートする（打ち上げを自由参加・途中入退場アリにする）のがおすすめです。

動的な ETM

ETM のワークフローは 3 ステップでした。別の言い方をすると、始まりと終わりがあります。しかし ETM をより動的に行っていく（日々に組み込んでいく）こともできますので、最後にこの点を取り上げます。

ETM にはワークスタイルのポテンシャルがあります。つまり普段の仕事の仕方として採用することができます。とはいえ、さすがにずっと No ABCD 的に進めるのは不可能でしょうから、探索モードと指向モードを用意して、探索モードのときに ETM を行うようにすると良いでしょう。

探索モード のときは ETM を行います。No ABCD も働くため管理は発生しません。**指向モード** のときは ETM は行いません。従来どおりのコミュニケーションや管理を使ってください。

重要なのは両者を同時に行わないことです。ETM は、No ABCD に基づいて全員が探索的に動くからこそ成立します。指向モードの混ざった人がいると ETM の秩序が乱れます。指向モードが必要な場合は、ETM からは抜けてください。あるいは少なくとも ETM を行っている間

は探索的に動いてください。ETM では「仕事として ETM に専念している（探索のみを行っている）」ことを期待します。忙しいから全然できてませんとか、重要な用事なので会議で伝えます皆さん参加してください等は通用しませんし、持ち込むべきではありません。探索期間の項でも書きましたが、指向モードの理を絶対に持ち込まないでください。

モードの切り替えにはいくつか戦略があります。

- チーム単位でモードを切り替える
  - チーム全員が探索モード or 指向モードになります
  - ひとりだけ指向モードになっている等は許しません
- ひとりひとりが必要に応じてモードを切り替える
  - 5人チームがあるとして、2人が探索モード、3人が指向モードで過ごしている等がありえます
  - 探索モードだった A さんが指向モードに移る、といったこともありえます
- 人ごとにモードを固定する
  - 常に探索モードで ETM を行っている人と、常に指向モードで動いている人が生じます
    - ちなみに現代以前の従来のあり方は後者 100% だということができます
  - 指向モードの人は探索モードの人を拘束せずにインプットを行う必要があります
    - 「探索結果を知りたいから会議開催して説明して」は通用しません
    - 指向モードの人が ETM の会場を読みに行くか、そこでやりとりを行う必要があります

正解はありませんが、合うやり方と合わないやり方があるので、合うやり方で進めましょう。いずれにせよ切り替えは疲れるため、なるべく少なくて済むあり方が良いです。

まとめ

- ETM のワークフロー
  - Step1: セットアップ
    - スコープ、探索範囲、メンバー、会場を設定します
  - Step2: ETM の実施
    - ゴールもこの段階で導きます
  - Step3: ゴール判定
    - ETM の再開要否と今後何するか（ざっくりで良い）を決めます
    - 節目として過ごすことも想定しており、数日以内で打ち上げか自由時間に充てるのも良いです
- 探索モードと指向モードを切り替える、と捉えれば ETM をもっと動的に取り入れることも可

ETM ことはじめ5 ～やり方や考え方のヒント集～

紙面の都合上、次章にて扱いますのでそちらを参照してください。

おわりに

高度なタスク管理としてプレーンテキストによるタスク管理（PTTM）、文芸的タスク管理（LTM）、そして本章にて探索的タスク管理（ETM）を紹介しました。

指向的・探索的アプローチの話から始まって、従来のタスク管理とは全く異なるあり方を整理してきましたが、無論この ETM が探索的な状況における唯一解ではありません、本章で挙げたこと以外のやり方や考え方もあるでしょう。特に本章では No ABCD など大胆な原則を定めており、誰もがすぐに適用できるほどかんたんで利便性の高いものではありません。

それでも従来成す術がなかった探索の領域に、新たな可能性を提示するものと思います。

=== Chapter-26 探索的タスク管理（ヒント集）===

前章では探索的タスク管理こと ETM の基礎編を解説しました。本章では参考にできそうなやり方や考え方のヒントを紹介していきます。

※本来は前章のことはじめ part5 で扱う想定でしたが、本書のプラットフォーム Zenn 側の仕様で 50000 文字以上書けないため、このような形で分けてあります。

ETM ことはじめ5 ～やり方や考え方のヒント集～

ここ part5 では参考にできそうなやり方や考え方のヒントを雑多に紹介します。ゼロから試行錯誤したり既存事例を探したりするよりも近道できるはずです。逆に「どうにもノリが合わないな」と感じたら、自分達で模索した方が良いでしょう。これらはあくまでヒントであって正解ではありません。

まずは文芸的タスク管理を

Ans: ETM は文芸的タスク管理をベースにしているので、その諸々が使えます。

基本的なテクニックは文芸的タスク管理の章も参照してください。ETM はネットワーク型のノートツールを使うため、まさにこれを使う文芸的タスク管理は参考になります。というより、ETM は文芸的タスク管理の上に探索的なアプローチを重ねて再構築したものです。前章冒頭にて前提としていると書いたのもそのためです。

スクイーズアウト

Ans: 有限化には「今の自分で 100% ひねり出した」との目安が使いやすいです。

ETM では「見えてきたら」を一つの合図にしています。最終的に求めたいゴールもここまでのトピック達を踏まえて「見えてきたこと」ですし、トピックもある意味「見えてきたこと」を言語化してつくります。

一方で、いつも見えてくるとは限りませんし、見えてこないこともざらにあります。特に見えてくるほどリソースを費やせない場面は非常に多いです。極端な話、アイデアが 100 個あるとして、この全部に対して見えてくるまで費やせるかというと不可能です。

探索的な ETM において正解はないため、通常はどこかで自分の意思決定をもって中断しなければなりません（有限化）。もちろん、テキトーに中断しても納得などできません。比較的上手くいく、ざっくりした目安してスクイーズアウトが使えます。

スクイーズアウト（**Squeeze Out**）とは、現状の自分で 100% ひねり出すことを指します。原義は「絞り出す」であり、果物を握りつぶして絞り出すイメージです――と書くと、死にものぐるいなニュアンスを抱くかもしれませんが、そうではありません。「現状の自分で」なので、今の実力や調子やモチベーションも反映できます。台詞でいうなら、たとえば「今の自分で出し切れるのはこんなところかな～」「これ以上は出なくなったんでいったんここまで」「飽きたんでこれくらいで」となります。

「出し切る」とか「出なくなった」といった言い方をしていることに注目してください。これは目に見える成果やかけた時間は関係無く、現状の自分がどれだけ出し切れるかを強調しています。筆者は「胸を張れるか？」との自問自答をよく使います。他を犠牲にして献身する必要はありませんが、現状の自分の範囲で 100% くらい絞り出したぞと言えそうかを問います。ベストエフォートと同義かと思われがちですが、少し違います。ベストエフォートは「まあ最大限頑張りますね」と言っているにすぎず情報量はありませんが、スクイーズアウトは出し切ったかどうかを見ます。主観でしかないのですが、端的な目安です。最初のうちは自分で自分の出し切り感を評価しづらいと思いますが、慣れてくると限界や傾向がわかってきて評価できるようになってきま

す。

スクイーズアウトは、ETM においてはかなり重要です。というのも、スクイーズアウトができているのならば（すでに出し切っているはずなので）納得感が高いですし、仮に出し切った上でも上手くいかなかったのであればインプットが足りないとか、何か見落としているとかいった形で疑うこともできるからです。すでに出し切っているので、これ以上自分で何とかすることはできない、意味がないのだと胸を張れるようになります。胸を張れると意思決定が加速します。つまり**スクイーズアウトにより意思決定が加速します。**

ブロックライティングの話

Ans: ブロックベースでトピックをつくっていく手法を取り上げます。

本節ではトピックを育てていくのに便利な書き方の紹介します。CBS という単純なやり方に始まり、ブロックライティングの話へと広げていきます。

CBS（Consideration Breakdown Structure）

Ans: CBS はスクイーズアウトする際に使える発散収束方法の一つです。テキスト + 箇条書きで行えます。

ことはじめの part3 でも書きましたが、ETM は知的生産的な営みです。情報を出す発散と、それらを整理する収束を繰り返し行っていくことで本質を導こうとします。part3 ではトピックを広げる、詰める、導くとの言い方で説明しました。

知的生産の手法は様々存在します。カジュアルな意味でのブレストは発散を行うものとしてよく知られていますし、体系化されたものには KJ 法があります。書き方としてはフリーライティングやマインドマップが知られている他、Miro など無限キャンバスツールもあります。SCAMPER やオズボーンのチェックリストといったトリガーリストもあります。

本項で紹介する CBS は、書き方にあたるものです。名前は WBS（Work Breakdown Structure）から取っていますが、WBS のように階層的な網羅性は意識しません。

CBS（Consideration Breakdown Structure） とは、検討（Consideration）をブレイクダウンしつつ構造化しながら記述していくことで知的生産を支援する手法です。やり方はシンプルで、次のようなフォーマットを使うだけです。

以下のように行指向的に書きます
インデントは 1-space を使います

CBSとは
 Consideration Breakdown Structure の意
 検討のブレイクダウンと構造化を支援する知的生産手法
 メリット:
  数少ない知的生産手法の一つであること(知的生産のハードルを下げること)
  テキストで完結できるため、テキストエディタなどで素早く言語的に書けること
   またノートツール全般に適用できること
  Scrapboxのフォーマットや文化と親和性があり、知っている人はすぐに使えること
 デメリット:
  行指向、インデント、テキストといった世界観に慣れるハードル
  イメージや空間を使えない(使わない)ため、それらを用いて思考する人には不便
  通常の文章以上に、他人に読んで理解してもらうのが難しいこと

一般化すると次のようになります。

(検討1)
 (検討1の内容)
 (検討1の内容)
 ...

(検討2)
 (検討2の内容)
 (検討2の内容)
 ...

ちなみに検討(Consideration)とは「タイトルのついた情報の塊」くらいの意味です。検討の名のとおり、自分で思考して言語化したものを想定していますが、思考と言えないほどの羅列や思いつき、また単に他者や他所の情報を集めて並べて名前をつけたもの等も含めても構いません。タイトルとありますが、ただの一言コメントや感想でも構いません。しかし、必ずタイトルは含まれている必要があります。一方で、一時的に、あるいは良いのが浮かばないのであえて無題を設定することも可能ですが、ここを許容するとただの情報保存場になってしまい知的生産ではなくなってしまいます。雑でも少量でもいいので、必ず自分の言葉で、少なくともタイトルはつけることを意識してください。

以下、CBS の書き方の説明を少し続けます。

インデントは 2-space 以上も可能です。

```
indent0(1つの検討を示す部分)
 indent1
  indent2
   indent3
    indent4
     ...
```

また、インデント配下は緩い包含を示します。厳密な包含関係や論理構造は必要なく、主観で関係があると思ったものを突っ込むだけで OK です。要はただのグルーピングです。

```
インデント配下は緩い包含を示します
 (内容1)
  ここにぶら下がった内容は、内容1に関するもの、という意味になります
  ここにぶら下がった内容は、内容1に関するもの、という意味になります
  さらに細分化もできます
  (内容a)
   ここは内容aに関するものが書いてあります
   ここは内容aに関するものが書いてあります
  ここは内容aに関するものではありません(内容1に関するものではあります)
 ここは内容1に関するものではありません(内容1の配下にはありません)
```

次に空行についてですが、空行無しで繋がった部分が「1つ分の検討」を示します。これを **検討ブロック** と呼びます。ですので、検討内容がたとえば 4 個あるのなら、以下のように 3 つの空行で区切られて、検討ブロックが 4 つ並ぶはずです。

```
(検討1)
 検討1の内容...

(検討2)
 検討2の内容...

(検討3)
 検討3の内容...

(検討4)
 検討4の内容...
```

空行は 2 行以上でも構いません。使い分けも自由です。たとえば「直近見ていない古い部分」をまとめて退避するために、3〜5行(あるいはそれ以上)の空行を空けたりできます。あるいは --- のような区切り線を入れても構いません。以下は 2 行ずつ + 区切り線で空けている例です。

（今はこの辺の検討をやっている）
  内容...

（今はこの辺の検討をやっている）
  内容...

（こっちは古い内容）
  内容...

（こっちは古い内容）
  内容...

（こっちは古い内容）
  内容...

