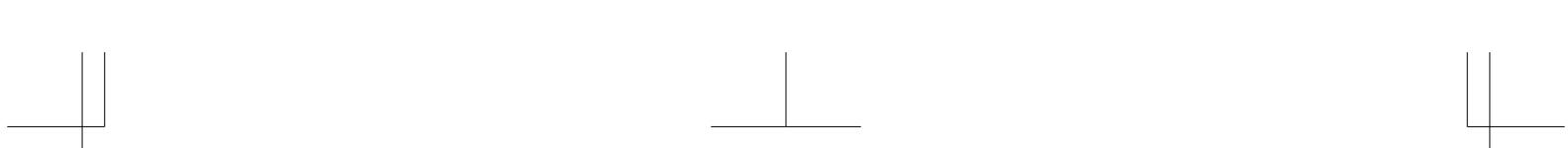


現場で使える!自動化入門

北崎 恵凡、山田 雄一、鎌田 誠、瀧川 大樹、
青木 敬樹、大野 泰歩、松岡 光隆、新山 実
奈子、高井 美佑、小崎 肇、澁谷 匠 著

2023-07-31 版 発行



はじめに

手にとっていただきありがとうございます。

私は某通信会社でインフラ設備の運用保守業務を担当し、日夜、現場に根ざした自動化・効率化に取り組み、日常の課題を解決しています。

SRE(Site Reliability Engineering)を目指して活動していますが、運用保守業務はいわゆる「コストセンター」と呼ばれ、サービスやシステムの信頼性を高める活動や付加価値を創造する活動にもあまりコストを掛けられません。

同様の悩みを抱えておられる方も多いのではないでしょうか？そこで本著では、既刊「現場で使える!Google Apps Script レシピ集」、「図解と実践で現場で使える Grafana」の執筆メンバーに加えて、本書は RPACommunity の有志が自動化のノウハウをまとめました。

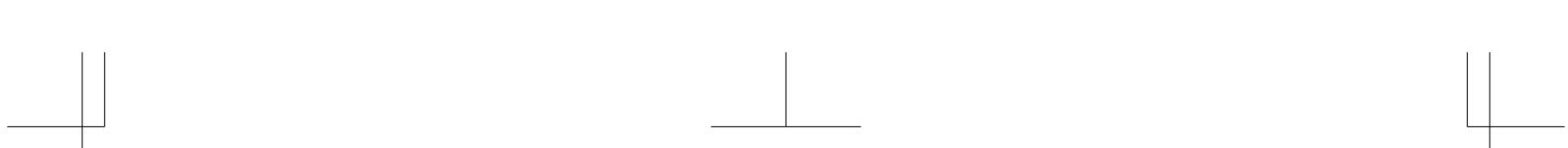
普段、現場で自動化に取り組んでいる経験が誰かのお役に立てれば幸いです。

2023年7月
北崎 恵凡

表記関係について

本書に記載されている会社名、製品名などは、一般に各社の登録商標または商標、商品名です。

会社名、製品名については、本文中では©、®、™マークなどは表示していません。



目次

はじめに	iii
表記関係について	iii
第1章 RPACommunity とは	1
第2章 RPA をいれたり自動化を検討する前に考えたい事	5
2.1 はじめに	5
2.2 世の「情報システム」にも色々ある	6
2.3 SIer によるスクラッチ開発の自社基幹システム	6
2.3.1 今のシステムを改修して隙間を埋める	6
2.3.2 RPA で隙間を埋める	7
2.3.3 RPA じゃないツールで隙間を埋める	8
2.3.4 今のシステムをそのまま諦めて使う	8
2.3.5 人を増やしてオペレーションで解決	8
2.4 パッケージソフトを使った自社基幹システム	8
2.4.1 今のシステムを改修して隙間を埋める	8
2.4.2 RPA で隙間を埋める	9
2.4.3 RPA じゃないツールで隙間を埋める	10
2.4.4 今のシステムをそのまま諦めて使う	10
2.4.5 人を増やしてオペレーションで解決	10
2.5 自前システム (SQLServer+AccessVBA とかでしっかり目の構成)	10
2.5.1 今のシステムを改修して隙間を埋める	10
2.5.2 RPA で隙間を埋める	11
2.5.3 RPA じゃないツールで隙間を埋める	11
2.5.4 今のシステムをそのまま諦めて使う	11
2.5.5 人を増やしてオペレーションで解決	11
2.6 自前システム (と呼べるかどうかわからないけど Excel+VBA ぐらいの もの) や、システムっぽいものが無いかも…?	11

目次

2.6.1	今のシステムを改修して隙間を埋める	12
2.6.2	RPA で隙間を埋める、RPA じゃないツールで隙間を埋める	12
2.6.3	今のシステムをそのまま諦めて使う	12
2.6.4	人を増やしてオペレーションで解決	12
2.7	色々と書いてきましたが…	12
第 3 章	RPA から始める DX 推進、成功の秘訣はこれだ！	15
3.1	はじめに	15
3.2	トヨタ生産方式と RPA	16
3.3	7つのムダとは	18
3.4	RPA 推進のベストプラクティス	20
3.4.1	現場主導	21
3.4.2	推進部門主導	21
3.5	魂の入った業務選定	22
3.6	終わりに	24
第 4 章	業務断捨離のすすめ	27
4.1	今日から私は庶務担当	27
4.2	はじまり	27
4.3	何をする（した）か	28
4.4	なぜ庶務業務は属人化しやすいか	28
4.4.1	庶務業務一覧の作成	28
4.4.2	当番表	30
4.4.3	情報の公開	30
4.4.4	業務確認書	31
4.4.5	報告フォーマット	32
4.4.6	Before / After シート	33
4.4.7	事務局の立ち上げ	33
4.5	WHY から始めよ	33
4.6	断捨離（できたケース）	34
4.6.1	根本原因が既に解決していた	34
4.6.2	他では簡単に実施、または、実施していなかった	37
4.6.3	機械化・自動化	37
4.7	断捨離（できなかつたケース）	41
4.8	悩み（悩んだこと）	42
4.9	気づき	42

4.9.1	良かったこと	42
4.9.2	反省点	42
4.9.3	副次的効果	43
4.10	さいごに	43
第5章	対象とする業務が決まったら	45
5.1	はじめに	45
5.2	「仕様書」	45
5.3	業務情報	46
5.3.1	必要な情報	46
5.3.2	任意な情報	47
5.4	入力情報	48
5.4.1	必要な情報	49
5.5	出力情報	51
5.5.1	必要な情報	52
5.6	処理情報	54
5.7	補助情報	54
5.8	結び	56
第6章	スマートマットをハックしてさらに便利にする	59
6.1	スマートマットとは	59
6.2	スマートマットライト	61
6.3	スマートマットライト1と2の違い	65
6.4	スマートマットライトをハックする	69
6.5	パケットキャプチャで通信の内容を確認	69
6.6	ブラウザの開発者ツールで通信の内容を確認	70
6.7	APIの仕様を推察	71
6.8	GASの実装	73
6.9	結果表示	82
6.10	さいごに	84
第7章	RPAによるリアルタイム処理	85
7.1	RPAのメリット	85
7.2	RPAのリアルタイム処理とは?	85
7.2.1	リアルタイム処理を知る	85
7.2.2	リアルタイム処理のメリット・デメリット	87
7.3	リアルタイム処理の実装	88

目次

7.4	リアルタイム処理の適用事例	90
7.4.1	製造業における工程間の在庫移動	90
7.4.2	製造業における完成品入力	92
7.5	リアルタイム処理の課題と解決策	94
7.5.1	エラーを素早く知るには	94
7.5.2	エラーの原因を知るには	95
7.6	リアルタイム処理の今後の展望	96
7.7	最後に	96
第8章	Raspberry Pi と GCP を使って、SMS の E2E 監視を実装してみた!	99
8.1	はじめに	99
8.2	E2E 監視の重要性と構築した経緯	99
8.3	監視の構成	100
8.3.1	SMS 送受信シミュレータ	100
8.3.2	監視ダッシュボード	101
8.3.3	データ連携用の Web API	102
8.3.4	予算の確保	102
8.3.5	社内のセキュリティガイドラインとセキュリティポリシーを満たす事	102
8.4	SMS 送受信シミュレータの実装	102
8.5	SMS 送受信スクリプトの処理概要	102
8.5.1	AT コマンド投入用の USB デバイスファイルの認識	103
8.5.2	SMS の送受信	103
8.5.3	Web API で送受信結果を Post	103
8.6	インターネット接続処理の概要	104
8.7	ドングルに設定必要な AT コマンド	104
8.7.1	APN の設定	104
8.7.2	APN のユーザとパスワードの設定	104
8.7.3	Packet Data Protocol(PDP) 有効化	104
8.8	Web API の実装	104
8.8.1	ファイアウォールで接続 IP を制限	105
8.8.2	認証キーの提供と有効期限の短時間化	105
8.8.3	入力値のバリデーション機能とプレースホルダの利用	105
8.8.4	リクエスト数とリクエストエラーの監視	106
8.9	自社内製の監視ダッシュボード側の実装	106
8.10	初期費用とランニング費用	107

8.11	初期費用について	107
8.12	ランニング費用について	108
8.12.1	GAE と Cloud SQL の費用	108
8.12.2	通信事業者の基本料 + データ使用料	108
8.12.3	SMS の送信費用	108
8.13	運用について	109
8.13.1	対処 1: 複数の監視シナリオで監視する。	109
8.13.2	対処 2: 複数の監視手段を用意する	109
8.13.3	対処 3: 複数のネットワークから API を叩く	109
8.14	まとめ	110
第 9 章 現場で使える Python 自動化入門		111
9.1	はじめに	111
9.2	頭をすっきりテスト駆動開発	111
9.2.1	はじめに	111
9.2.2	テスト駆動とは？	112
9.2.3	誰が為のテスト駆動？	113
9.3	みんなで読もう Sphinx	114
9.3.1	はじめに	114
9.3.2	Sphinx 入門	116
9.3.3	型アノテーションのすすめ	119
9.3.4	現場での活用方法	121
9.4	最後の救い log 取得	121
9.4.1	はじめに	121
9.4.2	log を設定しよう	121
9.4.3	DEBUG は DEBUG	122
9.5	さいごに	123
第 10 章 GAS と AWS を使って Web 操作をハックしてみた！		125
10.1	はじめに	125
10.2	情報収集の自動化 (GAS)	125
10.3	Web 操作の自動化 (AWS)	127
10.4	GAS から Lambda の呼び出し (GAS)	131
10.5	さいごに	131
第 11 章 フローで思い通りの結果を得るために大切なこと		133
11.1	はじめに	133

目次

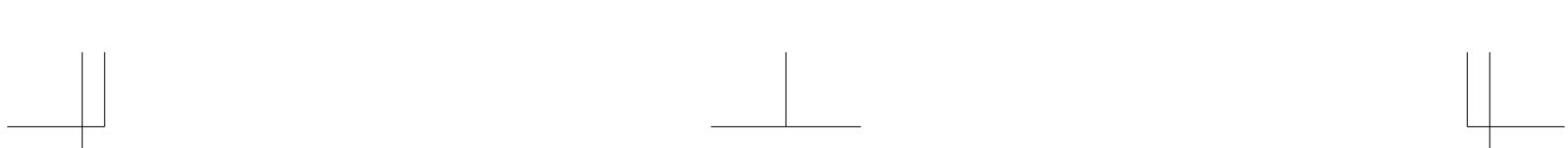
11.2	テストは、2段階に分けられる	133
11.3	Case アップロードしたはずのファイルが開かない	135
11.3.1	どんなフローか	135
11.3.2	このフローで期待される正常な動作はどういうものか	136
11.3.3	テスト機能を使って、フローの動作テストを行う	137
11.3.4	出力結果が正常な動作をするか確認する	142
11.3.5	「心当たり」をすべて確認し、問題を切り分ける	144
11.3.6	実行履歴を確認してみる	145
11.3.7	ファイルサイズを確認してみる	147
11.3.8	ファイルの作成アクションを確認してみる	149
11.3.9	ファイル コンテンツ は存在するのか?	153
11.3.10	どのアクションで「ファイルのコンテンツ」を取得するか	157
11.3.11	「添付ファイルのコンテンツを取得」アクションを理解する	158
11.3.12	フローをコピーし、「添付ファイルのコンテンツを取得」アクションを追加する	159
11.3.13	ID と ファイル識別子が取得できるか確認する	161
11.3.14	「ファイルの作成」アクションの設定し直す	163
11.3.15	フローをテストし直す	165
11.4	バグのない正しいフローを作るために	171
11.4.1	テスト項目ってどうやって洗い出すの?	171
11.4.2	テスト方法と想定結果、実際の結果を一覧表にして確認する	171
11.5	終わりに	172

第 12 章 普通の OL なら業務に役立つロボットを必ず作れます。もちろんこれは 体験談です。 173

12.1	自動化の手順も手腕も、三者三様にござりまする。	174
12.1.1	WinActor	175
12.1.2	UiPath	176
12.1.3	Power Automate (デスクトップ版)	177
12.2	ロボット作成時の注意事項を申し上げますね。	179
12.3	突然のツール変更も視野に入れた準備が必要です。	179
12.4	“最初から 100% 自動化を目指さない”が成功の鍵!	181
12.5	安易に下請けを使うと、危険です。保守ができずに、すぐ錆びてあっけなく壊れます。	182
12.6	最初は、半分だけ手伝ってもらうつもりで進めましょう	184
12.6.1	自動化の優先順位を決めましょう。	186

12.6.2	これから作成するメモのサンプル	186
12.7	現状の状態と、どの程度効率化できるかを可視化する	187
12.7.1	作業を塊に分けましょう。	188
12.7.2	これから作成するリストのサンプル	189
12.7.3	リストの修正タイム	190
12.8	およそ大体の RPA アプリで準備されている部品の簡単なまとめ	192
12.8.1	リストの修正タイム	192
12.9	黄金パターンをご紹介します (Excel 間のデータやり取り、ログイン等に 使用)	195
12.10	作業を見直します。	197
12.11	(そのまま乗せ換えることが正解とは限りませんよ？)	200
12.12	分解して再構成する。	201
12.12.1	メモの追記・追加タイム	202
12.13	分担を決める。	203
12.13.1	メモの追記・追加タイム	204
12.14	ツールを使い自動化/ロボット組み立てを行います。	204
12.15	普及型ローコード RPA ツールに有効な共通の構成	206
12.15.1	ロボット内に、雛形とするグループ構成および、その中にに入る定 型部品を格納	206
12.15.2	ロボットとその資料を入れておく定型のファイル構成を決めておく	207
12.16	Excel はお好きでしょう？ もう少し話しましょう。	207
12.17	終わりに これからはロボットと二人三脚で	212

筆者紹介	217
北崎 恵凡 (きたざき あやちか)	217
山田 雄一 (やまだ ゆういち)	217
鎌田 誠 (かまだ まこと)	217
瀧川 大樹 (たきがわ だいき)	218
青木 敬樹 (あおき ひろき)	218
大野 泰歩 (おおの やすほ)	218
松岡 光隆 (まつおか みつたか)	218
新山 実奈子 (にいやま みなこ)	219
高井 美佑 (たかい みゆう)	219
小崎 肇 (こさき はじめ)	219
濵谷 匠 (しぶや たくみ)	219



第1章

RPACommunity とは

RPACommunity 代表 Mitz (松岡 光隆)

IT を活用した業務の自動化や IT 推進全般に興味を持つ全国の有志が集うコミュニティで、2018 年 3 月に立ち上げました。

名称には「RPA」(ロボティック・プロセス・オートメーション) が付いていますが、RPA 専用のツールやサービスだけに限らず、様々なツールやプログラムも含め IT 技術全般を対象にした学びの場を提供しています。

発足から 5 年以上経ちメンバー数も増え続け 8,000 名を超えており、今現在でもコミュニティ内では様々な手法での業務自動化トークが飛び交っています。

そんな RPACommunity 内の「自動化ネタ」を、まだ RPACommunity を知らない方々にも知っていただきたいという想いを持つ有志が集って今回の書籍作成に至りました。

RPACommunity^{*1}では 2023 年 5 月時点で 300 回以上のイベントを実施しており、その資料はこちらに掲載しております。

^{*1} <https://rpacomunity.connpass.com/presentation/>

第1章 RPACommunity とは



図 1.1: connpass の QR コード

また、2020 年以降のイベントは全て YouTube^{*2}に公開しております。

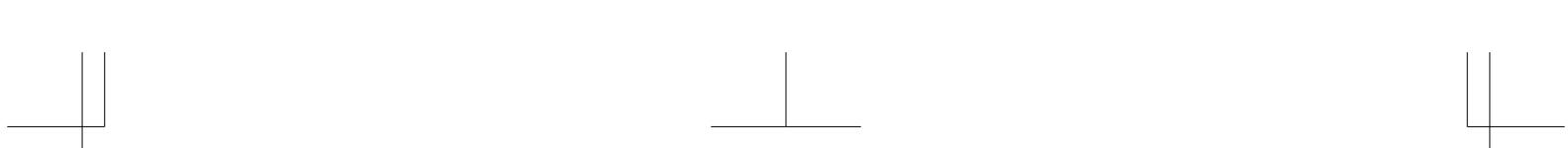
^{*2} <https://www.youtube.com/@RPACommunity>



図 1.2: YouTube の QR コード

ご興味ある方はぜひご覧ください。

本書はこれら RPACommunity での情報や登壇をベースに、RPACommunity 有志メンバーが書き上げております。



第2章

RPA をいれたり自動化を検討する 前に考えたい事

山田 雄一

2.1 はじめに

RPA を導入したり、自動化を検討したり、DX を始めたい、またそれ以前の話として現状の情報システムを改善したい……と思ったこと、ありませんか？ また、自分は乗り気でなくても、上から社命として言われてしまったり^{*1}したこと、ありませんか？

筆者としては、ここ数年（2015年、16年以降）は、RPA やノーコード、ローコードツールが一般的になった事により、単純なシステムリプレース以外に、取れる選択肢がずいぶんと増えたな……という印象があります。また既存システムにちょい足ししてみたり、既存システムのデータエントリー部分をノーコード、ローコードツールを使うようにすることで、こういった業務上の課題を解決できるかも……？ という機運が高まってきており、実際に活用例も多くなってきているように思います。

これらの RPA 系のツールは、「RPA 系のツールを使えば業務改善ができる」っぽく見える事が多いのですが、実際は打出の小槌のようにあらゆる業務改善や課題解決に効いてくれ、私達の業務を楽な方へ導いてくれるものなのでしょうか？

この章では、こういった時に何を判断基準にしたり、どういった懸念点があるのかをまとめてみました。RPA 導入以前の検討段階として、「RPA を導入したり、自動化を検討する」までに必要な検討の助けになり、ある程度^{*2}の道筋を示せれば良いなと思っています。

*1 こっちの方がもっと深刻である

*2もちろん筆者の個人的な考え方と独断と偏見によるものです

2.2 世の「情報システム」にも色々ある

世の中には色々な会社があり、色々な業務プロセスがあり、そして色々な情報システムがあります。

情報システムについては、色々なものがありますが、ターゲットを絞らないとこの手の話はついつい発散してしまうものなので、参考となりそうなケースを次の様に分けてみましょう。

- スクラッチ開発の自社基幹システム
- パッケージソフトを使った自社基幹システム
- 自前システム（SQLServer+AccessVBA とかでしっかり目の構成のもの）
- 自前システム（と呼べるかどうかわからないけど Excel+VBA ぐらいのもの）
- システムなし

これらについて、それぞれ次のような対応策を今回は考えてみましょう。

- 今のシステムを改修して隙間を埋める
- RPA で隙間を埋める
- RPA じゃないツールで隙間を埋める
- 今のシステムをそのまま諦めて使う
- 人を増やす

上で挙げたそれぞれの **システム x 対応策** について、それぞれの対応策の場合の検討ポイントをそれぞれ考えていきたいと思います。

2.3 SIer によるスクラッチ開発の自社基幹システム

SIer による受諾開発（スクラッチ開発）で納品され、運用されている自社基幹システムの場合、受諾開発という契約の特性を考える必要があるケースがあります。

2.3.1 今のシステムを改修して隙間を埋める

この対応策は、開発してもらった SIer に再度追加改修をお願いする事になることが多いですが、改修時のコストがかさむ事もあります。

また、SIer の開発人員も暇ではないので、継続的に追加改修をお願いしていない場合、改めて追加改修をお願いしたとして、実際に当時作っていた人が継続して対応してくれる

2.3 SIerによるスクラッチ開発の自社基幹システム

か、品質が担保^{*3}されるか、みたいな所が悩ましくなることもあります。だいたい裏側は分かんないので……

また筆者の実体験として、継続的に改修を行っているシステムであれば、継続改修されているのでコストもそこまでかかる事もありますが、しばらく触っておらず、時間が経ってから改修を行うようなケースでは、過去のことを思い出す/既存システムの解析にもコストがかかるので、結構地味に大変だったりします。(大変なときは総じて見積もりに跳ね返ります。)

2.3.2 RPAで隙間を埋める

現行システムの改修と比べ、RPAで隙間を埋めた時の最大の利点は、「システムの内部を修正していないのに、業務オペレーションは簡単にできる(かもしれない)」点です。

RPAは良くも悪くも「画面操作をするツール」で、システムの内部を修正しませんが、これはケースによっては大きなメリットになります。

○うれしいケース1

たとえば、社内でISO認証などの外部監査がある認証を取得しており、この監査情報の取得や集約がシステムに組み込まれている場合、社内システムの改修をすると、改修箇所(時によっては監査箇所全般)の処理内容について、再度監査が必要になってしまうケースがあります。

ところが、RPAでの業務改善を行うとあら不思議! 社内システムから見ると、「画面操作をしているだけ」ですね!