空行や区切り線の使い方には好みが出ます ので、自分の好みを反映して構いません。ただし複数人で ETM を行う場合は（自分ひとりが書く備忘的なページは好きにしていいとしてそれ以外は基本的に）統一した方が良いです。

検討ブロックの単位ですが、ここは本当に自由です。自分にとってわかりやすい単位をつくればよいです。先にゼロインデントで検討名（タイトル）を適当に書いた後、その中身を書いていくこともできますし、逆にたくさん書いた行を見ながら「これとこれはくくれる」「これは同じこと言ってるよな」「この辺はよくわからんからいったん未整理で束ねるか」などと判断しながら構造化していくこともできます。

検討ブロックや各行の並び順も自由です、かつ **順序性はありません**。もし順序に意味があるのなら、その旨を明示してください。筆者は冒頭に数字を振る書き方をよく使います。ただし数字についても、単に「n 個存在しますよ」なのか「順番にも意味がありますよ」なのかがわからないので、その点は各自調整してください。ひとりで ETM を行っている場合も、未来の自分が汲み取れるとは限らないのでなるべく配慮した方がいいでしょう。

以下、典型的な書き方の例を示します。

まとめ

検討3をさらに掘るかどうかってところかなー

1 まず検討4にてスクイーズアウトして、検討2と3が出た

2 検討3をさらに深堀りして、こっちはダメそうとわかった

ただスクイーズアウトできるほど粘ってはいない

なにかありそうな気はするけど、いったん諦めてる

3 検討2は早々にダメってのは論文見つかって判明した

4 検討1では有識者や関係者の声を列挙してるが、目新しいことはなし

（検討1）

内容...

（検討2）

内容...

（検討3）

内容...

（検討4）

内容...

いかがでしょうか。「まとめ」ブロックを見ると、どの検討ブロックがどういう風に流れていったのかが分かると思います。もちろん各検討の詳細は、そちらの検討ブロックを見ればわかります。

CBS の解説は以上です。細かいテクニックは他にも色々ありますが、単に 1-space インデントの箇条書きで、空行区切りで検討ブロックを書いていくというシンプルな手法です。

サマリー駆動

Ans: 適宜サマリーを切りながら仕切り直していく探索スタイルです。

CBS に関して、一つ重要な点を挙げるとしたら、書き込みが増えてくると収拾がつかなくなってくることです。これに対処するためには、どこかで大胆にまとめる必要があります。たとえば、きりのいいところで「この辺の検討ブロック達からはこういうことが言えます」のような要約または蒸留を**まとめブロックをつくる形で** 行い、以後そのまとめブロックをインプットにして、もう一度検討ブロックをゼロからつくる――のようなことをしたらどうでしょうか。収拾をつけつつ、先に進められる気がしませんか。

言ってしまえば意思決定の言語化にすぎませんが、これをしないと書き込み量の多さに収拾がつかなくなって破綻します。特に次に再開するときに思い出すのが絶望的になってしまいます。この点はコンテキストの管理でも述べました。まとめブロックをしっかりつくるのがきつい場合でも、最低でも高頻度で着手し続けるコンテキストメンテナンスか、再開時にすぐ思い出せるよう軽い要点を書いておくコンテキスト駆動くらいは行っておきたいところです。

このように、要所要所でまとめブロックをつくる → それをインプットにして仕切り直す、を行っていく営みを **サマリー駆動** と呼びます。サマリーの言葉がわかりづらければチェックポイントやセーブポイントだと考えてください。ETM においては重要な概念です。認知コストの限界とコンテキストスイッチングの壁を超えるために、面倒でもサマリーをつくって絞っていく必要があるわけです。前項では CBS を紹介しましたし、他の手法も紹介しますが、最も重要なのがこのサマリー駆動です。これと同等のことができるのなら、手法は重要ではありません。

下から上に書く

Ans: 通常テキストは上から下に書いていきますが、CBS は下から上にブロックを置いていくこともできます。

私たちは通常、上から下へとテキストを書いていきますが、CBS では逆方向も可能です。以下は検討1が古く、検討3が新しいものとして、下から上に書いた例です。

```
（検討3）
 内容...

（検討2）
 内容...

（検討1）
 内容...
```

直感的ではないかもしれませんが慣れの問題です。新しい順に並んでいると考えれば、それほど違和感は無いと思います。主なメリットは書くときにスクロールしなくて済むことと、読む時も結論や最新の状況から読み進めていけることです。

ただし上から下に書いていくスタイルもあります。つまり CBS では **上から下なのか、下から上なのか、と二通りの方向が想定されます**。方向がどちらかであるかは通常読み取れますが、たまにわからない・わかりづらいことがあります。書く人がわかりやすく補足を入れるか、見てわかるよう設計（検討ブロック名に数字を振るなど）するべきです。

ブロックライティング

Ans: CBS の手法を一般化したものです。検討とまとめ以外のブロックも使うため、より汎用的です。

CBS では検討ブロックをつくることと、最後にまとめブロックをつくることの二点が特徴でした。これをより一般化したものが **ブロックライティング（Block Writing）** です。CBS はブロックライティングの言葉で言うなら、検討ブロックを自由に使い、サマリーブロックは最後に一回だけ使ったものだと言えます。ブロックライティングをかなりシンプルにしたものです。

ブロックライティングの基本的な書き方やフォーマットは、CBSと同じですのでそちらの項を参照してください。ここではブロックライティングがサポートするブロックの種類に主軸を置いて説明します。

まず **ブロック** とは、空行で区切られた箇条書きの塊を指します。CBS ではブロック≒検討ブロックでしたが、ブロックライティングでは他にも様々な種類があります。

ブロックライティングの目的はサマリーを導くことです。CBS では最後にまとめを書きましたが、これと同じです。「ここまでのまとめ」をブロックとして書きます。これを **サマリーブロック** と呼びます。さらに、サマリーブロックは 1 個とは限りません。CBS では最後にまとめを書いて終わりでしたが、ブロックライティングではサマリーブロックをつくった後、それを踏まえてまた続けることができます。つまり前項のサマリー駆動ですね。

これで大筋は述べました。あとはどんな種類のブロックがあって、どう使っていくかの話となります。

先にブロックの種類を示すアルファベットについて解説します。**ブロックレター**、あるいは単にレターと呼びます。CBS ではインデント無しの行には（タイトルとしてたとえば）検討の名前や結果などを書くだけでしたが、ブロックライティングでは多数のブロックが存在するため種類がわからないと後で読みづらいです。そこでレターの形で指定します。**指定は必須ではありませんが、あるのとないのとでは後で読む際の認知コストがかなり違うので指定を推奨します。**すでに登場したサマリーブロックは s です。Summary から取っています。

ここまでのまとめ:

- s サマリーブロック(Summary)

種類の解説に入っていきましょう。最も基本となる **ABCD ブロック** から説明します。これは Action（タスク）、Brast（ブレスト, ブレインストーミング）、Concept（概念）、Direction（方向性）の略です。

ブロックライティングでは、まず **ブレストブロック** を使って発散させていきます。スクイーズアウト（できるだけ頭を絞り切る）するつもりで出し切ります。この過程で収束ができそうなら行っても構いませんし、深堀りしたいテーマが見つかったらそちらに入っても構いません（後述する検討ブロックを使います）。ブレストブロックは一度のブレストで終わらせる必要はありません。たとえば 5 日後にちょっと足すことも可能です。ただし、終わりはないので適当なところで切り上げます。

発散（とついでに行えるなら収束）を経ていると色々と見えてきます。新しい概念や用語を導入した方が上手くいくと感じたならば、それを **概念ブロック** で整理します。ここで整理した言葉は、以後使うようにしてください。逆に使わない、あるいは使えないのであれば概念ではないかもしれません。概念というよりは「こんな感じが良さそう」など方針や方向性に絡む何かが見えてきた場合は **方向性ブロック** で整理します。

もう一つ、ABCD ではないですが、**検討ブロック** もよく使います。ブレストブロックはテーマ未定で何でも発散しますが、検討ブロックは特定のテーマ（お題）を定めて、それについてブレストします。ブレストも検討も行っていることは同じで、発散的な営み＆収束も適宜行っていくものです。検討とブレストを厳密に区別することに意味はないので、あまり深く考えなくても結構です。ブレストの最中に一つのテーマを深堀りしに行くこともありますし、逆にあるテーマの検討中に色々浮かんできて実質ブレストになっていることもよくあります。筆者は検討ブロックを使うことが多いです。検討ブロックは k で記述します。

最後に、やることが見えてきたらタスクブロックを使います。レターの a はアクションから来ていますが、意味的にはタスクです。タスクブロックはタスクリストや TDOO リストと同等です。特に具体的に実践可能なものを指します。なので行ごとに □ などをつけて区別できるはずです。実際、どこまで終わってるかを可視化したいので区別は重要です。とはいえ、いちいちつけるのも面倒くさいので、運用には好みが出ます。

ここまでを踏まえると、一連の流れを定義できます。ブレストや検討で発散して、概念や方向性やタスクで肉付けしつつ、きりのいいところでサマリーをつくる――これを **サイクル** と呼びます。ブロックライティングはサイクルを何回でも繰り返せますが、通常終わりはないので、どこで切り上げるかは自分で選びます。もう一つ、次のサイクルを開始する際は、なるべく前回のサイクルの サマリー **だけを** インプットにしてください。ブロック全部をいちいち見ていてはきりがないためです。本質をサマリーブロック内にコンパクトにまとめることを意識してください。

ここまでのまとめ:

- ブロックの種類
  - s サマリーブロック(Summary)
  - ===
  - a タスクブロック（Action）
  - b ブレストブロック（Brast, Brain Storming）
  - c 概念ブロック（Concept）
  - d 方向性ブロック（Direction）
  - k 検討ブロック（検討, Kento から k です）
- ワークフロー
  - 以下で 1 サイクル
    - 1: b や k で発散していく
    - 2: c や d で概念や方向性をつくっていく
    - 2: やることが見えてきたら a でタスクリストを書く

        - 3: きりのいいところで s でサマリーをつくる
    - サイクルは n 回繰り返すが、どこで切り上げるかは自分次第
    - 次のサイクルを行う場合は、前回のサマリーをインプットにする

これだけでもブロックライティングは十分に行えますが、他にも便利なブロックがいくつかありますので紹介します。

用語ブロック（g, Glossary） は、新しい用語が必要そうなときに使うと便利です。概念ブロックとの使い分けが悩ましいですが、用語とは「以後その言葉を実際に使う」ことを前提とするものです。トピックを見ている読者も用語ブロックに書いてある用語は「このノート内で使われている」と捉えます。名前のとおり、用語集です。一方、概念ブロックは「こういう概念を定義したら便利だよね」「まだあまり言語化できないけど、こんな感じの概念があるよね」といったことを言語化して名前をつけて捉えるのに適しています。用語になるとは限りませんが、おそらく重要なものではあります（重要だからこそわざわざ名前をつけて概念化しています）。とはいえ、使い分けは細かい話です。適当で構いません。あるいはサマリーブロックの中で解説してしまう手もあります（筆者もそうしがちです）。用語ブロックは、どちらかというとサマリーをつくる前に読む人が主要用語を知るためのブロック、と言えるかもしれません。

参考情報ブロック（r, Reference） は、主観と事実でいう事実の方を集めたブロックです。何らかの URL だったり、書籍や文書の名前だったり、あるいは人の発言だったりします。これらは検討時の制約として踏まえることが多いので、ブロックライティングでも扱います。とはいえ、バカ真面目に書いていてはただの勉強ノートになってしまいますので、重要な部分のみ取り上げます。特に引用や抜粋をコンパクトに行うのがコツです。一つ例を挙げます。異文化理解力の「日本は階層主義で合意形成的」との文化をトピックにした場合です。

s
 異文化理解力の用語
  階層主義とは、.....
  合意形成的とは、.....
 日本は階層主義と合意形成的が両方当てはまる唯一の国
 象徴的な文化が「稟議」

k コメントエリア
 Aさん
  ...
 Bさん
  ...
 Cさん
  ...

---

s
 勉強して考察と議論して理解が一致した

□k 本文読んで勉強する 引用ベース
 ...

□k 考察と議論
 ...

r
 エリン・メイヤーの異文化理解力: https://www.amazon.co.jp/dp/B013WB5BJS/
 [中国人やインド人が、すぐにちゃぶ台返しをする理由:日経ビジネス電子版
https://business.nikkei.com/atcl/seminar/20/00046/081700004/]
  ※ちなみにこれは Scrapbox の記法です

参考情報ブロックは最後の r の部分です。

※良い機会なので、ここまでの書き方も入れています。これを見ると、まず 1 サイクル目で理解を一致させて、2 サイクル目で改めてまとめをつくっており、検討ブロックでコメントエリアも置いています。参考情報ブロックは 1 サイクル目の方にあります。またサイクル間には --- で区切りを入れています。なくてもいいですが、あると見易いです。ちなみにサイクルは「上から下に」並んでいます。--- の下が 1 サイクル目で、上が 2 サイクル目です。

文脈ブロック は、開始時や現在の文脈をまとめたブロックです。前章のコンテキスト駆動にあたります。レターは cx を使います。開始時の文脈なら cxo、それ以降はきりのいいところで現在

の文脈を cx1, cx2 のように番号を増やす形で書いていくと良いでしょう。目安としては、1-サイクル 1-文脈ブロックです。あるいは cx0 以外はサマリーブロックに書いてしまっても良いでしょう（筆者もよくやります）。

最も重要なのは cx0 で、開始時の文脈を書きます。「このトピックを検討しているのはなぜなのか」「なぜ起きたのか」といった背景や事情や思いなどを書いておくわけです。すると検討が煮詰まったときに原点に立ち返ることができます。やっぱり重要だからやろうと判断できたり、逆に別に要らないよなーと中断や放棄の判断もできます。開始時の文脈がわからないと、捨てていいかがわからないのでいつまでも気にすることになり疲弊します、あるいはギャンブルのようにランダムに捨てることしかできず後で（もっと重視して取り組むべきだったのにしてなかったせいで）後悔するかもしれません。

例を挙げます。以下はひとりで ETM をしているとして「引っ越ししたい」トピックです。

```
cx1
  色々検討始めてるけど、選択肢多すぎてなんかだるいなーってきもち


r  友達に聞いてみる、一言でまとめる
  友達1「...」
  友達2「...」
  ...

□k  都会？田舎？
  徒歩で生活できるならどこでもいい
  徒歩片道20分くらいなら許容、チャリもあるし


k  どんな家が欲しい？
  ネットは必須、動画見たいゲームしたい
  1K  25㎡は狭すぎ、もう一部屋広いの欲しい
    KとかワンルームとかLDK？とかわからんので調べる  →  [間取りの一覧]
  最上階、上の足音うるさいの本当にむり


k  郊外の選択肢を調べる
  隣接する県
    神奈川  埼玉  千葉  □山梨
    23区なので山梨はない
  路線を調べる
    ...


cx0
  都内の家賃キツイので引っ越したい、郊外？
```

検討ブロックを立てて色々検討しています。見ての通り、軽い気持ちから始めたものなので検討はしばらく続くでしょう。都会？田舎？については結論が出てます（ので□をつけて完了を示してます）が、全体としてどう転がるかはまだわかりません。ここで重要となるのが、cx0 として書いた開始時の文脈です。ここでは「家賃がキツイから始めた」と読み取れます。また現在の文脈 cx1 ではダルさを吐露しています。

このトピックは、おそらく cx0 を読み返して「問題の本質は家賃だ」と気付ければ、家賃を下げるまたは収入を上げるための検討に絞れるでしょう。もしかすると「転職する」「副業を考えてみる」のようなトピックに派生するかもしれません。逆に cx0 を書いてなかったとしたら、家賃という本質を思い返せず迷走するでしょう。

まとめ:

- ブロックの種類
  - s サマリーブロック(Summary)
  - ===
  - a タスクブロック（Action）
  - b ブレストブロック（Brast, Brain Storming）
  - c 概念ブロック（Concept）
  - d 方向性ブロック（Direction）
  - k 検討ブロック（検討, Kento から k です）
  - ===
  - cx 文脈ブロック（ConteXt）
    - 開始時の cx0 はできれば書きたい
    - それ以降の cx1, cx2... はサイクルごとに1つで良い、またサマリーに書いても良い
- ワークフロー
  - 以下で 1 サイクル
    - 1: b や k で発散していく
    - 2: c や d で概念や方向性をつくっていく
    - 2: やることが見えてきたら a でタスクリストを書く
    - 3: きりのいいところで s でサマリーをつくる
  - サイクルは n 回繰り返すが、どこで切り上げるかは自分次第
  - 次のサイクルを行う場合は、前回のサマリーをインプットにする
- その他
  - 完了したブロックには□などをつけるとわかりやすいが、律儀な運用は面倒。好みが出る
  - サイクルなど区切りを書くと視覚的にわかりやすい
    - --- や ===、その他何でも良い
    - 文字数も問わないと多すぎると逆にうるさくなる、筆者的には 1〜4 くらいがおすすめ

レターをカスタマイズする

Ans: 必要に応じてレターをつくったり変えたりしても構いません。

ブロックライティングはカスタマイズしやすい手法でもあります。本質的には色んな種類のブロックを並べているだけであり、ブロックの種類も上述したものがすべてではありません。他にも便利なものがあれば適宜定義しても構いませんし、レターについても自分が使いやすいもので構いません。

たとえば筆者は文脈ブロックは cx0、cx1 などは使わず、開始時の文脈を cx で記載しています。これ以外のサイクルごとの文脈はサマリーブロックに書くか、あるいは書きません（サマリーがあれば事足りるため）。他にもエンジニアの立場として「具体的にどうつくるか」を考えるときに実装ブロック（Implementation, レターは i）を使ったり、サイクルの名前をつけるのにタイトルブロック（Title, レターは t）を使ったりもします。もちろん、タスクブロックのレターが a なのはわかりづらいから t にしよう、といった改造も可能です。

ただし複数人で ETM をしている場合、メンバーによって認識が違うと混乱するので統一はしましょう。「レター一覧」のようなトピックをつくってメンテナンスすると便利です。

下から上に書く part2

Ans: ブロックライティングでも下から上にブロックを並べる書き方はよく使います。

少し前に CBS では下から上に書くこともあると書きましたが、ブロックライティングも同様です。さらにサイクルごとに異なる場合があります。例を示します。

cx1
  2サイクル回した、もう時間ないので次で決定したい

---

s
 ...

k
 ...

b
 ...

---

cx0
 ...

b
 ...

k
 ...

k
 ...

s
 ...

これは以下の構造になっていると読み取れ、

3サイクル目

2サイクル目

1サイクル目

かつ、1サイクル目は上から下に書いており（cx → ブレスト → 検討 → 検討 → サマリー）、2サイクル目は下から上に書いている（ブレスト → 検討 → サマリー）ように読み取れます。あくまでも例です。実際は人によってクセが出ます。方向の読み取りが負担になるようなら、一言補足しておくと良いでしょう。

理想を言えば、そもそも読む方向はあまり重要ではありません。ブロックライティングはブロックを並べることで本質を導くものであり、ブロックの読み方や順番自体にさほど意味はありません。書くときや読むとき（特に書くとき）にちょっとやりやすいかな、というくらいです。それよりもサイクルの区切りとサマリーの内容の方がはるかに大事です。

個人と協調の線引き

Ans: 複数人で ETM を行う場合、各個人の書き込みを上手く尊重できないと居心地が悪くなり形骸化します。

どのように尊重していくかを解説します。

トピックは個人でつくる

Ans: トピックは誰かひとりがつくってから、それを他の人に見てもらうのが良いです。

複数人で ETM に取り組む際、トピックは誰か一人で率先してつくります。従来のメンタルモデルだと最初から皆と一緒に打ち合わせ等で話し合いながら進めていくことが多いですが、この発想は ETM では捨ててください。むしろ、一人でつくってそれをレビューしてもらうスタイルが合っています。前章にて当事者は可能な限り一人が良いことを説明しましたが、当事者が出る前の、単にトピックを自由に書いていい段階（3T でいう Topic の段階）であってもこの原則は有効なのです。というのも、トピックそのものにれっきとした正解が無いことが多く、最初から協調すると十中八九こじれてしまうためです。誰か一人がとりあえず形にしてみて、それを皆で好き勝手にこねる方が良いのです。

たとえばAさんが書いたトピックT1が気に入らないBさんがいるとします。従来のメンタルモデルだとT1を直したり直してもらうよう議論したくなりますが、そうはしません。代わりにBさんがT2をつくって、それをAさんに見せるようにします。Aさんもまた、Bさんの T2 を直接修正しようとはせず、自分の T1 を修正します。**個人の領分を尊重しながら**、お互いに言及し合ったり見せ合ったりするイメージです。この例では領分の単位はトピックですが、ブロックや行の場合もあります。

パーソナルスペース

Ans: 個人の領分をパーソナルスペースと呼びます。比較的上手く共存できて、理解もしやすい捉え方としてレイヤーがあります。

前項で言及した個人の領分を **パーソナルスペース** と呼びます。

パーソナルスペースには以下の種類があり、レイヤーと呼びます。

レイヤー	名前
4	会場
3	トピック
2	ブロック
1	行

前項ではトピック（つまりレイヤー3）の例を示しましたが、これより大きな単位（レイヤー4の会場）もあれば、ブロックや行といった小さな単位もあります。

どのレイヤーを領分と感じるかはその人次第、その状況次第で変わります。いずれにせよパーソナルスペースを適切に取り、また尊重することは ETM を継続する上で重要です。ここが犯されてしまうと、まともに書けなくなってしまい形骸化の一途を辿ります。

とはいえ、いきなり尊重しきるのは不可能ですから、通常は ETM を行いながらすり合わせていくことになります。たとえばAさんはパーソナルスペースが広めだがBさんはかなり狭いなど人によっても違ってきますし、チーム全体としても傾向が出てきます。そういったことを暗黙のうちに心がけていくか、それが厳しいならガイドラインを書いて明文化します。筆者の体感としては、日本はハイコンテキストであり「言わないけど我慢してる」「無視してる」ことが多いので、率直な議論はしてみた方が良いと感じます。そうしないと、パーソナルスペース尊重派がマジョリティとなってあまり書き込まれなくなるか、逆に遠慮なく書き込む派がズカズカ踏み込んできて前述のトピックは個人でつくるが発生しにくくなります。前者は想像しやすいですが、後者もあるあるで、たとえるなら会議で空中戦になって何も決まらず終わるようなものです。

パーソナルスペースの各レイヤー

各レイヤーの詳細も見ていきましょう。

レイヤー	名前
4	会場
3	トピック
2	ブロック
1	行

4 会場レイヤー

「ひとりで ETM を行っている会場」そのものがその人の領分になっている状態です。

たとえば複数人で ETM を行っているメインの会場とは別に、各個人が自分用の会場を別途（あるいは普段用として）つくる場合などが当てはまります。この会場は通常非公開で運用されがちですが、公開またはメンバーだけに限定公開することもできます。

ETM においては非推奨です。ETM をやるならメイン会場に書き込んでください。会場レイヤーのパーソナルスペースが乱立すると ETM は成立しなくなります。ETM をやるからには、メイン会場一つだけをつくって、皆がここに書き込むことは絶対です。メイン会場内だけでも成立させるのは大変なのに、会場が分かれてしまっては絶望的です。

会場レイヤーのパーソナルスペースの主な用途は以下のとおりです。

- メイン会場に書けないか、書いても明らかに他のメンバーの役には立たない「個人的すぎる」内容を書く
  - 他人が読んでもわからないメモや思考の断片
  - 個人情報などセンシティブな情報
  - 他のメンバーには参考にならないと思われるレベルの備忘録や下書き etc
- 自分の情報発信手段として運用する
  - 文章を公開するブログは知られていますが、その発展系として個人の ETM を公開する例があります
  - Scrapbox 界隈で稀に見られます（※1）
    - ただのブログと違うのは、探索的なタスク管理の過程も公開していることです（※2）

いずれにせよ他のメンバーに読んでもらうことは期待できません。少なくともメイン会場からリンクを張る必要があります。

- ※
  - 1 本書で紹介できそうな、最もわかりやすい例はHelpfeel社のCosenseを一部公開でしょう。ただし ETM 的ではなく、単にコミュニケーション手段・ナレッジ共有手段として使っている側面が強いですが、会場のイメージは掴めます

- 2 ただのブログや備忘録として使っているものはカウントしません。あくまでも ETM または同等の文章を公開しているものをカウントします。2024 年 8 月時点では珍しいため、稀にと書きました。筆者が知らないだけで、Scrapbox に限らず存在しているかもしれません。たとえば Obsidian などのノートツールでバリバリひとり ETM を行っていて、それを静的サイトや GitHub などで公開している人くらいはいるかもしれません

3 トピックレイヤー

ETM では頻出するレイヤーです。詳細はトピックは個人でつくるを参照してください。

2 ブロックレイヤー

おそらく ETM において最頻出するレイヤーです。トピック、つまりは 1 つのノート内に n 人のパーソナルスペースが存在する形となります。別の言い方をするとブロック単位のパーソナルスペースであり、A さんのブロック1、B さんのブロック1、A さんのブロック2――といった形で各メンバーのパーソナルスペースが並ぶことになります。

各自が好き勝手に書き込めるので非常にやりやすいですが、ともすると「パーソナルスペースが並んでるだけ」でそれ以上発展しなくなる現象がよく起きます。ETM としては意思決定していきたいところなので、これは望ましくありません。これの対処として ETM では当事者などターゲットの管理（3T モデル）がありますが、別解としてパブリックスペースがあります。

以下、少し寄り道してパブリックスペースと表札の解説をはさみます。

パブリックスペース

パブリックスペース とは、その名のとおり誰でも入れるニュアンスです。かつ、パーソナルスペースとは違って誰かが領分を主張することはできません。あくまでも共有地であり、メンバー全員が自由に書き込める他、メンバー全員の利を優先します。いわゆる Wiki は、個人的な事情や無駄な情報を省いて客観的に皆に役立つナレッジを端的にまとめます（※1）が、そのイメージです。つまり **トピックにはパーソナルスペース的なブロックと、パブリックスペース的なブロックが並ぶことになります**。このハイブリッド戦略であればパブリックもパーソナルも両立できます。