これならどんな改善をしても安心!

○うれしいケース2

たとえば、最初のスクラッチ開発はSIerに頼んだが、その後の改修については他社に頼んだり、自分たちで行うケースもあるかもしれません。(プロジェクト炎上したりするとありがち)

よく受諾開発の契約書にはあるあるなのですが、システムを開発納入した会社以外が改修した場合は、納入会社はシステムの瑕疵担保責任をその後一切追わないとする契約となっているケースがあったりします。(責任を負えなくなるのでそれはまあ当たり前といえば当たり前)

こういった場合に、システムを直接改修するのではなく、RPAを使って改善を行った場合、やはり社内システムから見ると、「画面操作をしているだけ」になるのでシステム利用時の瑕疵担保責任が継続され、そういう点でも安心だったりします。

^{*3}ここで言っている品質とは単に「バグがない」というだけではなくて、ソースコードの保守性であったり、応答速度であったり、データ量であったりといった非機能要件も含みます。

第2章 RPA をいれたり自動化を検討する前に考えたい事

2.3.3 RPA じゃないツールで隙間を埋める

RPA じゃないツールで隙間を埋める というのもケースによっては非常に効きます。
基幹システムを作ったはいいが、そんなに今後長いことこれを使わないと思うので、
システム自体に改修は入れたくない、でも現状の業務は厳しいので改善をしたいとか、
RPA ではやりにくい大量データの捌きを行った上で、処理を実現する必要がある……
といった場合に、たとえば ETL ツールがハマったりする事があります。

ただ、RPA じゃないツールの場合において、データの I/O は API 経由だったり、DB
への直接アクセスが手段になりますが、パフォーマンスが劣化したり、本来の基
幹システムからのデータ I/O ではない I/O (特に書き込み処理) が発生するので、既存シ
ステムとのそういう所の統制を取りたりするのが思った以上に大変なのと、当然ながら
前述したような監査があるシステムの場合、悩ましい事になります。

2.3.4 今のシステムをそのまま諦めて使う

改修コストとそれにともなって得られるメリットが見合わない場合はこうなりがちで
す。やむを得ないです……

2.3.5 人を増やしてオペレーションで解決

人を増やして対応するはある意味最強の一手段です。ハネムーン係数なども改善するか
かもしれません。やったね！ 業務が強くなるね！

※ただしボトルネックに人が増えない場合はなにも変わらず教育コストだけがかさむ
ケースも……

2.4 パッケージソフトを使った自社基幹システム

パッケージソフトの場合にはパッケージソフトならではの検討ポイントがあるのでは？
と筆者は考えています。

2.4.1 今のシステムを改修して隙間を埋める

そもそも、大規模な基幹パッケージ (例えば SAP とか) を入れていて、既に運用が回っ
ているフェーズならば、システムを改修したいと考える会社であれば、ほぼほぼアドオン
によるカスタマイズが入った状態になっているはずで、その上で現在のシステムと業務に
ギャップが発生しているので、そこに更にアドオンによるカスタマイズを付け加える形
(付け加えたいと考えている状態) になっているのではないかと考えられます。

2.4 パッケージソフトを使った自社基幹システム

ただパッケージソフトのカスタマイズというのは、一般論でいうと2割の処理にカスタマイズを入れるのであればフルスクラッチで作るのとコストが変わらないと言われているぐらいのもので、非常に開発コストがかかるものです。

また、一般的な大規模パッケージのベストプラクティスは「極力カスタマイズなしで使い、業務をパッケージに合わせる」となっている事が多く、そういった意味でも改修は悪手となりがちですが、現場要望もあって、情シスとしても押し負けてしまうことが多く、なかなかそういった理想的な形にはならない実情がありますね……

■コラム：パッケージソフトにおいて、カスタマイズがないとなにが嬉しいのか？

カスタマイズなしで使うと何が嬉しいかというと、パッケージのバージョンアップの際にカスタマイズはすべて見直しが要る事が多く、カスタマイズがなければパッケージのバージョンアップがサクサクで進むので、維持管理コストがずいぶん軽くなるのです……（カスタマイズがあればあるほど、再検証や再開発のリスクが上がる）

なお、大規模パッケージを使う時にも、会計系の箇所については原則として一切カスタマイズを入れないケースが多く、これは前述の「監査がある場合」に近い話が待っている事が多いです。

具体的には、こういったパッケージを入れる時の大きなメリットの一つとして、会計監査の省力化があげられます。

フルスクラッチで会計システムを構築した場合、大きな企業さん^{*4}では担当の会計事務所にシステムの監査をして貰う必要があり、これが経理担当の悩みのタネだったりするのですが、有名な会計パッケージであれば、それをそのまま素直に使ってさえいれば、ある程度の会計システムとしての正しさをパッケージが保証してくれるため、会計的にチェックすべき点が相当シンプルになり、会計事務所が嬉しいので、経理担当も非常に嬉しいんだそうで……

2.4.2 RPA で隙間を埋める

こういったケースでもやっぱりRPAはオイシイです。だって、画面しか触らないですからね。

また、前述の「2割の処理にカスタマイズを入れるのであればフルスクラッチで作るのとコストが変わらない」みたいな話を、パッケージソフト側ではなく、その外で持てるよ

^{*4} 上場してたりだとか…

第2章 RPA をいれたり自動化を検討する前に考えたい事

うになるので、そういう意味でも良いです。

加えてコスト面でも有利なことがあります。フルスクラッチ開発と比べても、大規模パッケージのカスタマイズができる人というのは、そのパッケージに精通したコンサルレベルのSEであることが多く、そういう人にお願いするとどうしても単価そのものが高くて、結果これが開発コストとして高くなりがちなのですが、RPAであればそういった高単価の人をアサインせざとも、業務改善が実現できるかもしれません。これは嬉しいポイントですね。

2.4.3 RPA じゃないツールで隙間を埋める

RPA じゃないツールも前項の「RPA で隙間を埋める」と同じ観点でオイシイです。ただ、内部データを直接 API で触る事になるので、投入したデータがパッケージ上でうまく処理されることを検証しておく必要があります。(筆者は実際にそういうテストをやったこともあります……笑)

2.4.4 今のシステムをそのまま諦めて使う

人生諦めも肝心です。投入コストより回収リターンのほうが少ない場合は「何もしない」が最適解だったりしますよね…(特にパッケージは改修コストがかかりがちなので、ノーカスタマイズで使って人が合わせる方が良かったりすることもままあります。)

2.4.5 人を増やしてオペレーションで解決

人こそパワー！(※前の「スクラッチ」の話と全く論点は変わらないです)

2.5 自前システム (SQLServer+AccessVBA とかでしつかり目の構成)

さて、コストをがっつりかけてSIerに基幹システムを作ってもらったりはしておらず、パッケージソフトを入れていない会社というのも意外に多いものです。(私も前職は社内SEだったのですが、このパターンでした。)

2.5.1 今のシステムを改修して隙間を埋める

自前システムだと改修しやすい面と、改修が難しくなる面があります。

改修し易いポイントは、自前で作っているのでコスト面や対応時間がコンパクトですみます。改修が難しいポイントは、そもそもそのために人数を多く割いていなかつたりする

2.6 自前システム（と呼べるかどうかわからないけど Excel+VBA ぐらいのもの）や、システムっぽいものが無いかも…？

のでマンニングが難しい事があるのと、少人数で開発をする事になるので、そんなチームにエキスパートがごろごろいるわけでもなく、結果として技術的負債が蓄積しやすい点でしょうか。（そのため、改修コストがどんどん上がっていく傾向にあります。）

2.5.2 RPA で隙間を埋める

RPA については自前システムだと悪手になるケースが多いです。（自分たちでシステムの表も裏も全部ハンドリングできている筈なので、RPA を使うより直接改修したほうが普通は望ましい筈）さて、RPA を使ったほうが良いケースはあるのでしょうか？

たとえば、自前システムを構築している基盤に無い機能を RPA で簡単に付け足せる場合などは、RPA を導入するメリットがありそうです。（正直、前職ではこのシナリオで少し導入を検討したことがあります。結局入れなかったのですが。）

2.5.3 RPA じゃないツールで隙間を埋める

外部データ連携などをする場合や、新しい機能を実現したい場合など（最近だと AI とか？）、RPA じゃない他のツールが刺さるケースもあるかもしれません。

2.5.4 今のシステムをそのまま諦めて使う

諦めも肝心です。でもどこかで重い腰をあげないといけないタイミングというのも……

2.5.5 人を増やしてオペレーションで解決

もうここまで来ると、既存のシステムを守るという観点よりは、新しいシステムや業務プロセス改善を実現するために人を増やす必要があるかもしれません。

2.6 自前システム（と呼べるかどうかわからないけど Excel+VBA ぐらいのもの）や、システムっぽいものが無いかも…？

小さい会社で簡単なシステムを作つて運用しているケースも多いでしょう。ExcelVBA などがよく活躍するケースです。あとは、「情報システム」と言われて思い浮かぶものが無いケースもあるでしょう。

でも、IT が基幹システムのようにしっかりと、連動して動くようになっていなくても、IT を使つた業務プロセスが回つているケースなども広義のシステムと言えるかもしれません。

第2章 RPA をいれたり自動化を検討する前に考えたい事

2.6.1 今のシステムを改修して隙間を埋める

簡単なシステムや小さなマクロに分割されているほど、改修というよりは1つのシステムにまとめていくほうが良いかもしれません。

2.6.2 RPA で隙間を埋める、RPA じゃないツールで隙間を埋める

ここはもうツール導入の可否みたいな話で論点がまとまっています。

そもそもの話として、RPA やその他のツールで隙間を埋めるにせよある程度の粒度のシステムになってからな気がします。その際に、ハンドオペレーションが入っている箇所をシステム化し、一気に自動化できるかもしれません。

なお、それは一般的には「DX」と言います。

改善の伸びしろがいっぱいある状態は、ある意味では良い事なのかもしれません。

2.6.3 今のシステムをそのまま諦めて使う

まあそんな判断になることもあるかもしれません。投入できるコストは有限ですし。

2.6.4 人を増やしてオペレーションで解決

これぐらいの規模感だと、「増やせるものなら増やしたい」というのが本音になりそう。

2.7 色々と書いてきましたが…

色々なケースを挙げ、RPA が効きやすそうなケース、効きにくそうなケースを様々に検討してみましたが、いかがだったでしょうか？

ちなみに情報システムの再構築における様々な論点については、「SEC BOOKS：システム再構築を成功に導くユーザガイド 第2版～ユーザとベンダで共有する再構築のリスクと対策～」という書籍があったのですが、絶版になってしまいました。ただ、PDF は無料公開されていて、誰でも読めるようになっており、ここに様々な参考となる情報が掲載されています。