- ※
  - 1 このあたりの議論としてわかりやすいのが、またもや Scrapbox の例になりますが自分のアイコンを中心にコメントを書くのを避ける - Cosense サポート資料だと思います。Scrapbox でもまさに（自分のアイコンを置いてその下にぶら下げる形で）パーソナルスペースを確保する文化があるのですが、それだとナレッジが育たないことに言及しています。

表札

ブロックレイヤー以下のパーソナルスペースを主張するときに使う文字や画像を **表札** と呼びます。

表札がないと、どのブロックが誰のスペースなのかがわかりません。古典的には（吉良野すた）のように名前を書いて示しますが、このやり方はいちいち書くのも面倒ですし視認性にも優れません。

良い例は Scrapbox です。アイコン記法が採用されており、Ctrl + i キーで自アバターの画像をアイコンとして挿入できます。

このような機能がない場合は、絵文字で代替します。たとえば筆者は□（うさぎ）をよく使います。よく使うので辞書登録などを使ってすぐに打てると便利です（筆者は「うさ」で登録）。何の絵文字を使うかは自由ですが、アバター画像と同様、誰であるかがわかるよう対応を取ってください。慣れないうちは対応を記載したノートをつくって、わかるようにしておくと良いでしょう。あとで変更するのは面倒くさいので、なるべく変更は想定せず使い続けるようにしてください。

表札はブロックレイヤー、または後述する行レイヤーで使えます。以下に例を示します。「ブレストって何？」トピックです。

ブレストという言葉の意味に乖離がある
皆の意見を聞きたい

- アイデア出し、くらいの認識
- 個人か集団かは関係ない
  - ただ集団でやるのは正直無理、人類には早すぎると思ってるので基本ひとりブレストをイメージする
    - 人類には早すぎると思う理由が気になる
    - こっちで → ［ブレストは人類には早すぎる］

- オズボーンが考案した会議手法じゃないの？
- ［ブレインストーミングの原点］

- 複数人で対面で集まって行うアイデア出し

ブロックレイヤーでは三人分の表札があります。また、□のパーソナルスペースには、□による行レイヤーの表札もあります。これにより、基本的に□のスペースだが、この行だけは□のスペースであることが見て取れます。

1 行レイヤー

レイヤーの解説に戻りましょう。行レイヤーとは行単位で設けるパーソナルスペースを指します。

具体的には単一行と複数行があります。上記の例を再載して説明します。

ブレストという言葉の意味に乖離がある
皆の意見を聞きたい

□
アイデア出し、くらいの認識
個人か集団かは関係ない
ただ集団でやるのは正直無理、人類には早すぎると思ってるので基本ひとりブレストをイメージする
□人類には早すぎると思う理由が気になる
（ここも□のパーソナルスペース）
（ここも□のパーソナルスペース）
・・・
□こっちで → [ブレストは人類には早すぎる]

まず□の表札で始まる「人類には早すぎると思う理由が気になる」の行は、行レイヤーのパーソナルスペースです。これは単一行です。この行の次行からインデントを入れてぶら下げたとすると、同様に□のパーソナルスペースになります。これが複数行です。ですので、□はいちいち表札を書かずにぶら下げることができます。一方、□以外の人は、表札を入れた方がいいでしょう（でないと□が発言したかのように見えてしまいます）。

意思決定はパブリックスペースで

Ans: パーソナルスペースばかりだとまとまりがないので、パブリックスペースをつくってまとめていきます。

ETM では多数のトピックを扱いますが、意思決定をはさんでいかないとなかなか終わりが見えません。特にパーソナルスペースばかりだと各自が意見を並べて終わり、となってそれ以上進みませんので、パブリックスペースをつくって（暫定的でもいいので）確定をしてください。このトピックではこの意見を採用するのだ、と確定するのです。

あるいはブロックレイヤーの項で言及したとおり、3T モデルを使って当事者を決めても構いません。当事者を決めるとは、そのトピックをその人のパーソナルスペースにすることを意味します。ただ完全にパーソナルだと他のメンバーが絡みづらいので、通常よりも読みやすさやツッコミのしやすさを確保してください。

もちろん、すべてのトピックでこれら意思決定を行うのは非現実的です。特に意見を聞いたり雑談をしたりする用のトピックではパーソナルスペースだけが並んでいることがよくあります。

個と和のバランス

Ans: 個人の尊重と全体の整合のバランスをどう模索していくのか、そのヒントを紹介します。

複数人で ETM を行う場合、基本的に個人を尊重するべきです。やり方や考え方が違っていても是正するのではなく容認します。気に入らなければ直接是正するのではなく持論を書いてそれを見せたり、単にスルーしたりします。一方で、パブリックスペースの話を前述したように、何らかの決定をして皆がそれに合わせることも必要です。このあたりのバランスを取るためのヒントをお伝えします。

興味ドリブン

Ans: 自分の興味関心に基づいて自然に動くと上手く共存しやすいです。

仕事や私生活では通常、あまり興味がないことや場合によっては嫌いなことであっても我慢して対応することがよくあります（義務ドリブン）が、ETM では望ましくありません。ETM は各人がトピックの形で残しながら自由に動くことによる化学反応を期待します。遊びのようなものです。従来の我慢は要りません。

それで整合性が取れるのか、と疑問に思われるかもしれませんが、意外と取れます。重要なトピックや皆が面白いと感じるトピックは自然と盛り上がるからです。そういう意味で、ETM では自分の興味関心に基づいて動くのが基本的には良いと考えます。これを **興味ドリブン** と呼びます。おそらく大半のトピックは自分ひとりか、多くてもう一人が反応するくらいの寂しいものになりますが、それでも構いません。1の成功には100の失敗があるという類の格言は多数ありますが、ETM も同じで、最終的に導くカードやゴールの下にはその何十倍何百倍もの寂しいトピックがあります。

何より興味ドリブンでなければモチベーションや精神が保ちません。ETM はただでさえ探索的な営みであり、正解がなくて孤独です。従来の指向的なアプローチにおいて義務ドリブンが成立するのは、正解があるからです。たとえ興味関心がなくても、正解に向かって進んでいけばいいという作業的な営みであれば人間は比較的耐えられます。ETM にはそれがないのです。

興味を引く

Ans: 他のメンバーに読んでもらう・書いてもらうために指示命令による強要はしません。代わりに、興味を引く書き方をして誘います。

他のメンバーに読んでほしかったり、コメントが欲しかったりするときはよくありますが、ETM では強要はできません。代わりに興味や関心、もっと言えば注意を引くことで誘導します。やりすぎるとうるさい宣伝のようにノイジーとなってしまいますが、様々なやり方があります。いくつか例を挙げます。

- 動線上で紹介する

    - 日付トピックで雑談的に取り上げるのは有効です
- 文芸的タスク管理のアイキャッチとエキサイトメント
- トピックの名前を平易でわかりやすくしたり、キャッチーにしたりする
- 皆がよく見に来るトピック T1 上で、読んでほしいトピック T2 を紹介する
    - T2 の見せ方が強引だと意味不明なので、T1 の問題解決方法として T2 が使えるなど自然な連携を心がけます
- ギブアンドテイク、というよりギブ
    - 他のメンバーの検討を支援したり、コメントを書いて助けたりすると、相手も助けてくれます
    - 情報は出す人のもとに集まるとも言います。ギブ重視で会場を肥やしてあげると存在感も増えて、他メンバーから注目されやすくなります
    - しかし他のメンバーに尽くしすぎると自分が疲弊しますので、興味ドリブンを逸脱しないようにします

要はアピールの仕方を工夫するという話です。すでに仕事や趣味などで実践している人も多いと思いますので、そのテクニックを使っても良いでしょう。ETM の他メンバーは仲間であると同時に、アーリーな顧客でもあるのです。

スルーを許容する

Ans: 無理してリアクションを取ったり返信を書いたりする必要はありません。

従来のあり方ですと、相手のことを考えてリアクションを増やしがちです。お礼や感謝を頻繁に差し込んだり、挨拶のみならず定型文を書く人も多いでしょう。しかし ETM では情報を並べる世界となるため、このようなリアクションはあまり自然ではありません。意識しなければ（あるいはよほど他人思いの人でなければ）従来よりも書く機会が減りますが、**それで構いません**。むしろこういったリアクションは情報量自体は皆無であり、ノイジーにもなりかねません。

一つの目安として **スルーを許容する** と良いでしょう。無理してリアクションをつける必要はないということです。チャットでは既読無視が問題視されたり、その息苦しさのせいで SNS 疲れが起きたりしますが、ETM ではありませんし、あってはなりません。これはもっと言えば、**リアクションが必ず来ることを前提にしてはいけない** とも言えます。

リアクションするな、と言っているわけではないことに注意してください。やりたい人ややってしまう人がいれば、それはそれで構いません。ですので、全体としては、たとえば「よくお礼を言う人」「たまに言う人」「ほとんど言わない人」などが混在したりします。よく言う人は言わない人を、言わない人はよく言う人を気持ち悪く感じたりしますが、そこは個性として許容しましょう。ETM では別に仲良くなったり関係を深めたりする必要などありません。トピックを皆で書いてゴールをひねりだせればそれで良いですし、それでも楽しいのです。そして楽しければ尊重と共存も進みます。

オタク注意報

Ans: 自由だからといって自分しか興味がないコンテンツをたくさん共有するのは控えましょう。

ETM では各人が自由にトピックを書きますし、雑談的な営みもできます。この快適な状況下では、たまにまるでオタクの早口語りのようにたくさんトピックをつくる人が現れます。たとえば自分が嗜んでいる書籍や音楽や映像などを 1-作品 1-トピックで一気に20個つくったりします。これを **捲し立て（まくしたて）** と呼びます。

捲し立てはノイジーなのでやめましょう。ETM の会場がネットワーク型ノートベースで、かつスルーも許容できる場所であるとはいえ、限度というものがあります。おそらく会場にはトピック一覧を表示するビュー、そして最近更新された順で表示するフィルタリングがあるはずで、捲し立てをされるとここが乱れます。捲し立てられたコンテンツがずらりと並ぶわけです。

どうしても捲し立てをしたい場合は「映画好きの俺がおすすめをひたすら紹介するページ」トピックをつくるなど、専用の場所をつくってからそこでやります。これなら 1 トピックしか使わないのでノイジーにはなりません。1つの作品やその中のワンシーン、ワンセンテンスなどをトピック化したいと思ったら、そのときにつくりましょう。作品には有益な金言やわかりやすい説明があることが多く、ETM でもよくトピックになります。ただ最初から捲し立ててたくさんつくるのは違うよね、限度を弁えたいよねという話です。

もちろん細かいラインはチームや状況によって違うので、必要なら話し合ってください。少なくとも「Aさんの捲し立てがうっとうしい」「でも直接言うのはなぁ……」なんてことがないようにしてください。ここを放置すると、捲し立てが増えて会場の治安が下がります。各人が自分の好きなことをブクマするだけの展示場となってしまいます。ETM の会場は展示場ではありません。このあたりの議論は Scrapbox 開発者 shokai による右記ページも参考になります: WOM(Write Only Member) - 橋本商会。

明示的な否定

Ans: 全員のコメントを要求されている場合、その気がないならその気がない旨を明示的に書きます。

ETM は各自で好き勝手に進めていくものですし、No ABCD の Ball のとおり返事待ちを前提とした個別宛のコミュニケーションさえも行いません。それでも次第にトピックが収束してきたり、全員に関係する概念が出てきたり、あるいは特に仕事だと会社の関係上全員に見てもらわないといけない話題が出てきます。メンバー全員（あるいは大半）のコメントを聞かねばならない場面が起きるのです。これを **ウェイト（Wait）** と呼びます。

ETM ではウェイトは可能な限りゼロに近づけるべきです（※1）。できないにしても、通常は全員自律的に動けているはずですので現実的な時間内にコメントが返ってくるはず（もし慢性的に「来ない」場合はメンバーとして向いていません）ですが、それでも返ってこないことがあります。

自分にとってポジティブなトピックだとすぐ返せるのですが、ネガティブだと返しづらくなりスルーしてしまうからです。

このようなときに使えるのが **明示的な否定（Explicit Negative）** で、これは「いいえ」とか「だるいんでやめときます」とか「興味ないんでどうでもいいです」とかいったリアクションを明示的に書くことです。日本の文化だと否定はしづらいですが、ウェイトが宙ぶらりんになっていると精神的な負荷になりますし、反応がない人を催促する行為は催促者に負担がかかりますので、さっさと書いて済ませてください。否定したことで問題が生じるなら、その点は別途議論してください。トピックを一つつくって、そこでやるといいでしょう。

ウェイトは **反応を返さない怠惰なメンバー** からもたらされます。怠惰が許容されてしまうと、常に誰かが催促の負担を負ってしまいますので許容してはなりません。この怠惰の主因が（そもそもメンバーとして向いてないことを除けば）否定のしづらさですので、明示的な否定により打開するのです。明示的に否定を出すのはやりづらいかもしれませんが、怠惰の許容による弊害にはETM を形骸化してしまう力さえあるので、頑張って慣れてほしいと思います。

- ※
  - 1 基本的にウェイトは起きないはずです。起きるようなトピックを扱っているとしたら、おそらく扱い方がおかしい（ETM なのに他のメンバーに指示しすぎ）か、そもそも ETM で扱う事柄ではないかのどちらかでしょう。特に会社の場合はありがちですが、全員に必要となる諸々については、ETM とは別の手段でカバーすることをおすすめします。たとえば 3 週間の探索期間（1 日 8 時間勤務とする）を設けているとしたら、1 日の探索を 7.5 時間にして、残り 0.5 時間は会社雑務をこなす時間に充ててねとする等です。ただし、このやり方も油断するとすぐ「毎日 16:30 から 30 分を雑務時間にしようか」「定例会議にします、この間は通話やメンションには答えてもらいます」などとなってしまい、ETM の自由さが損なわれます。一見すると探索時間からは外れていますが、探索直後に会議があることには変わりはなく、会議の参加をすっぽかさないよう準備・調整が必要となるからです。実質的に探索時間の一部が侵食されているわけですね。そもそも無闇に会議をすると（本来 ETM で議論するべきことを）会議の場で行ってしまい、かつその内容を会場にフィードバックしない現象が起きます。会場の裏で活動することがまかりとおってしまうのです。それでは ETM の意味がありません。前章でも書きましたが、**探索期間・探索時間という聖域は侵食させてはなりません。**

役割を工夫する

Ans: 各メンバーには得意苦手、好き嫌いがあるので、それらを「役割」の形で明示化します。

ETM ではしばしば適性が分かれます。ある得意領域とそれに基づく行動を示したものを **役割** と呼び、通常メンバーには複数の役割がつきます。

役割の例を以下に挙げます。

- ブレスター。思いつきやアイデアなど発散が得意な人
- サマライザー。発散した事柄の要約や蒸留が得意な人
    - 要約と蒸留を分けたい場合は「サマライザー」「ディスティラー」と分けてもいいかもしれません
- リンター。会場（ノートツール）の文法的・慣習的に不適切な書き方にツッコミを入れたり修正したりする人
- リンカー。あるトピックを見て、それと関係がありそうな別のトピックを提示する人
- 意思決定者。最終的な意思決定を行う人。通常は社長やマネージャーなど一人
- スティッカー。厳しい意見をズバッと書いてくれる人
- トリックスター。他のメンバー全員が到底真似できない独特なことを書く人
- リマインドマン。反応が遅かったり検討が進んでなかったりする人に催促する人

たとえばAさんはブレスター（発散が得意）とリンカー（結びつけが得意）、Bさんはトリックスター、Cさんはリンカー（結びつけが得意）とリンター（書き方の修正）といったように役割が分かれます。ここで **役割のとおりに行動しろと言っているのではなく、その役割が示すアクションが得意であるか、よく行うか、との意味にすぎない** ことに注意してください。強要するものではありません。

役割はこれ以外にも無数にありますし、上手く言語化できれば何でも定義できます。ともかく、色んな役割が通常自然に発生しています。役割に則った行動を重視し、それ以外は別の役割の人に担ってもらう意識で過ごすとETMは上手く回りやすくなります。なぜかというと、人によって得意苦手や好き嫌いが違うからです。それらを役割の形で捉えて、明示的に尊重できるようにするのです。

ETMでは役割は動的です。同じ人が常に同じ役割を続けるやり方もできますが、おそらく状況や気分に応じて変わるのが自然だと思います（ちなみに探索は遊びでもあるので気分も重要です）。これを **動的な役割（Dynamic Role）** と呼びます。あうんの呼吸で過ごしても構いませんが、自分達なりに役割を言語化しておくと、やり取りする上でもスムーズで便利です。言語化の過程で「こんな役割があると便利なんだけど」「Aさんは～～の役割に向いてる気がする」といった議論から始めるのも良いでしょう。しかし最も使いやすいのは、「じゃあ今からブレスターとして書き込んでみます」のような使い方でしょう。使う役割を宣言し、そのとおりに行動するのです。これを **ロールプレイ(Role Play、RPと略す)** と呼びます。いまいち進展がないときや、やる気が起きないときでも、特定の役割であればロールプレイできることがあります。もちろん、やりすぎるとウザくなりますので使う頻度やタイミングを調整しながら会場の様子を見てください。また、役割にすがって怠けてしまう心理もあるあるですので気をつけてください。特にリンターやリンカーなど、頭を使わず指摘すればいいだけの役割は「すがり先」として使われがちです。

さきほどから注意点ばかりですが、ストレングスファインダーやMBTIといった適性診断とも違うことに注意してください。適性のある役割を担うのが自然ではありますが、ETMでは「苦手だけど一時的にやる」「興味ないけどやってみたい」のような場面もありえます。発展的にはランダムに役割を割り当ててみることさえあります。役割とは、いうなれば「～～役」と書かれた帽子があっ

て、メンバー各々が勝手に被ったり、他メンバーに薦めたりするようなものです。かつ、被った帽子は見えているので「ああ、A さんは今はブレスターなんだな」といったこともわかります。帽子を被りさえすれば誰でもなれます。誰が何を被るかは自由であって、適性の有無で縛られるものではありません。

便利なトピック

Ans: つくると便利なトピックをいくつか紹介します。

ここで紹介したトピック以外にも便利なものは多数存在します。また、必要に応じて生み出すこともできるでしょう。便利な使い方を見つけたら、ぜひとも名前をつけて提案してみてください。トピックの使い方を制する者は ETM を制します。一方で、使い方を定めすぎると窮屈になってしまい、自由のメリットが損なわれてしまいますのでバランスも重要です。

日付トピック

Ans: 共同日記トピックがあると何かと便利です。

文芸的タスク管理の章にて日付ノートを紹介しましたが、これは複数人で ETM を行う場合も便利です。ひとりずつパーソナルスペースをつくって、つぶやきや日記を並べていくイメージになります。今日が 2024/09/02 なら 2024/09/02 ノートに、明日は 2024/09/03 ノートに皆が並べていく感じです。

日付トピックがあると誰が何をしているかわかりますし、雑談や相談も弾みやすいです。情報共有もやりやすいでしょう。日付ノートはホームでもあり、皆が高頻度に見に来る場所なので、ETM の最中はここを拠点にする感覚で過ごすと自然に過ごせます。

単位としては日単位と週単位がありますが、書き込む量はそれなりに多いでしょうから日単位がおすすめです。逆に明らかに少ないのであれば週単位でも良いかもしれません。ノート自体は誰かが毎回作る必要があります。面倒くさいなら自動化してください。誰かがつくることを期待して様子見していると、意外と誰もつくらず何も書き込まれないなんてことがよく起こります。こういう手抜きが続くとすぐに形骸化してしまうので、なるべく意識的に潰したいところです。

メンショントピックとポストトピック

Ans: 特定の人に見てほしいときは、それ用のトピックをつくります。

ETM は No ABCD なので、従来のように声を掛けたりメンションして通知飛ばしてといった形で見てもらうことはできませんし、するべきではありません。そうはいっても、見てほしい場面はや

はり存在します。そんなときに使えるのがメンショントピックとポストトピックです。

メンショントピック は、ノート名（トピック名）の形で直接聞くこと、特にそのようなトピックを指します。たとえばビジネスアイデアコンテストが 3 日後に迫っているとして、アイデア X が完成したが有識者 A さんに見てほしいときは `@Aさん アイデアX見てほしいです` のようなトピックをつくります。わかりづらいですが、これは「@Aさん アイデアX見てほしいです」という名前のノートをつくる、ということです。これを見ると、いやノートの内容を見なくともノート名だけで明らかに A さんに向けて書かれたトピックだとわかりますので、A さんもおそらく気付けるはずです。

前提として、ETM では多数のトピックが日々並ぶので、あるトピックが必ずチェックされる保障はありません。スルーされることも多いです。より盛り上がっているトピックであれば目にする機会が多くなるためほぼ気づけますが、大半はそうではありません。よって、どうしても見てほしいものがあれば、気づいてもらう率を上げる必要があるのです。メンショントピックはこの役に立ちます。

もう一つのやり方が **ポストトピック** です。郵便ポストなどのポストに由来しますが、これは A さんのインボックス（受信箱）的なノートをつくっておいて、A さんに言いたいことがある人はそこに書き込みをする、というものです。たとえば `inbox @A` というノート――つまりは「inbox @A」という名前のノートをつくっておき、A さんにお願いしたい人がお願いを書いたりします。この性質上、A さんは自分のポストトピックをチェックする義務に駆られます。あるいは「単に更新されたら上がってくるのでそれでたぶん」「普段はチェックしてないです」「もしかしたら見逃すかもだけどごめんねー」ならそれでも構いません。ETM では強制はしませんし、自分自身もまた義務感に駆られる必要はありません。

メンショントピックとポストトピック、どちらのやり方が合うかは状況によりますが、**両方使うとノイジーなので片方だけ使いましょう**。筆者としてはメンショントピックを推奨します。というのも、ポストトピックは持ち主の負担が大きいからです。何が書いてあるかは開けてみるまでわからず、まさに郵便ポストに入っているチラシのような負担感があります。これは自由な探索をウリとする ETM の心地を損ねます。メンショントピックの方が、トピック単位で相手にする感じがあって自然にやりやすいです。

いずれにせよ分量が多すぎるとノイジーなので、ここぞのときだけ使いましょう。**1日1回届く、でもまだ多いと感じます**。そもそもこのようなやり方に頼るのは ETM に反します。基本的に一度も使わないに越したことはないですし、通常は日付トピックに書いておくなり、A さんのパーソナルスペースに一言書き込んだりすれば気付いてもらえます。もしメンションやポストを何回も使わないといけないのだとしたら、おそらく ETM の前提を満たしていません。前章のセットアップを参照してください。

サルベージトピック

Ans: ある話題が会場内でどう扱われているか知りたいときは、検索や再読をしてみてまとめノートをつくります。

情報収集先として「ETM を行っているこの会場」も挙げることができます。しかし会場は各自が自由にトピックを広げたり詰めたりしている場所であり、記事や書籍のようには整っていません。そのままではインプットしづらいです。というより、**ヒントは散らばっているが正解が示されていない状態であることが多い** ので、単に読むだけではあまり役に立ちません。「これらを踏まえて私はこう理解した」といったように自分の解釈を交えることが重要になってきます。

これを行うのがサルベージトピックです。サルベージは救出・復旧の意であり、トピックの海から特定の話題を復旧するニュアンスを込めています。具体的には検索結果をコメントつきでまとめていく、ということをします。

仮にリモートワークについて深堀りしたいとして、「リモートワーク」をサルベージしてみるとしましょう。以下のようなノートをつくることになります。

```
リモートワークのサルベージ
 （リンク1）。（一言二言で所感を書く）
 （リンク2）。（一言二言で所感を書く）
 （リンク3）。（一言二言で所感を書く）
 ...

リモートワークのサルベージ
 （リンク1）
  （一言二言で所感を書く）
  ...
 （リンク2）
  （一言二言で所感を書く）
  ...
 （リンク3）
  （一言二言で所感を書く）
  ...
 ...
```

つまり検索してヒットしたノート名とそれを見た上での所感を並べていきます。具体的に書いてみた例を挙げます。

リモートワークのサルベージ

［Remotty］

ソニックガーデンのバーチャルオフィスツール。2分ごとに自席のカメラ画像を反映したものを一覧表示してて皆の様子がわかる

□さんが［ソーシャルプレゼンス］と一般化してる

［デスク爆弾］

>職場などで自分のデスクに前触れもなく近づいてきて延々と雑談や仕事の相談をされること

Z世代のワードとあるが、リモートワークの普及に伴って生まれた現象の一つってイメージ

［ラセングルのリモートワーク］

FGOをつくってる会社のリモワエ夫事例

会社としては「場所を問わない」がコンセプトなので、ツールと啓蒙を頑張ったそう

...