<https://www.ipa.go.jp/archive/publish/secbooks20180223.html>

RPA での改善を検討する際にも、本来的なアプローチとしては

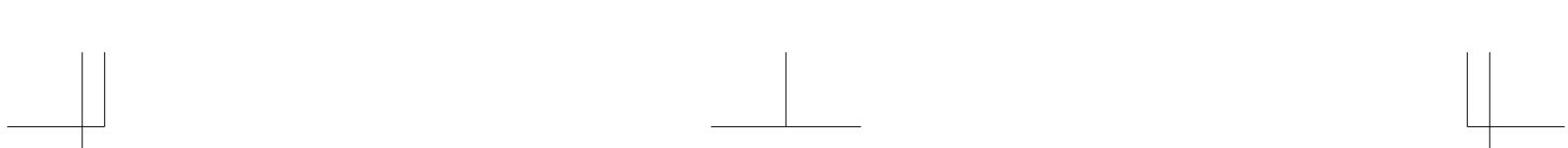
- システムリプレースや改修するほどではないね
- RPA で改善をしようね

2.7 色々と書いてきましたが…

という検討結果を経て、その後から RPA での改善が始まるのでは？ と思いますので、こういったソースにも当たってみて、一読した上で論点整理をしておくと、少し導入がスムーズになるかもしれません。⁵

この記事が最適な IT システム構築、ツール選定戦略の一助となれば幸いです。

⁵ 上長に説明するときとか、資料があると説明力が段違いだったりします



第3章

RPA から始めるDX推進、成功の秘訣はこれだ！

瀧谷 匠

3.1 はじめに

皆さまこんにちは。株式会社 ASAHI Accounting Robot 研究所の瀧谷です。ここからは私が前職の製造業で16年間研鑽し、現在も日々勉強をしているトヨタ生産方式とRPAのお話から始まり、RPAの導入が決まった後の社内推進体制をどうするか、RPA推進で気を付けるべきポイント、自働化したい業務をどのように優先順位をつけていくかについて、そしてRPA推進がどのようにDXにつながっていくかをお話していきます。

第3章 RPA から始めるDX推進、成功の秘訣はこれだ！

自己紹介

詳細な自己紹介


 しぶ や たくみ
澁谷 匠 / μ

株式会社 ASAHI Accounting Robot 研究所
テクニカルエバンジェリスト / DXアドバイザー



全ての人の仕事を楽にするをモットー
製造業16年、うち4年間DX・RPA推進を経験

認定DXアドバイザースペシャリスト
DXの「何を・どこから・どうやって」を共に推進

図 3.1: 自己紹介

会社紹介

あさひ会計グループ内の業務効率化を目的に結成された我々株式会社 ASAHI Accounting Robot 研究所（通称：ロボ研）は、現在、全国、世界の企業に向けて”ヒトとロボット協働時代を推進する”というミッションを掲げ、遊びごころを持って全力で取り組んでおります。RPA や AI を中小企業が身近に活用する時代になることを目指しています。

私のパートでは通常とは異なる漢字を意図的に使用しています。言葉に意味を持たせることで、皆さんによりわかりやすくイメージをもっていただきたいからです。以下にその解説をいたします。

- 自働化 → RPA による業務改善は自動（自ら動く）ではなく自働（自ら働く）という意味。RPA がただ動くのではなく人に代わって働くということを表している
- 観る → 目で見るだけではなく、しっかりと観察して分析するという意味

3.2 トヨタ生産方式とRPA

ここからはトヨタ生産方式とRPAというテーマでお話をします。

このテーマを聞いて、きっと皆さんこう思ったんじゃないでしょうか。

- 生産方式って製造業の話で自分たちには関係ないよね

3.2 トヨタ生産方式と RPA

- 生産の仕方の話と RPA って何の関係があるの？

トヨタ生産方式は決して製造業のみに関係する話ではありません。また、RPA 推進に大いに役立つヒントがあることを私は身をもって実体感、実体験しております。

まずはトヨタ生産方式についてトヨタ生産方式の原点は、自動織機（自動的に糸を織物にする機械のこと）の開発で有名な豊田佐吉氏が研究してきた織機に“縦糸が切れると自動的に機械が止まる”といった、異常が起きると機械が自動的に判断して止まる機構を機械に組み込んだ事に原点があります。

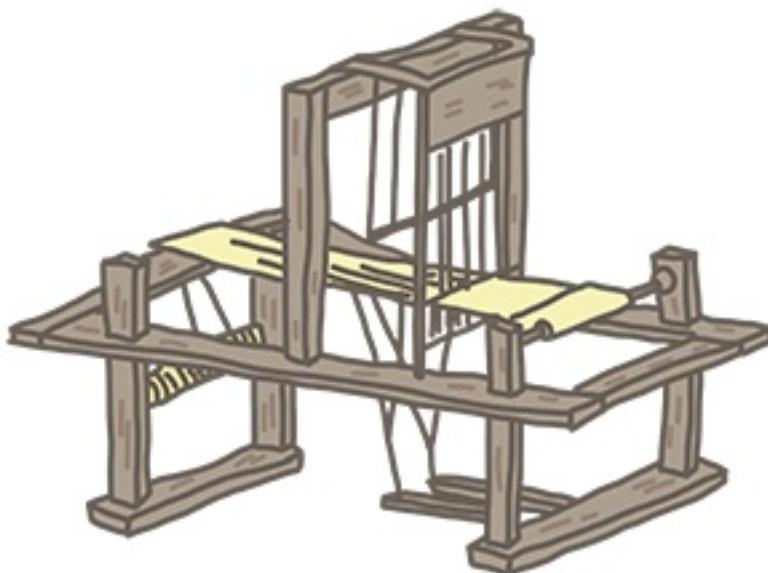


図 3.2: 機織り機イメージ (イラストは人力のもの)

その後不良品を造らない、人を機械の番人にしないといった「ニンベンのついた自働化」の考え方と、佐吉氏の息子でトヨタ自動車の創業者である豊田喜一郎氏が 1937 年ごろに提唱した必要な時に必要なものを必要なだけつくるといった「ジャストインタイム」の考え方方が「トヨタ生産方式の二本柱」と言われています。これを戦後、具現化したのはトヨタ生産方式の生みの親である大野耐一氏です。大野耐一氏は、資金や資材の乏しい中、日本と比べて、非常に高い生産性の欧米の企業に打ち勝つには「ニンベンのついた自

第3章 RPA から始めるDX推進、成功の秘訣はこれだ！

「効率化」と「ジャストインタイム」の実現しかないと考え、現場にてその具現化にむけた研究を重ね、効率化によるムダ徹底的排除とアメリカのスーパーマーケットにヒントを得た「後工程引取り」の実施によりジャストインタイムを実現しました。

異常（エラー）があると自動的に止まる、不良品（不良書類・不良データ）を造らない、人を機械（パソコン）の番人にしない、必要な時に必要なものを必要なだけ（ロボットを）つくる、これらキーワードを聞いて私は真っ先に「これこそRPAだ！」と思いました。

- 1. 異常があると自動的に止まる
 - エラーが出たら例外処理でエラー内容をメールで報告したり、ウィンドウを閉じて終了する
- 2. 不良品を造らない
 - 不良書類・データを造らないよう、事前検証や照合をしっかりする。ミスも起こらない
- 3. 人を機械の番人にしない
 - ロボットのスケジュール実行で無人化
- 4. 必要な時に必要なものを必要なだけ
 - 深夜や休日であっても、24時間365日ロボット実行できる

どうでしょう。これらのトヨタ生産方式の重要なポイントは、RPAの機能や全社展開に相通じるものがあるという気がしませんか。言い方を変えると、RPAを導入し業務効率化を進めていく際、このトヨタ生産方式の概念が必ず“推進の羅針盤”になると考えます。

事実、トヨタ生産方式は製造業のみではなく業界問わず、例えば医療関係や物流業、小売業、はたまた片づけや貯金などにも応用されています。

皆さんは毎日の仕事の中で「付加価値を生む仕事」ができていますでしょうか。俯瞰して観てみると、意外とできていないものです。私自身普段の仕事の中にムダはまだまだ潜んでいると感じます。

ムダ・ムリ・ムラ（3Mと言います）の改善手段はいろいろありますが、その中の一つとして付加価値のない業務はRPAという名のデジタルレイバーに任せて、人は人にしかできない創造性のある仕事に注力できるようにしていきましょう。

3.3 7つのムダとは

さて、前章で3M（ムダ・ムリ・ムラ）の話が出たところで、ここからはもう少し掘り下げて7つのムダというテーマでお話をします。

トヨタ生産方式には7つのムダという考え方があります。これらのムダが自分のまわりに無いか常に目を光らせて、見つけたら徹底的に排除しなければなりません。

3.3 7つのムダとは

トヨタ生産方式「7つのムダ」

- ・【か】加工のムダ
- ・【ざ】在庫のムダ
- ・【つ】造りすぎのムダ
- ・【て】手待ちのムダ
- ・【と】動作のムダ
- ・【う】運搬のムダ
- ・【ふ】不良・手直しのムダ

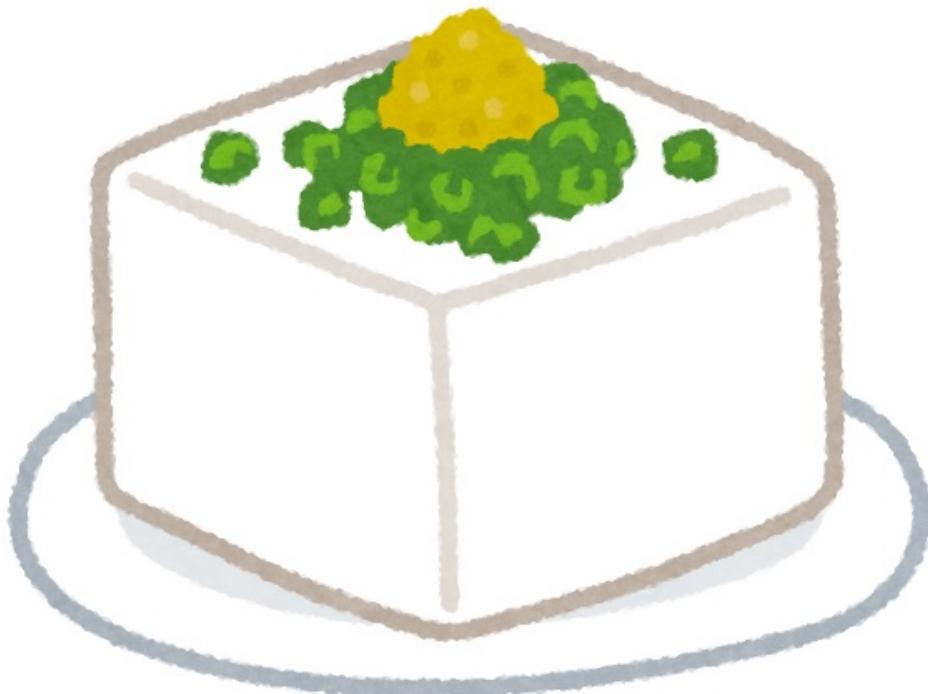


図 3.3: 飾って豆腐 (かざってとうふ)