ここでは 3 トピックほど挙げましたが、選び方に指定はありません。3 つでもいいですし、10 つでもいいです。先に検索結果からめぼしいトピックを 5 つあげて、そこからさらに 3 つだけ読むことにする、などでも構いません。何を挙げてどう書くかは完全に自由です。

サルベージを行うと、（そもそも会場にそれなりの情報が書き込まれている必要はありますが）その話題に関する様々な情報が見つかります。本質が見えてきたり、自分が知らなかった視点と出会えたりして検討が先に進みやすいです。どちらかと言えば個人で ETM をやっている場合に重宝すると思います。

複数人で行っている場合でも、高頻度で登場するワードであればサルベージする価値があるかもしれません。特にプロジェクト、タスク管理、DX など「人によって捉え方が全然違う言葉」は結構ありますので、サルベージトピックをつくることでそのあたりの違いを可視化できます。

デメリットを挙げるなら二つで、純粋に疲れやすいため多用しづらいことと、実践する人の主観がゴッリゴリに混ざるため内容に偏りが生じやすいことです。

ステーショントピック

Ans: ある話題 T を深堀りしたいときに、T の前線基地（ステーション）をつくります。

ETM ではトピックが散らかりがちなので、作業デスク的な前線基地をつくって、そこで作業するようにするとやりやすくなります。このようなトピックを **ステーション** と呼び、トピック名には必ず「ステーション」をつけます。たとえば雑談的に自転車の話をガチで行いたい場合は「自転車ステーション」トピックをつくって、ここから各種議論や情報を辿れるようにします。名前に「ステーション」が入っているので「ああ、ここは自転車ネタの基地なんだな」とわかります。

ステーションは主にリンク集になります。ここで検討を深めるとすぐにとっ散らかって収拾がつかなくなるので、検討は別にトピックをつくってそちらで行い、ステーションからはなるべくリンクを張るようにします。全体状況やブロックライティングの文脈ブロックなど書いておくと便利な情報は適宜書

いてもいいですが、まずは **ステーション≒リンク集** のつもりで運用してみると感覚がわかってくると思います。

つくりかたのコツは、リンクを自分の言葉で束ねることです。以下に例を示します。

日記で盛り上がってる日
[2024/08/21]
[2024/08/15]
[2024/08/14]
[2024/08/11] ここが発端、なぜか自転車の話題が大盛りあがり

自転車ネタのトピック例
[ママチャリ vs クロスバイク]
[A-bike city 折りたたみ自転車]
[とあるロードバイクガチ勢が皆さんを沼に引きずり込むページ]
[Bianchi ROMA 4]
[自転車通勤]

自転車ネタから出てきた議論やアイデア
[自転車通勤を認めてほしいです]
[広い敷地を持つ事業者向けにスポーツバイク導入を提案する]
[自転車エバンジェリストとか自転車エクササイズとか]
[自動車はすでにソフトウェアの塊だが、自転車も間もなくそうなる説]

[自転車どうしてます？]
皆さんの現状や所感を書いてほしいです！

これは 4 つの観点で整理していますが、ただリンクを無造作に並べるよりもわかりやすいと思います。たとえばビジネスアイデアを見たい人は 3 番目のブロックを見ればいいとすぐにわかります。

余談ですが、4 つ目のブロックだけは少し書き方が違っていて、束ねるものがない代わりに□さん（□の表札を使っているメンバー）がお願いを書いてますね。本来なら 2 つ目のブロックにぶら下げてもいい気がしますが、□さんが書き込んでほしくてあえてそうしたのかもしれません。

良いステーションの目安は「このトピックから辿って読んでいくのが一番近道」だと言えることです。通常、ETM では、話題 T について知りたいときは自分なりに探索的に読み進めていきますが、ステーションがある場合は T のステーション（≒リンク集）から辿っていく形になります。

文芸的タスク管理の言葉で言えば、ステーションとはリンクばかりが並んだトランク（幹）です。

まとめ

- ETM に役立つヒントを紹介しました
  - 個人で取り組む場合に役立つもの、複数人で取り組む場合に役立つもの、両方当てはまるものが混在しています
- ====
- 文芸的タスク管理
  - そもそも ETM は文芸的タスク管理をベースにしているので、困ったらまずはおさらいする
- スクイーズアウト
  - 絞り切る（自分なりに 100% を出し切る）ことが納得感に繋がり、意思決定も加速する
- ブロックライティング
  - 1-space インデントの箇条書きの塊（ブロック）をつくっていく手法
  - 発散や検討を行いながら概念、方向性、タスク、そして要約をつくるのを 1 サイクルとする、これを繰り返す
  - ブロックには種類があり、種類を示すためにアルファベット一文字（レター）をつける
- 個人と協調の線引き
  - トピックは個人でつくって、それを見てもらうのが良い
  - パーソナルスペースとパブリックスペースを使い分ける
- 個と和のバランス
  - 興味ドリブン、スルーの許容、オタク注意報という感じの世界観を心がけると上手くいくと思う
  - ETM は No ABCD だがメンバーからの反応を待つときがある（ウェイト）、ウェイトはなるべくなくしたい
    - 主因として「否定的な反応をしづらい」があるので、明示的に出すと良い
  - 役割も意識して、言語化して捉えられると意思疎通がスムーズになる
    - 役割は動的になりがち
    - 帽子を拝借して一時的にロールプレイするつもりで望むと、使いやすいかもしれない
- 便利なトピック
  - 日付トピックは日々の動線になる
  - 特定のメンバーに言いたければメンショントピックやポストトピック
  - トピック化されていないが、よく使われている言葉があればサルベージすると何か見えてくるかも
  - ステーショントピック（リンク集）をつくるのも便利

=== Chapter-27 参考文献 ===

参考文献

1

2016, D・カーネギー

Amazon.co.jp: 道は開ける 文庫版 電子書籍: D・カーネギー, 香山 晶: Kindleストア ## 2 2013, スティーブン・R・コヴィー (著), フランクリン・コヴィー・ジャパン (翻訳)

完訳 7つの習慣 人格主義の回復: Powerful Lessons in Personal Change | スティーブン・R・コヴィー, フランクリン・コヴィー・ジャパン | ビジネススキル | Kindleストア | Amazon

3

2024, マーク・フォースター (著), 青木 高夫 (編集)

仕事に追われない仕事術マニャーナの法則・完全版 | マーク・フォースター, 青木 高夫 |本 | 通販 | Amazon

4

2013, 釘原直樹

人はなぜ集団になると怠けるのか 「社会的手抜き」の心理学 (中公新書) | 釘原直樹 | 哲学・思想 | Kindleストア | Amazon

5

2018, フレデリック・ラルー (著), 嘉村賢州 (著), 鈴木立哉 (翻訳)

Amazon.co.jp: ティール組織 – マネジメントの常識を覆す次世代型組織の出現 eBook : フレデリック・ラルー, 嘉村賢州, 鈴木立哉: Kindleストア

6

DX白書2021 | 書籍・刊行物 | IPA 独立行政法人 情報処理推進機構

7

2021, マッキンゼー緊急提言 デジタル革命の本質:日本のリーダーへのメッセージ

https://www.mckinsey.com/jp/~/media/McKinsey/Locations/Asia/Japan/Our%20Work/Digital/Accelerating_digital_transformation_under_covid19-an_urgent_message_to_leaders_in_Japan-jp.pdf

8

2015, センディル ムッライナタン (著), エルダー シャフィール (著), 大田 直子 (翻訳)

Amazon.co.jp: いつも「時間がない」あなたに　欠乏の行動経済学 (早川書房) eBook : センディル ムッライナタン, エルダー シャフィール, 大田 直子: 本

9

2017, 帚木 蓬生

Amazon.co.jp: ネガティブ・ケイパビリティ　答えの出ない事態に耐える力 (朝日選書) eBook : 帚木　蓬生: 本

10

2021, 及川 卓也 (著), 小城 久美子 (著), 曽根原 春樹 (著)

Amazon.co.jp: プロダクトマネジメントのすべて 事業戦略・IT開発・UXデザイン・マーケティングからチーム・組織運営まで eBook : 及川 卓也, 小城 久美子, 曽根原 春樹: Kindleストア

11

2019, 丸山 宏

新 企業の研究者をめざす皆さんへ | 丸山 宏 | 実践経営・リーダーシップ | Kindleストア | Amazon

12

2018, James Clear

Amazon | Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones (English Edition) [Kindle edition] by Clear, James | Business Management | Kindleストア

13

2003, エリヤフ ゴールドラット (著), 三本木 亮 (翻訳), 津田 公二 (寄稿)

クリティカルチェーン | エリヤフ ゴールドラット, 三本木 亮, 津田 公二 | 英米の小説・文芸 | Kindleストア | Amazon

14

2021, 國分功一郎

Amazon.co.jp: 暇と退屈の倫理学（新潮文庫） 電子書籍: 國分功一郎: Kindleストア

15

2018, ジョン・T・カシオポ (著), ウィリアム・パトリック (著), 柴田裕之 (翻訳)

Amazon.co.jp: 孤独の科学　人はなぜ寂しくなるのか (河出文庫) eBook : ジョン・T・カシオポ, ウィリアム・パトリック, 柴田裕之: 本

16

2015, エリン・メイヤー (著), 田岡恵 (著), 樋口武志 (翻訳)

Amazon.co.jp: 異文化理解力 – 相手と自分の真意がわかるビジネスパーソン必須の教養 eBook : エリン・メイヤー, 田岡恵, 樋口武志: Kindleストア

17

2022, 吉良野すた

Scrapboxing(スクラップボクシング) | 吉良野すた | 工学 | Kindleストア | Amazon

18

2023, ジム・クリフトン (著), ギャラップ (著), 古屋博子 (翻訳)

さあ、才能(じぶん)に目覚めよう　最新版 ストレングス・ファインダー2.0 | ジム・クリフトン, ギャラップ, 古屋博子 |本 | 通販 | Amazon

19

2024, デヴォン・プライス (著), 佐々木寛子 (翻訳)

Amazon.co.jp:「怠惰」なんて存在しない 終わりなき生産性競争から抜け出すための幸福論 eBook : デヴォン・プライス, 佐々木寛子: Kindleストア

20

1999, Amy Edmondson

Psychological Safety and Learning Behavior in Work Teams

21

2019, 吉良野すた

ルーチンタスクの底力: やり忘れとストレスをなくす仕組みと実践 | 吉良野すた | 家事・生活の知識 | Kindleストア | Amazon

22

2024, 片野 秀樹

Amazon.co.jp: 休養学－あなたを疲れから救う eBook : 片野 秀樹: Kindleストア

23

2021, 溝口敦 (著), 鈴木智彦 (著)

Amazon.co.jp: 職業としてのヤクザ（小学館新書） eBook : 溝口敦, 鈴木智彦: Kindleストア

24

2019, ライダー・キャロル (著), 栗木さつき (翻訳)

バレットジャーナル 人生を変えるノート術 | ライダー・キャロル, 栗木さつき |本 | 通販 | Amazon

25

2019, 吉良野すた

執筆を効率化したい人のための秀丸エディタ実践入門 | 吉良野すた | 工学 | Kindleストア | Amazon

26

1969, 梅棹 忠夫

Amazon.co.jp: 知的生産の技術 (岩波新書) eBook : 梅棹 忠夫: 本

ツール

Habitica

Habitica - Gamify Your Life

> やる気を出して目標を達成しよう。
>
> 楽しみながら仕事を片付けよう！400万人を超えるHabiticaの冒険者達と一緒にあなたの人生もタスクも一気に改善しましょう。

Chill Pulse

Steam : Chill Pulse

> Chill Pulseは、生産性ツールと穏やかな環境を組み合わせています。ポモドーロタイマーやToDoリストを使用して集中力を高め、サイバーパンクの都市から古代の風景まで、環境音がリラックス感を高めます。

todo.txt

Todo.txt: Future-proof task tracking in a file you control

筆者も以前解説記事を書きました: タスク管理メソッド todo.txt が面白そう #タスク管理 - Qiita

> タスク管理メソッドの一つ。米 Lifehacker の Gina Trapani が考案した。いわゆる Todo リストの一種だが、todo.txt というテキストファイル一つのみで運用するためシンプルで、軽くて、管理もしやすいのが特徴。テキストエディタを使って手で編集しても良いが、ツールも多数存在する。コマンドラインツールからスマホアプリまで様々。

GitHub Issues

GitHub とはソースコードをソーシャルに管理するプラットフォームで、リポジトリという単位でソースコードや関連ドキュメントをアップロードします。ソーシャルと書きましたが、複数人で分業したりコメントやレビューをしたりといったことができる能力を備えています（その分勉強と慣れは必要で本職のエンジニアも苦戦しますが）。

この GitHub には Issues という 1-課題 1-チケット で課題を発行する仕組みがあります。ソーシャルなので誰でも（たとえばそのソフトウェアの利用者でも）チケットを切ることができます。一般的には BTS ――バグ追跡システムと呼ばれるものですが、汎用的にも使えます。要は 1-アイテム 1-チケットであり、1-チケットには1-ページが割り当てられるので、1-話題 1-ページと捉えれば汎用的に使えます。まして、課題管理だけあってタスク管理に相当する機能もありますから、タスク管理としても使えます。

この GitHub Issues は、スッキリしていて使いやすいので、個人タスク管理として使っても抵抗感がありません。特に近年は Project というカンバン機能もリリースされており、俯瞰もしやすくなったので、より使い勝手も上がっていると思います。ただ、それでも元はソースコードを扱うエンジニア向けですので、非エンジニアの方にはハードルは高いでしょう。

リンク:

- 公式
  - GitHub Issues のドキュメント - GitHub Docs
- 参考記事
  - GitHubを最強のToDo管理ツールにする #GitHub - Qiita

Asana

チームの仕事、プロジェクト、タスクをオンライン管理 • Asana

ClickUp

ClickUp™ | One app to replace them all

monday.com

【公式】monday.com | 業務管理の、一歩先へ。

Work Management という製品がプロジェクトタスク管理機能に相当します。

Trello

どこにいてもチームのプロジェクトを管理 | Trello

Todoist

Todoist | ToDo リストで仕事と生活を整える

TimeTree

TimeTree（タイムツリー） | 予定の共有と相談がおどろくほど簡単にできるコミュニケーションアプリ

Cross

夫婦のためのToDo共有アプリ Cross「クロス」

Toggl

Toggl Track: Time Tracking Software for Any Workflow

Redmine

Redmine.JP ― オープンソースのプロジェクト管理ソフトウェア Redmine 日本語情報サイト

Backlog

Backlog | チームで使うプロジェクト管理・タスク管理ツール

hown

howm: Hitori Otegaru Wiki Modoki

Scrapbox

Cosense - Share Experience, Make Co-sense, and Go！

※2024/05/21、名前が HelpFeel Cosense に変わりましたが、本書では Scrapbox の呼称を使います。

Obsidian

Obsidian - Sharpen your thinking

Tana

Tana

※2024/08/03 現在は Waiting List への参加を受け付けています。

Miro

Miro | 革新をもたらすビジュアル ワークスペース

ルータム

ルータム | 毎日のタスク時間を記録、より良くするアプリ

Routinery

Routinery | 루티너리

メソッド

アイビー・リー・メソッド

100年の歴史を持つ“生産性向上”メソッド「アイビー・リー・メソッド」とは｜Business Insider Japan

> このメソッドでは、毎晩寝る前に、次の日にやらなければならない重要なタスクを重要度順に6つ書き出す。そして翌日、タスクに順番に1つずつ取り組んでいく。

たった 6 個にまで絞るという大胆な取捨選択をします。ChatGPT 曰く、TODO リストの起源だそうです。いわば 6 項目限定の TODO リストと言えるでしょう。そもそも TODO リストは「1行1項目」「上から順番に処理する」「終わった項目は完了にする」などの概念があり、自然ではありません（ビジネスなどで縁がないと大人でも知らない人はいます）。100年以上前であり、TODO リストの起源でもあるといわれても納得感はあるかなと個人的には思いました。

時間管理マトリクス

書籍『7つの習慣』で言及されたマトリックスで、タスクを 4 つに分類することで優先順位付けをサポートするもの。

	緊急	緊急じゃない
重要	1	2
重要じゃない	3	4

1 の「重要だけど緊急じゃないもの」に投資すること（勉強やトレーニングや健康のメンテナンス）や、4 の「重要でも緊急でもないもの」を捨てること、3 の「重要じゃないのに緊急性の高いもの」を減らしていく努力をすることなどが強調されています。ちなみに娯楽は 4 に入るので、かなり意識が高いです。

何が重要かの判断は主観で構いませんが、書籍では人格主義に則っています。たとえば「私はオタクなのでオタ活が重要である」といった生き方は想定されていません（4 の娯楽に入るのでむしろ捨てる対象です）。とはいえ、重要と緊急の二軸での分類は汎用性が高く、現代でも使いやすいでしょう。

タスクシュート

大橋悦夫によって考案されたタスク管理ツールで、タスクを順に並べて上から順に消化していくことと、その際に開始時刻と終了時刻を厳密に記録するのが特徴です。タスクには見積時間

も書けますので、厳密に運用した場合は「見積」と「実績（かかった時間）」が記録されることになります。記録を見れば見積との差がわかりますし、自分のパフォーマンスも見えますから、次第に高精度な見積もりが可能になっていきます。

元は Excel ツールでしたが、その後スマホアプリやクラウド版ツールなど強化が図られ、今ではメソッドとしての側面も備えています。特に自分の行動のすべてを記録していくことで、自分の生活リズムを定量化できます。定量化した実データは「あなたの実態」「あなたのリズム」なわけですから、あとはこれを踏まえた上で無理のない生活を組み立てていけます。ツールでは定期的なタスクを入れたり、未来日に入れておいたりもできますから、これ一本で普段の行動すべてを管理できます。ヘビーユースになると「起床後のトイレ」「歯磨き」「爪切り」「ゴミ捨て」といったレベルでタスクを詰め込んで最適化することさえできます。

リンク:

- タスク管理ツール・TaskChute2
  - 大橋悦夫によるオリジナル、Excel 製
- TaskChute Cloud
  - SaaS 型のタスクシュートツール
  - 2024年現在、ツールと言えばこちらを指すと思います
- 一般社団法人タスクシュート協会
  - 2022年11月設立
  - 「自分らしい時間的豊かさを追求する」を掲げています

タイムボックス

事前に時間枠を定めておき、その枠内でのみ仕事をする、という手法です。

シンプルなのは「仕事は8:30〜11:30、13:00〜17:00の間だけ行う」のようなものです。これは午前に3h、午後に4hのタイムボックスをつくっていますが、ボックス内で行うタスクの指定はありません。

アジャイル開発という製品開発手法ではスプリントという単位が使われます。これは1〜2週間というタイムボックスを区切り、何のタスクを行うかも決めた上で、それだけをやります。ボックスを終えたら、打ち合わせをはさんで振り返りつつ次のボックスで何をするかを決めます。そうやって一つずつ確実にこなしながら前進していきます。

他にも色々あるでしょうが、本質は時間枠を決めることと、その枠を超えてだらだらと粘らないということです。メリハリをつけるとも言えます。筆者のような几帳面な人間は言われるまでもなく採用していますが、そうじゃない人には目からウロコかもしれません。

リンク:

- Timeboxing - Wikipedia

- 生産性を上げたければ定時上がりが効率が良い｜牛尾　剛

カンバン

ソフトウェア開発手法の一つで、1-タスク 1-カードでタスクをカード化し、これを進行状況（未着手、着手中、完了）ごとに区切られたボードに配置することで可視化を行う手法です。ポイントはキャパシティで、チームが抱えきれる程度の分量のタスクだけが存在していることを保証するところにあります。

たとえば、タスクの負荷をポイントで表現する（1ポイントよりも5ポイントのタスクが重たい）として、あるチームは一週間で 30 ポイント処理できるとしましょう。このとき、週のスパンで見た場合、カンバン上に 30 ポイントを超える数のタスクがあってはいけませんよね。今週残り 2 日で、すでに 20 ポイント完了している状態で、「進行中」に 15 ポイント分のタスクがあるとしたら、これはおかしいわけです。少なくとも 5 ポイント分は過剰な負荷になりますので取り下げるべきです。そもそも、こんな超過にならないようにカンバンの状態を維持しましょう、というわけです。

このカンバン、元は1950年代にトヨタで開発されたもので、工場において JIT（ジャストインタイム）――必要な部品を必要な時に必要な数量だけ用意するためのプロセスとして使われていたそうです。生産ラインを流れる物や箱に、文字通り看板が貼り付けられていて、そこに数量などの情報が書かれていたのだとか。

リンク:

- カンバン - 概要 | Atlassian
- かんばんとは - Azure DevOps | Microsoft Learn
- Kanban - Toyota Production System guide - Toyota UK Magazine

ミッションステートメント

書籍『7つの習慣』で登場する概念で、個人や組織の信念を文章化したものです。憲法とたとえられることもあります。

GPT-4に問い合わせた結果を引用します。

> 『7つの習慣』は、スティーブン・R・コヴィーが著した自己啓発およびビジネスの本であり、個人や組織が効果的に成果を上げるための原則を示しています。この中で取り上げられる「ミッション・ステートメント」とは、個人または組織の目的、価値観、および長期的な目標を定義した宣言です。これは、その人または組織が何のために存在し、どのような原則に基づいて決断を下し、どのような大義や貢献を目指しているかを明確にします。
>
> 個人のミッション・ステートメントは、自分自身の理想、目標、価値観を反映したもの

> で、人生の意思決定の基盤となるものです。それに沿って行動することで、より一貫性のある人生を送ることができ、自分自身の最も重要な優先事項に焦点を当てることが可能になります。
>
> 組織の場合、ミッション・ステートメントはその組織が追求する目的、提供する価値、及び達成しようとする具体的な目標を示すものであり、従業員やステークホルダーに対してその組織の意義を伝える重要なツールです。これにより、組織内の全員が共通の目標に向かって努力することが促されます。
>
> ミッション・ステートメントの作成には、自己反省と深い考察が必要であり、しばしばこれを作るプロセス自体が個人や組織の成長に大きく貢献します。

末尾で述べられているとおり、かんたんにつくれるものではなく、深い内省を必要とします。ひとりブレインストーミングやフリーライティングなど「頭の中の諸々を外に出す」営みも必要でしょう。他者との対話により気付きを得るのも有効です。

筆者としては「継続的な自問自答」と「自分のコンプレックスから逃げずに向き合うこと」が重要かなと感じます。極めてプライベートな内容になる可能性もあるため、人に見せることは考えない方が良いと思います。むしろ見せる意識を持ってしまうと、取り繕ってしまいます。取り繕ったステートメントは、心の何処かでひっかかってしまうため役に立ちません。一方で、書籍では人格主義を謳っており、「人に見せられないような恥ずかしいステートメントなどありえない」としているところがあると思います。それで済むのならそれでいいと思います（筆者はそうではありません）。

組織の文脈で言えば、企業理念と同義だと思いますが、大人数で議論しても収束しないので、少なくとも最初のたたき台と最後の収束は少人数で行うべきでしょう。

いずれにせよ、憲法レベルの主柱になりますから、そうかんたんに変えるものではありません。しかし全く変えないものでもなく、継続的に内省や議論を重ねて変更事項をストックしておく＆キリのいいタイミングで微修正や刷新を行うイメージになると思います。利用規約や就業規則のようなものと考えてもいいかもしれませんね。

ポモドーロ・テクニック

1980 年代後半、フランチェスコ・シリロによって開発された集中術で、25 分の集中 → 5 分の休憩のサイクルを繰り返すことで集中のリズムをつくります。

非常に有名なメソッドということもあり、様々な解釈や亜種が存在しますが、コアコンセプトは以下だと思います。

- 1ポモドーロという単位（25分取り組む → 5分休憩）
- 最初にタスクを決めて、それだけを行うポモドーロを回す
- ポモドーロはキッチンタイマーなどを使って厳格に管理し、厳格に守る

- 4ポモ回したらで長めの休憩を入れる（30分とか）

つまり、ポイントは以下となります。

- 進捗や優先順位は考えない。やると決めたタスクをポモドーロでひたすら回すのみ
  - 逆を言えば「これをやればいい」「これをやりたい」など、やるべきタスクが見えていない場合は回せません（あるいはやるべきかわからないけど回してみるかというギャンブルになりますが、経験上ギャンブル的だと集中しづらいと思います）
- タイマーが鳴ったら次の行動に移る、を強く心がける
  - 筆者は「タイマーの奴隷になる」と表現しています
- ポモドーロには 5 分休憩があるが、それでも疲れが溜まってくるので、長めの休憩を差し込む
  - そうしないと日中フルでもちません

なぜこんなことをするかというと、集中したいからです。集中して目の前のタスクに取り組み続ければ、進捗感が出て気分もいいし、取り組んだことで色々見えてきます。このタスクをすればいいとわかってる、だけどなぜか集中できない、やる気出ない、たすけて――それを叶えるのがポモドーロ・テクニックだと筆者は理解しています。

トレメント――トレーニング要素に適応できる強い人であれば「いやそんなことしなくても普通に集中して取り組めばいいでしょ」と疑問を抱くのですが、それができない人も多いです。あるいは、普段はできる人であっても、タスクによってはできなかったりするでしょう。たとえば興味の無いタスクやつまらないと感じるタスクはそうなりやすいと思います。こういうとき、ポモドーロというシンプルな制約はバランスが良いのです。

パラメータとして「25分」「5分」「4回」などがありますが、このあたりの好みは分かれると思います。まずはそのままセオリーどおりに使うのが良いでしょう。それで慣れてきたら、自分のリズムとのギャップも見えてくるはずですから、微調整していけばいいと思います。筆者は調整は要らない派です。というのも、ポモドーロ・テクニックはタイマーの奴隷となって「着手を続ける」ことが目的であり、細かい数字の合う合わないはどうでもいいからです。たとえ 25 分だと中途半端な事が多いなーと感じても、いたずらに伸ばしたりはせず、淡々とタイマーに従うのみ。むしろ細かい快適性を意識しだすと「自分が快適に取り組めるかどうか」の模索という本末転倒に陥ります（これはこれで面白いから厄介です）。