それぞれの頭文字を取って「飾って豆腐（かざってとうふ）」と覚えます。一見するとどれも製造業の現場ならではのムダのように見えますが、事務・バックオフィス業務に置き換えるとこのように考えることができます。

- ・加工のムダ →ファイル、資料作成に凝りすぎる（レイアウトやアニメーション）
- ・在庫のムダ →不要な書類・データを大量に保管している
- ・造りすぎのムダ →必要以上に資料を作っている（似たような資料を複数の担当者

第3章 RPA から始めるDX推進、成功の秘訣はこれだ！

で作っている)

- 手待ちのムダ →他部署や他の担当者からの情報を待っている
- 動作のムダ →ファイルや情報を頻繁に探している
- 運搬のムダ →仕事の導線が悪い
- 不良・手直しのムダ →資料の作り直しが発生している（指示が悪い）

会社勤めの方であれば誰もが思い当たるところがあると思います。このような付加価値を生まない作業を徹底的に排除する、そのためにはムダを見つける感性と、見つけたムダを即座にカイゼンする実行力、さらに諦めない気持ちが求められます。またこれらはRPA を社内で推進していく上でも気を付けなければならないポイントでもあります。

今やっている業務をそのままRPA で自働化するのではなく、初めに業務の棚卸をしてその業務が本当に必要なか、業務手順にムダがないか、複数の担当者で同じことをやっているのか、やり方を変えると一本化・標準化できないかという目線で観て、どうしてもやらなければならぬ業務のみを自働化する（私が講演でいつもお話をする KAIZEN の三原則、ヤメル・ヘラス・カエルのことです）。このように7つのムダの観点で業務を見直すことで、その後の自働化を最短で安定したものにすることができますし、その活動自体が部署や担当者のレベルアップにつながり考え方があわってくと実感しています。

カイゼンにも、RPA による自働化推進にも終わりはありません。トヨタ生産方式には「カイゼンした今が一番悪いと思え」という言葉があります。RPA の社内展開がうまくいってよかったなど安心するのではなく、今が一番悪い・もっと上を目指すんだという意気込みで今日も仕事に取り組んでいきましょう。

業務効率化、それはまさに“終わりなきカイゼン”です。

3.4 RPA 推進のベストプラクティス

さて、ここからは「RPA の進め方、現場主導か推進部門主導か」というテーマでお話をします。

なぜ今さらこんな話をするのかと申しますと、私はエバンジェリストとして、また北海道のRPA 推進者として、各種様々なイベントや勉強会等で登壇する機会が多いのですが、その際毎回といっていい程議題にのぼる内容があります。それが

「RPA を推進するベストプラクティス（最善の方法、最良の事例）は何か」
ということです。

結論から申しますと、どのような進め方がベストかは各会社の風土やIT リテラシー、推進者の立場やスキルなどなど、パターンによって変わります。その会社や企業ごとに進め方は違ってよいですし、マッチする進め方が必ずあります。

企業内のRPA 推進には大きく分けて「現場主導」と「推進部門主導」の二つがある

3.4 RPA 推進のベストプラクティス

と言われています。私が考えるそれぞれのメリット・デメリットは以下です。

3.4.1 現場主導

良い点

- RPA を適用する業務フローを一番わかっている人がロボットを作成できる
- 自分（所属担当）の業務改革に直結できる

難しい点

- 専任者を置きにくい（兼任にするとロボット作成時間の確保が難しくなりがち）
- ロボット作成者が多く必要なので、育成に時間及び費用がかかる
- ロボットの管理が大変（野良ロボットができやすい）

3.4.2 推進部門主導

良い点

- 専任者が集中してロボット作成できる
- ロボット作成者の育成は必要最低限の人数で済む
- ロボットの集中管理が可能

難しい点

- 業務の詳細を把握してロボット作成しなければならないため、現場の協力が不可欠となる（業務フローやマニュアルの作成が必須で、RPA 導入目線での業務変更や、詳細なヒアリングが必要）
- 現場の担当者は RPA を「自分ごと」としてとらえにくい

いかがでしょうか。どちらの進め方も一長一短があります。それぞれの良い点・難しい点を自社に置き換えたときにどうかしっかりと吟味して、どちらの方法で RPA を進めていくことが自社のベストプラクティスであるかをジャッジすることが肝要です。

私は前職の製造業で RPA 推進をしていた時は「推進部門主導」を選択して進めました。現場主導を選択しなかったのは大きく以下 3 つの理由があります。

- 理由①
 - 各現場がとにかく忙しく、現場主導で進めた場合 RPA 専任者を置くことはできず兼任となり、そうなった場合ロボット作成時間の確保が厳しいのは明らか。

第3章 RPA から始めるDX推進、成功の秘訣はこれだ！

- 理由②
 - 過去に別のITツール展開を行った際に現場主導を選択し推進を試みたが、遅々として進まずうまくいかなかった事例があったため。
- 理由③
 - 業務効率化の目標を立て、達成するための実績管理がしやすい。また、ロボット作成工数のタスクマネジメントが容易にできる。

もしこれらに心当たりがあれば、推進部門主導で進めるのも一考です。

RPA推進で悩みはつきものです。トヨタ生産方式では「困らんやつほど、困ったやつはおらん」と言います。悩むということは、問題を見つけているということ。問題を見つけなければ解決策を考えることはできません。一人で抱え込み、悩みは皆で共有し皆で解決していきましょう。

3.5 魂の入った業務選定

さて、ここからは「導入推進の切り札！ らむだ式！ RPA適用業務選定のすべて」というテーマでお話をします。

イベント・セミナー等に参加して、皆さんと情報交換・交流していると、このような悩みをとても多く聞きます。「RPAは費用対効果が出せない（出すのが難しい）」「自働化する業務がなかなか見つからない」

私は今まで、イベント登壇など機会があるごとに幾度となくお話してきましたが、あらためて私が前職でRPA推進時に実際に使用していたRPA適用業務選定基準について詳しくご紹介します。

RPAをスタートする際に「どの担当のどの業務からどういう順番でロボットを作成していくか」、これはRPA推進者であれば誰もが必ず最初に悩むところです。例えばロボットの作成は簡単だが効果（費用対効果）が少ない業務といった場合は、ロボット作成工数（時間）のほうが目立ってしまいRPA推進プロジェクト自体が効果がないものとみられてしまったり、逆に費用対効果はとても大きいかもしれないがロボット作成に〇か月要します、というような業務だと、実際に費用対効果が出るまでの間、活動していないようになってしまい、RPA推進プロジェクト自体の存続に関わってしまうことにもつながりかねません。つまり、①どの業務をRPAで自働化するか、②その業務をどの順番で進めてゆくか（自働化する優先順位）この2点はRPA推進の際、非常に重要なポイントとなります。と、わかってはいるものの、研修・セミナーを聞いたり、関連書籍を読んでみたり、そういう中に記載のあるような開発標準例ではなかなか自社に合った方法が見当たらぬといふ事も多いのではないでしょうか。

そんな時はこれからお話しする「らむだ式！ RPA適用業務選定」を使っていただくこ

3.5 魂の入った業務選定

とを強くおススメいたします。なぜこんなにも自信をもっておススメできるかというと、机上で考えたものではなく、「RPA 推進者と RPA を適用する現場が一緒に悩み、考え、何度も改良を繰り返して作り上げた適用業務選定」だからです。まさしく「魂の入った業務選定」です。ポイントは以下です。

1. RPA で自働化したいという業務を 7 つの項目で点数付けし業務を数値化
2. 点数は 1 点～5 点の 5 段階評価（最高得点は 7 項目 × 5 点 = 35 点）
3. 点数が偏る場合は採点条件を常に見直す（現在 7 代目）

らむだ式RPA適用業務選定

	項目	定義	1点	2点	3点	4点	5点
1)	開発工期	新規開発or既存ロボット改造度合を考慮した開発工期（トライ＆エラー期間含む）	3か月以上	1か月以上～3か月未満	2週間以上～1か月未満	1週間以上～2週間未満	1週間未満
2)	作業頻度	<月当たり> 作業回数×作業人数	1以下 <small>例) 月1回×1人</small>	2～3 <small>例) 都度(月2～3回) 月1回×1人</small>	4 <small>例) 週1回×1人</small>	5～20 <small>例) 毎日×1人</small>	21～ <small>例) 每日×2人以上</small>
3)	業務工数	<月当たり> 2) ×1回あたり作業時間（分）	~200 <small>例) 年間10h</small>	201～420 <small>例) 年間50h</small>	421～2,400 <small>例) 年間100h</small>	2,401～4,800 <small>例) 年間500h</small>	4,801～ <small>例) 年間1,000h</small>
4)	業務内容	全体業務フローに対する人固有の判断割合 (100%は開発不可)	80%～100%	60%～79%	40%～59%	20%～39%	0%～19%
5)	業務手順書	作成依頼から業務手順書を提出までの期間（※RPA開発部分の業務手順書）	3か月以上	1か月以上～3か月未満	2週間以上～1か月未満	1週間以上～2週間未満	すでにある or 1週間未満
6)	システム横断	業務手順がExcelのみで完結しない	Excelのみで完結 (マクロで対応可)				Excelのみで完結 しない
7)	標準化	業務手順パターン数、分歧条件数をもとに仕様確定までに要する準備期間 (例：作業者別に手順やフォーム異なる。 ※業務別に手順が異なる。 ※業務ヒアリングによって顧在化する標準化されない業務手順や複雑な分歧条件の潜在的リスク＝推進部門の過去の導入実績/経験をもとに算定)	3か月以上	1か月以上～3か月未満	2週間以上～1か月未満	1週間以上～2週間未満	1週間未満

図 3.4: らむだ式 RPA 適用業務選定

具体的な内容を観ていきましょう。

- 1. 開発工期 予想されるロボット開発にかかる日数
 - 既存ロボットを改造することで対応できると工期が短くなり高得点となる
- 2. 作業頻度 月当たりの作業回数×作業人数
 - これは定義そのままですね
- 3. 業務工数 月当たりの作業時間
 - 2. に 1 回あたりの作業時間をかけます
- 4. 業務内容 業務全体に対する人の判断割合
 - 人の判断が入る業務でも、その前後を自働化するという方法があります
- 5. 業務手順書 業務手順書があるか、なければ提出までどれくらいかかるか