やめどきは人それぞれで良いです。3 ポモくらいで不安がなくなったのなら、もうその日は切り上げていいでしょうし、「あとはタスク管理をして作業的にこなせればいけるな」とわかったなら以降はポモドーロを回さなくてもいいでしょう。逆にポモドーロを回さないと仕事できないというのであれば、毎回回せば良いでしょう。

リンク:

- Pomodoro® Technique - Time Management Method

インボックスゼロ

未処理の場所（インボックス）をつくっておき、以下の二つのルールを適用します。

- タスクを溜めていくこと
- タスクはすぐに処理しなくてもいいが、すべて処理して空にすることを目指すこと

つまり「0 件にする（空にする, ゼロにする）ことを目指した未処理領域」です。

この概念は個人タスク管理において非常に重要であり、ゼロという基準があることでメリハリが生まれます。私たち人間は機械ではないので、オンとオフを区別することは大事です。問題はその切替をいつ行うかですが、インボックスゼロはまさに「ゼロになったらおしまい」とできてシンプルなのです。

例としてメールの受信箱やチャットや SNS の通知欄はよく知られています。ここがゼロになるといったん安心です。また Dailer のデイリータスクリストなど日単位で仕切る（一日の最初に今日やるタスクを溜め、これが全部終わったらその日はもうおしまいにする）考え方もあります。

その他、応用範囲は広いと思います。たとえば筆者は玄関に書類整理用のインボックスをつくっており、ポストに届いた書類を全部ここに置いていて、適当なタイミングで目を通して処理しています。ここがゼロだと気持ちが浮きますし、ゼロにしたいのでさっさと処理しようという気にもなります。同様に、玄関には「日用品付箋ゾーン」があり、買い出したいもの（を書いた付箋）が貼り付けてあります。これもインボックスであり、買い物に行く前に目を通してメモしてから出かけます。買ってきたら付箋をストックに戻して終了。このゾーンについても、ゼロ（何も付箋が貼り付いてない）の状態＝何も買わなくていい、ですから精神的に楽できます。

習慣トラッカー

ハビットトラッカーとも呼びますが、習慣を管理するためのシンプルな手法です。

やり方としては、習慣名と日単位の時間軸を並べて、毎日各習慣を行ったどうかをチェックをつけて記録していきます。たとえば縦軸に習慣を、横軸に日を並べます。画像検索をすればイメージは一発でわかると思います。デジタルツールである必要はなく、アナログな手帳でも運用できます（し、むしろアナログの方がやりやすいくらいです）。

手法自体はきわめてシンプルですが、難しいのは「自分にとって定着しやすい習慣」を上手く選んでいくことです。自分がやりたいと思えるものを一つずつ追加していく、うっかり忘れてしまわないために生活の動線上に自然に配置する、モチベーションを上げるために人に報告したりアナログな手帳で気分が上がるデザインにしたりするなど、工夫の余地は色々とあります。

初心者の方はよく何でもかんでも追加しがちですが、それで律儀に従えたらそもそも苦労はしないのです。人間は怠惰な生き物であり、モチベーションは人それぞれですし、同じ人であって

も状況や調子次第なので、それらを上手く尊重した形で注入していかねばなりません。人や動物との接し方はその人その動物次第ですが、習慣も同じで、自分という生物に則らねばならないのです。この部分だけでも一冊の本になるでしょうし、実際その手の本は多数ありますが、本質は同じです。ただ、その本質に則るのが難しくて、通常は色んな発想や事例を見ながら模索していくことになります。もちろん、難なくこなせてしまう人もいます。本書でも度々上げていますが、結局はトレメントに強い人が有利です。

習慣トラッカーの起源は不明ですが、ChatGPT 曰く、書籍Atomic Habitsにて紹介されたことで広まったそうです。Amazon のレビューは 2024/05/30 時点で 16 万件あります。

習慣トラッカーには応用例もあります。たとえばセルフモニタリングです。セルフモニタリングの厳密な定義は割愛しますが、巷では体調、特に不調の原因を探るための行動記録を意味します。これは習慣トラッカーで実現できます。たとえば朝に薬を飲む、朝に日光浴する、夕方に薬を飲む、月と木で通院する、寝る前に薬を飲むといった行動を習慣とみなしてトラッカーに載せるのです。毎日記録していけば、たとえば不調が起きた日に「薬を飲んでないじゃないか」とか「通院をサボり始めてるな」といったことが記録としてわかるわけです。

GTD

Getting Things Done の略で、デビッド・アレンが開発した人生管理術です。

コアは4つあって、まず頭の中でもやもやしているすべての気になることを外に出すことです。次に、外に出したそれをワークフローに従って分類して、適切なタイミングで処理できるようにします。たとえばネクストアクション（次何するかという選択肢）、プロジェクト（すぐには終わらないがn回の着手で終わる程度のもの）、いつかやる（永遠にやらないかもしれないがいつかやりたいことを溜める）、インボックス（未処理を溜める場所でなるべく早くワークフローに流す）、連絡待ち（相手から連絡が来ないと進まない事柄を集める）などがあります。それからレビューという形で日々点検します。筆者は DWMY と呼んでいますが、Daily（毎日）、Weekly（毎週）、Monthly（毎月）、あとオプションですが可能なら Yearly（毎年）の点検をするのです。点検は振り返りと言っても構いません。そして最後に、人生全体で整合性を取るために信念や哲学、長期的に目指したいこと、数年スパンで達成したいこと、現在抱えている制約などをリスト化しておき、レビュー時に適宜参照・修正します。これは高度モデルと呼ばれます。

改めてまとめると、

- 1: 頭の中にあるものを全部外に出す
- 2: 1はワークフローに従って分類し、各々適切なタイミングで処理する
- 3: 日週月年ごとにレビューを行う
- 4: 人生全体の整合性を取るためのリストをつくる

中々に重厚なメソッドであり、本来は有償のセミナーを受けねばなりません。GTD は GTD® であり商標登録されています。筆者は公式の書籍を精読したり、ネットで情報を集めたり、人

の声を聞いたりして自分なりに解釈しました（GTDを噛み砕くとしてドキュメント化）。

もう一つ重要なのは、GTD が枠組みを捉えたものであり、具体的にどんなツールを使えばいいかは扱っていないということです。たとえば日々行動する際に使うネクストアクションをどのように実践するかは（少なくとも書籍上では）言及がありません。この部分は本書が役に立つでしょう。たとえば戦略の章で言えば、Dailer にてデイリータスクリストをつくったり、Robot としてタスクを全部洗い出して載せることができます。Calendarer としてひたすら予定にぶっこんで予定を消化するマシーンになるのもアリです。

鬼門はレビューだといわれます。たとえば最初に頭の中を出し切るのに数時間、日次レビューは毎日 10 分、週次レビューは毎週 1 時間、月次レビューも毎月 1〜2 時間くらいかかるわけです。これを全部律儀に、できるだけ欠かさず行う必要があります。これができないと全体的なメンテナンスができないので形骸化します。そして、この営みはまさに本書でも強調しているトレメントに他なりません。

PARAメソッド

情報管理メソッドの一つで、GTD より後発でありながらも、GTD 並の汎用性を備えています。ちなみに有償のメソッドです。

PARA の名の通り、情報を Project、AoR（Area of responsibility）、Resource、Archive の 4 種類に分けます。Project はゴールと期限を持つ一連の作業、Resource は今現在関心がある情報全般、Archive はいずれでもないものを入れる倉庫となっており、要は「プロジェクト」「情報」「それ以外」のざっくり 3 つに分ければいいだけ、と GTD と比べてずいぶんかんたんになっています。AoR は名前は怖そうですが、「健康で過ごす」のような「タスクというよりタスクを発生させる元となるもの」を指します。「維持するべき基準」や「現状抱えている制約」と言い換えてもいいでしょう（GTD では高度モデルの 4000m や 2000m に相当します）。AoR はタスクとして管理しづらいので、別途リストとして持っておいて、日々これを眺めながらプロジェクトに落としてこみ、行動としてはそのプロジェクトの方をこなすのです。

使うツールについては、ノートツールなら何でも構いません。仕事で OneNote、プライベートで Evernote など複数に分散しても構わないです。重要なのはどのツールでも一貫した構造を保つこととされています。単一メソッド、複数ツールというわけですね。階層は人間工学的に 4 段階までが良いとされていますし、ノートツールもそうなっています。たとえば OneNote は notebook - section - section group - page ですし、EverNote は application - stack - notebook - note です。トップ階層は P A R A の 4 つで固定して、その中は自由ですが最大 4 段階に収めるのが無難ですよ、というわけです。

筆者としては AoR の概念が画期的だと思います。また、PARA メソッドではコミットを大事にしており、ゴール無きプロジェクトは趣味でしかない、またはプロジェクト無きゴールは夢でしかないとも言っています。細かい運用は扱われていません（or 有償の部分で扱うのかもしれません）が、メソッド自体はたしかに汎用的で、ボリュームとしてもコンパクトであり、後発なだけあるなと

感じます。

リンク:

- The PARA Method: The Simple System for Organizing Your Digital Life in Seconds
  - 2021年時点では解説が充実していましたが、2024/06 現在（最新の更新は2023/04）では見れなくなっています

=== Chapter-28 あとがき ===

タスク管理の体系化は前々からやりたかったことでした。プロジェクト管理やチームワークで使われるようなものではなく、個人が個人のタスクを上手く管理するための諸々を上手く整理したかったのです。タスク管理、特に個人タスク管理は本質的に個人的なものだと思っていますので、真の意味での体系化は難しいでしょう。しかし、現状はあまりにも整理されてなさすぎるし、何より使われてなさすぎる――。所詮は趣味の範疇ですし、私もそのつもりでした。働き方や過ごし方なんて人の勝手ですから、どうとでもすればいいし、正直どうでもいいのですが、最近は特にそうも言ってられなくなってきました。

パンデミックによりリモートワークが加速しましたが、落ち着きを見せたところで世の中は再び出社に回帰しつつあります。週に n 回するとのバランス（ハイブリッドワーク）もよく見られますし、リモートワークは「最悪の過ち」か、グーグル元CEOらが是非に言及｜日経クロステック（xTECH）の記事も見て衝撃的でしたが、私は前々から心の中で何度も何度も「違う」「そうじゃない」と思っていました。オードリー・タンはブロードバンドは人権と言いましたが、私は「リモートは人権」だと唱えたいです。なぜなら、リモートワークのおかげで、通勤を初めとする「拘束」から解放され、各々の生活が尊重されるようになったからです。もちろん、あらゆるケースで当てはまるわけではありませんが、それができることが示されたからです。生活の尊重は水準となったのです。一度上がった水準はもう戻せませんし、戻すべきではありません。なのに、回帰に伴って、この水準も壊されつつある……。

私は仕事術を研究していることもあり、この流れを止めたくて微力ながら抵抗していました。常にリモートでいるべき、などと過激な主義を押し通したいのではありません。出社派も、リモート派も、どちらも共存できるようにしたいのです。リモートという選択肢を、多様性の一つとして確立したいのです。現状は前者が幅を利かせています。ハイブリッドワークもまさにそうで、週に n 回とはいえ強要されているわけです。いやいや、フルリモートしたい人がフルリモートするくらいもうできるでしょ、テクノロジーは十分なのだからできるはずだ――そう信じていましたが、道のりは非常に険しいのだとわかりました。

改めてわかったのは、テクノロジーだけではダメだということです。テクノロジー（技術）の他にメソドロジー（方法）も必要そうだな、との感触が日に日に濃くなってきました。特にリモートでは、各々が自律的に動いて情報を残すことが必須のリテラシーとなりますが、そもそもこれを持って

いる方が圧倒的にマイノリティなのです。テクノロジーがもっと発展すればリテラシー無しでも実現できるでしょうが、まだまだ遠そうです。まがいなりにも私がこんなことを言えているのは、単に私が研究してきたからにすぎません。実情としてはそもそも Unknown Unknown（何がわからないのかがわからない）ですし、人間が社会的な動物であり対面と非言語を求める仕様になっている点も手ごわさに拍車をかけています。こういったことをクリアしていくためにも、メソドロジーの開拓を進めていかねばなりません。長々と書きましたが、その一つが「タスク管理」であり、本書の執筆も開拓の一環なのです――

と、堅苦しい話はここまでにしましょう。

本書はいかがでしたでしょうか。読者の皆さまが自分なりに、自分のタスク管理というものを考えるきっかけになりましたら幸いです。また、本書をたたき台として、さらに色々なものが生まれていったら、作者としてこれほど嬉しいことはありません。

2024/09/11 吉良野すた（自宅でスズムシの音色を聴きながら）