第3章 RPA から始めるDX推進、成功の秘訣はこれだ！

- 業務が担当者の頭の中だけにあるようでは困るので業務手順書は必須です
- 6. システム横断 業務が複数システムを横断しているか
 - Excelだけの業務だとRPA以外でも効率化できます
- 7. 標準化 業務を標準化するまでにかかる日数
 - 業務パターンが多岐に渡るとロボットが長大になるので事前に業務標準化をします

非常にシンプルかつ明瞭でわかりやすい7つの項目のみにあえて絞ってあります。

この業務選定を使用した場合、大きな効果が2つあります。1つはその業務がどういう基準で選定されたかが数値という形で一目でわかりますので、点数が低くてロボット作成対象から外れてしまった場合でも、なぜそのようなジャッジになったのか納得のいく回答・説明ができるということ。もう1つはRPA推進の費用対効果などを上長や役員へ報告する際に明確な根拠を提示できるということです。事前に基準に合わせて選定をしておけば、効果の算出は非常に容易です。

トヨタ生産方式では「百聞は一見にしかず、百見は一行（行動）にしかず」と言います。100回聞くよりも1回見る方がよくわかる、というのは皆さまご存じの通りですが、さらに一步踏み込んで100のことを見たり、100回見るよりも、自分で1回行動してみることが大事という意味です。

今回私が紹介した「らむだ式！RPA適用業務選定」は私自身の経験からまとめたものでありあくまで一例です。もし冒頭で記載したように適用業務がなかなか見つからずに悩まれているようであれば、まずはこちらをたたき台にして自社のオリジナル選定基準を作成することで、RPAの社内展開をもう一步前に進める機会になると思います。

3.6 終わりに

ここまでトヨタ生産方式とRPAについて、社内のRPA推進体制を決めて、集まった自働化対象業務を適用業務選定を使って明確な根拠を持って自働化していくことの重要性についてお話ししてきました。

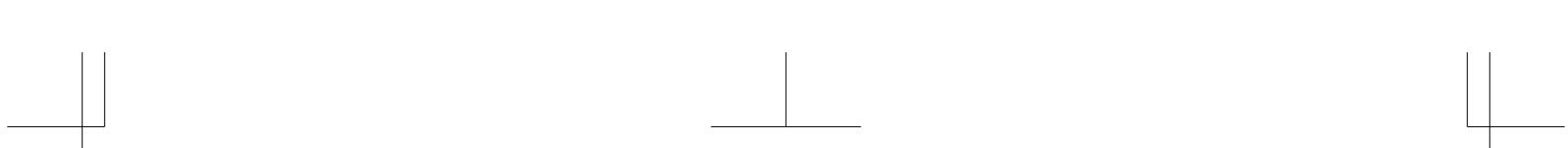
私の4年に及ぶRPA/AI-OCRの全社展開成功経験、またたくさんのRPA推進者との情報交換でわかったこと、それは企業のDX推進には現地現物で「デジタル化について広く正しく周知することと「デジタル化のハードルを下げる」ことが欠かせない、ということです。つまり、RPAを“きっかけ”として、DXの一歩を踏み出すことにより、RPAがDX推進の基礎土壌となってデジタル化のハードルが下がり、休むことなく次々にデジタル化に取り組むことで従業員の意識が変わります。この過程を進めていくことでデジタルを使用した企業変革（DX）は必ず実現できます。まさしくこれは「人に優しいデジタル化」と言えます。RPAもDXも進めるのは「人」です。人を中心に、人を大切

3.6 終わりに

に進めていくことがとても重要です。

私のモットー、活動の原点は「全ての人の仕事を楽にする」です。この気持ちは RPA と初めて出会った 2018 年から一貫しています。初めてロボットを作ってそれが完成して動いたとき、私は「これで周りのみんなの仕事を楽にできる！みんなを助けることができる！」と感動したことを今でも鮮明に覚えています。全社員が自分だけではなく、隣の・同じチームの・同じ部のメンバーの仕事を楽にしていく、これこそが RPA 推進の原動力であり、そのように進める中で社員全員の気持ちが前向きになって、もっと仕事を楽にしよう、みんなで一緒に頑張ろうとなり、そして会社の雰囲気がガラッと変わって、ふと気づいたらいつのまにか DX が進んでいた、そのようになるのが理想ではないでしょうか。

ぜひ我々と一緒に笑顔で楽しく RPA・DX を進めていきましょう。



第4章

業務断捨離のすすめ

北崎 恵凡

4.1 今日から私は庶務担当

これから話す内容は架空の会社の技術部門の一組織で、本業であるエンジニアリングの傍ら、庶務担当を命ぜられ、業務断捨離を進めてきた体験談（フィクション）です。どのように考えて行動し、苦楽を経験し、業務改善してきたかの一節を共有できれば幸いです。

4.2 はじまり

少子高齢化、働き手世代の人数減少と団塊世代の引退（2025年の崖問題^{*1}）。じわじわと差し迫ってくる不安がある日、現実のものとなる。担当者が3人→1人に減り（2人は円満引退・シニア再雇用で別の道へ）、技術系管理職の宿命とは言え、すべての庶務業務が自分に回ってきた。組織変更や人事異動も重なり、業務整理をせざるを得ない状況に陥った。何から始める（た）か最初に頭の中に浮かんだのは、業務のデューデリジェンス（本来は投資を行うにあたって、投資対象となる企業や投資先の価値やリスクなどを調査すること）だった。表現として「業務整理」「業務棚卸し」でもよかったが、判断軸を設定し、価値がないものに工数（コスト）をかけてもムダだと思ったからだ。

営利企業の目的は2つしかない。

- 売上（収益）増加（P）

^{*1} 経済産業省が「DX レポート」にて提示した日本の近い将来に対する警鐘で、日本企業が DX の取り組みを十分に行わなかった場合、2025 年以降に年間で最大 12 兆円の経済損失が発生し、国際競争力を失うという課題。合わせて老朽化した IT 設備が残ると、技術的負債によりサービス競争力も低下することが懸念される。

第4章 業務断捨離のすすめ

- コスト削減 (L)

本業はインフラ設備の運用保守業務を担当し、SRE(Site Reliability Engineering)^{*2}を目指して活動していますが、運用保守業務はいわゆる「コストセンター」と呼ばれ、サービスやシステムの信頼性を高める活動や付加価値を創造する活動にもあまりコストを掛けられず、日々昼夜、自動化・効率化に勤しんでいます。庶務業務もエンジニアリングの一種と捉えて、同じように取り組めないか、と考えはじめました。

4.3 何をする(した)か

- 庶務業務一覧の作成
- 当番表の作成+ロギング・数値化
- 報告フォーマット(1枚)の作成
- Before / After シートの作成
- セルフサービス化
- コミュニケーションコストを下げる(情報の公開、チャットボットの活用)
- 機械化・自動化
- 事務局の立ち上げ

4.4 なぜ庶務業務は属人化しやすいか

本業でないことは誰も興味が無いし、担当者に丸投げ・任せきりになり、属人化しやすいです。担当者が突然いなくなると困る潜在的な問題を抱えています。業務知識を公開(属人化→形式知化)し、業務理解に努める仕組みが重要です。

4.4.1 庶務業務一覧の作成

庶務業務一覧を作成しました。

^{*2} Google 社が提唱したサービス・システムの信頼性、安定性、拡張性、効率性を高めるためのエンジニアリングの一種で、システムのエラーを防止するために、自動化、モニタリング、テスト、リカバリなどのプロセスを導入することにより、人為的なミスを減らし、高いシステム信頼性を実現することを目指した方法論。システムの性能改善を目的としたメトリクスやデータ分析に基づく意思決定も重要な要素となっている

4.4 なぜ庶務業務は属人化しやすいか

No.	業務名称	業務内容	責任者	担当者	業務確認書	更新日	工数(h/月)
1	業務A	- 項目A	責任者A	担当者A 担当者a	資料A	2023年4月30日	12
2	業務B	- 項目B - bはbbをXXまでに行う	責任者B	担当者b	-	2022年1月15日	2
3	業務C						
4							
5							
6							

図 4.1: 庶務業務一覧

- 採番
 - 業務に詳しい担当者間であれば「XX の件」で通じるかもしれません、業務内容を知らなければ説明や指示にコミュニケーションコストが発生するので、庶務業務の「No.3」の件、というようにミニマムコストで認識の齟齬が発生しないコミュニケーションが大事です。

- 業務名称
 - 誰にでも(業務を知らない人でも)内容をある程度把握できる分かりやすい名称にします。

- 業務内容
 - 軽微な内容であればここに記載します。量が多く内容が煩雑な場合は業務確認書を作成します。

- 責任者
 - 責任者不在は担当者が困ります。相談相手も報告先もなく、業務が属人化・形骸化しやすい原因を作ります。

- 担当者

- 業務確認書

- 工数(h/月)

- 定量的に数値で判断できるのは重要です。工数=コストです。業務内容に対して適切なコストかどうか(サンクコストも含めて)判断します。

カテゴリ分け、優先順位づけ(重要度と緊急性の2軸マトリクスで行うことが多いですが、困り度も考慮に含めて)行います。

第4章 業務断捨離のすすめ

4.4.2 当番表

リマインダー、チェックリスト、ロギングの機能を持ち、自動的に数値化まで行います。SREではトイル (Toil)^{*3}と呼ばれる種類の業務で、1人あたりのトイルに割かれる時間を50%未満にすることが推奨されています。

2023年5月8日	当番	担当A		
作業種別	業務	頻度	内容	実施記録
アテンド				未
確認・収集				済

図 4.2: 当番表

4.4.3 情報の公開

密室の議論は内容が不透明で属人化を生みやすいので、基本的に情報は公開するようにします。(ただし、セキュリティ上の情報機密区分に応じて内容と公開範囲は適切に設定します) 情報が公開されれば情報の伝達が PUSH 型から PULL 型になることでレイテンシ(待ち時間)が短くなり、コミュニケーションコストが下がります。

^{*3} 時間 (Time)、オペレーション (Operation)、インシデント (Incident)、負荷 (Load) の頭字語。同じことを繰り返し手動で実行する、スケールしない作業を指します。

4.4 なぜ庶務業務は属人化しやすいか

情報の伝達(PUSH型からPULL型へ)

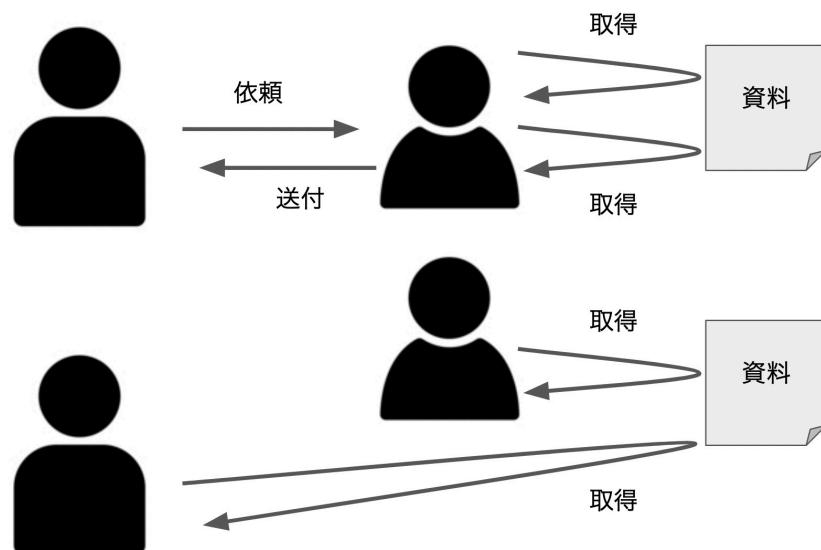


図 4.3: 情報の方向

4.4.4 業務確認書

異なる業務を同じフォーマットで見られるようにして、情報を横断的にアクセスできるようにしておくことが重要です。

第4章 業務断捨離のすすめ

情報のフォーマット化(形式化・体系化)



図 4.4: 業務確認書

4.4.5 報告フォーマット

庶務業務は煩雑で細々とした内容が多く、情報量が多くなりがちです。整理前と整理後を1枚で見えるようにすることが重要です。

2023年4月30日時点	整理後	整理前
変更あり		
(対応必要)	チームA No.AA, BB, CC	No.MM, NN, OO, PP
(対応不要)	チームB No.DD, EE, FF	
変更なし	チームC No.XX, YY, ZZ	

図 4.5: 報告フォーマット

4.5 WHY から始めよ

4.4.6 Before / After シート

これまでの習慣や慣れた行動を変える、変えさせるのは大変です。なかなか理解は得られなくても、シンクコストを下げるために、分かりやすく伝える工夫をします。基本は「人を仲介・介在させない=セルフサービス化」、「コミュニケーションコストは最小限に」です。

Before / After

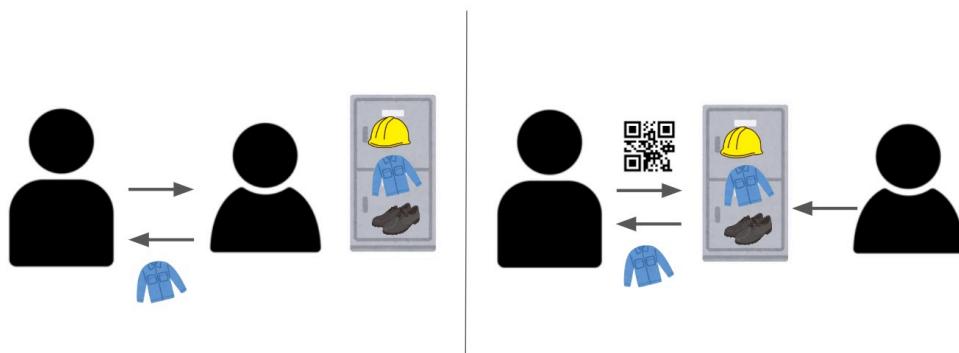


図 4.6: Before / After

4.4.7 事務局の立ち上げ

業務を前へ進めるためには、利害関係が対立した場合に備えて、取り敢えず、落とし所、最後の砦、困った時の駆け込み寺を用意することが重要です。

4.5 WHY から始めよ

ゴールデンサークル理論^{*4}を利用して業務を査定していきました。「何の業務をやっているのか」「なぜ、その業務をやっているのか」「その業務のやり方はベストなのか」業務を担当している方への敬意と感謝を忘れずに、問い合わせ続けることが重要です。

*4 経営者やマーケターによって使用される、より深い目的や価値観を明確にするためのフレームワーク。

なぜ、その業務が必要なのか

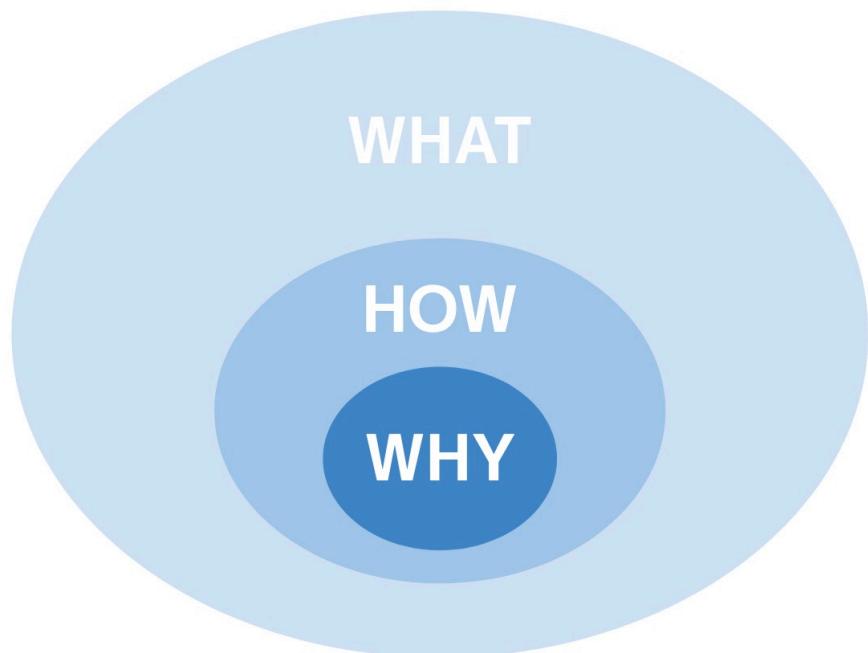


図 4.7: ゴールデンサークル

4.6 断捨離(できたケース)

4.6.1 根本原因が既に解決していた

時間経過と共にルーティン化・マンネリ化・形骸化していた。

4.6 断捨離 (できたケース)

横展開・横串 (はじまり)

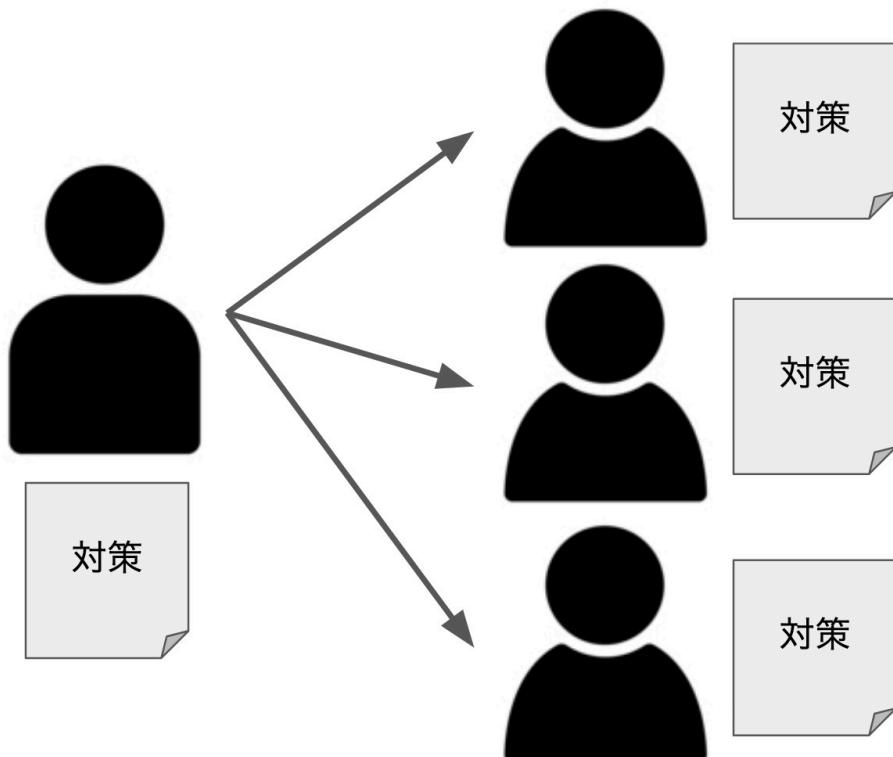


図 4.8: はじまり

横展開・横串(その後)

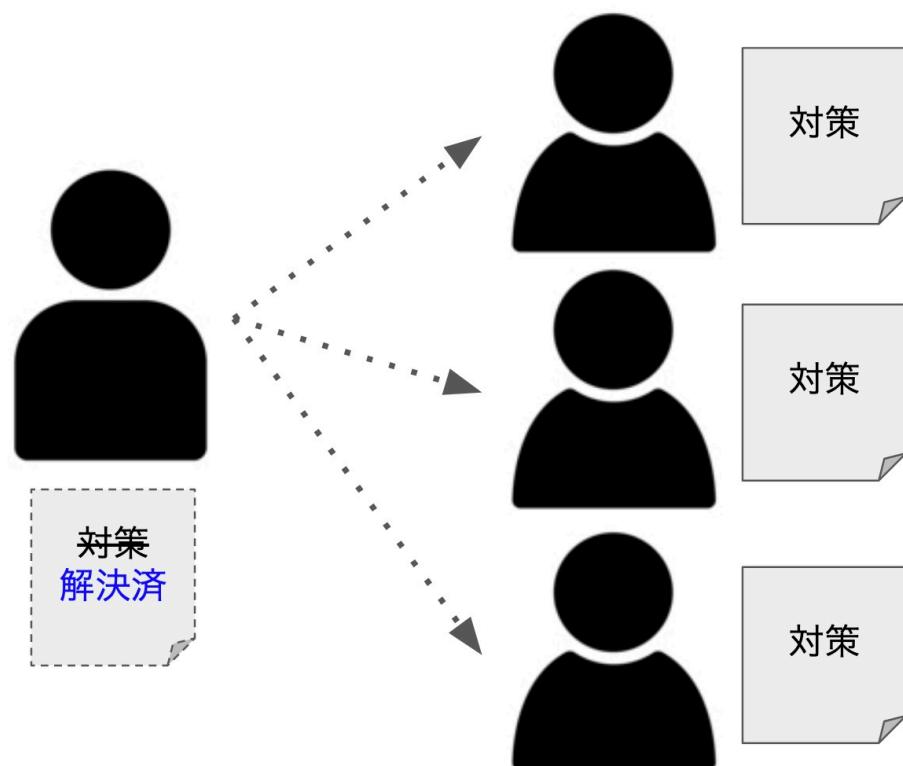


図4.9: その後

4.6 断捨離 (できたケース)

4.6.2 他では簡易に実施、または、実施していなかった

井の中の蛙

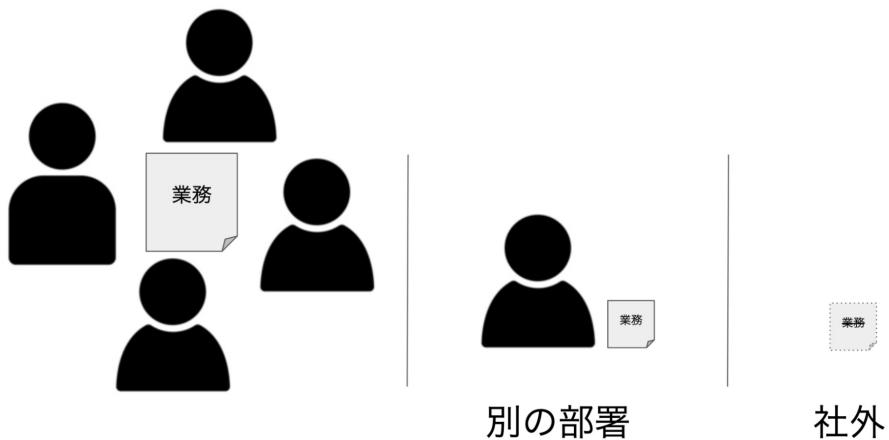


図 4.10: 井の中の蛙

4.6.3 機械化・自動化

巡回業務を無くすことができました。

- ・在庫確認 (スマートマットの活用) . . . 本書後述

第4章 業務断捨離のすすめ

マット1

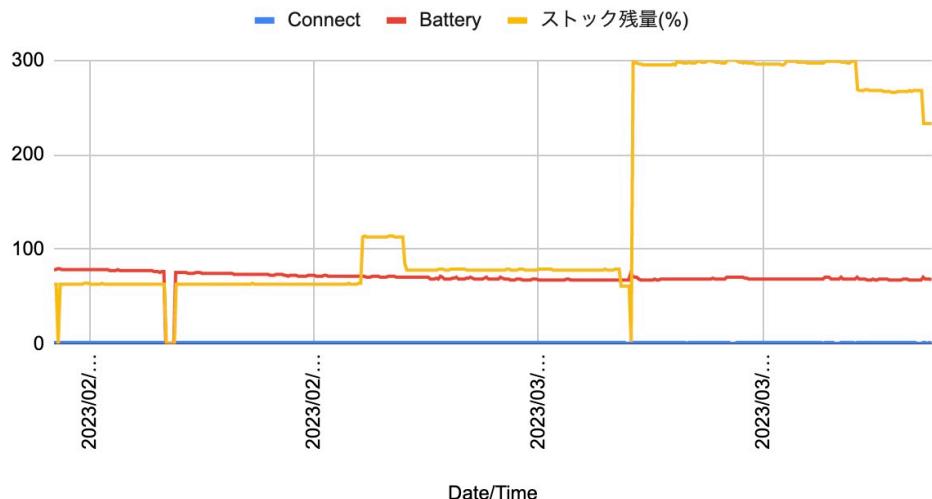


図 4.11: スマートマットライト 1

マット2

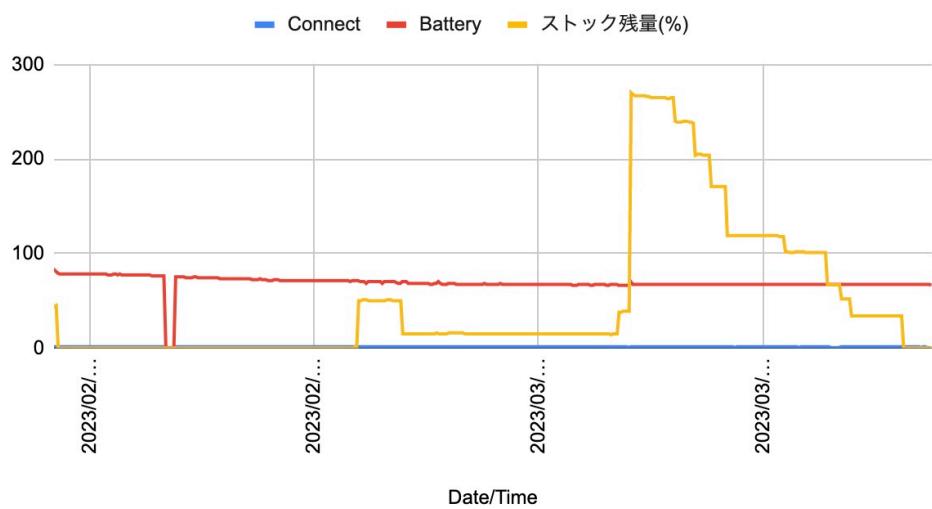


図 4.12: スマートマットライト 2

- ・滞在人数の可視化 (スマート AI カメラの活用)

4.6 断捨離 (できたケース)

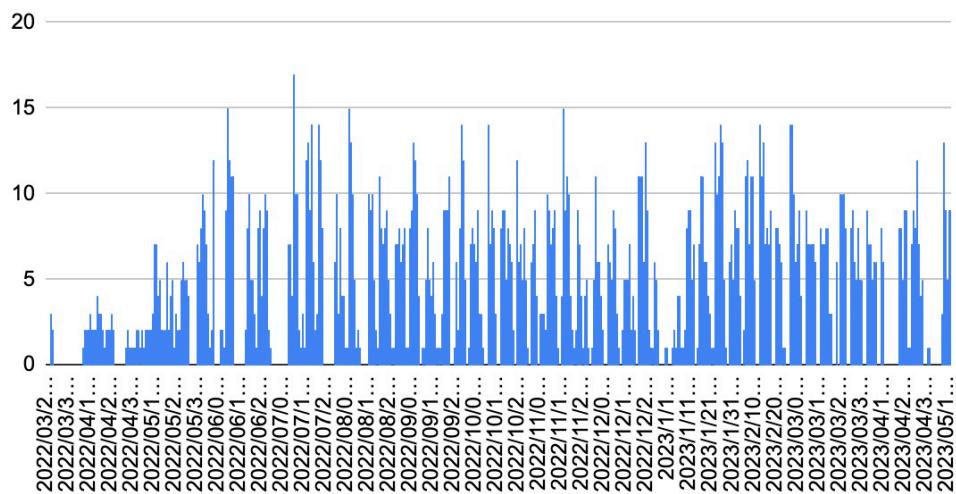


図 4.13: スマート AI カメラ 1

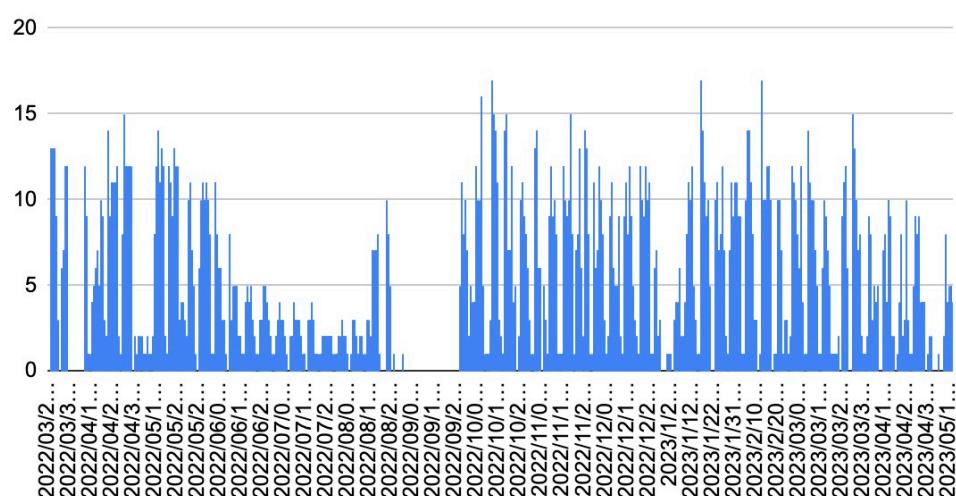


図 4.14: スマート AI カメラ 2

- ・職場環境の快適化 (IoT デバイスの活用)

第4章 業務断捨離のすすめ

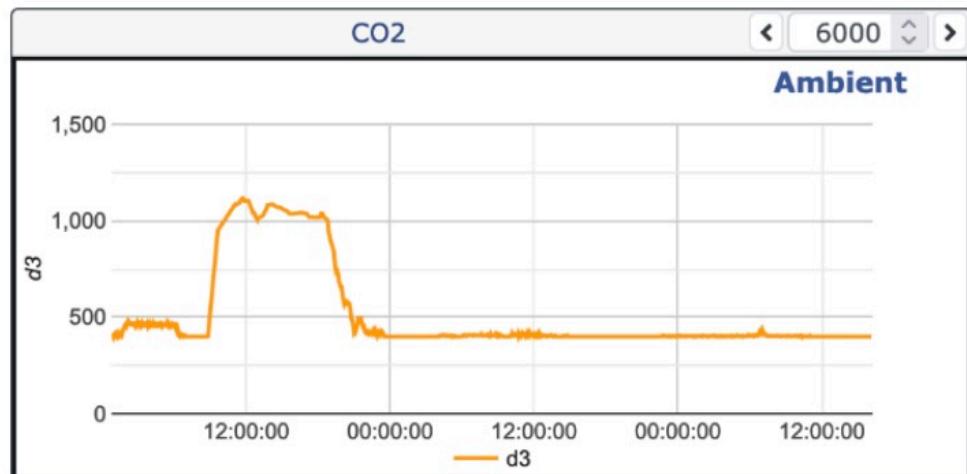


図 4.15: CO2 濃度

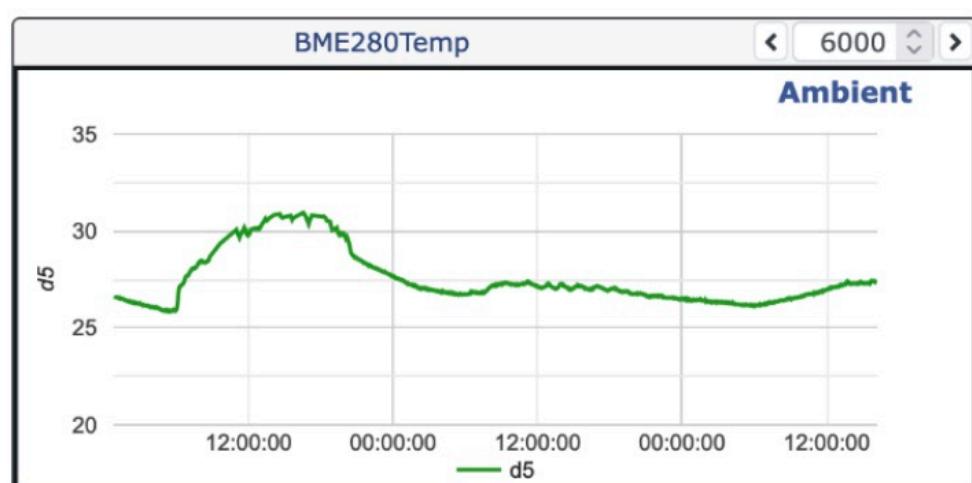


図 4.16: 気温

4.7 断捨離(できなかったケース)

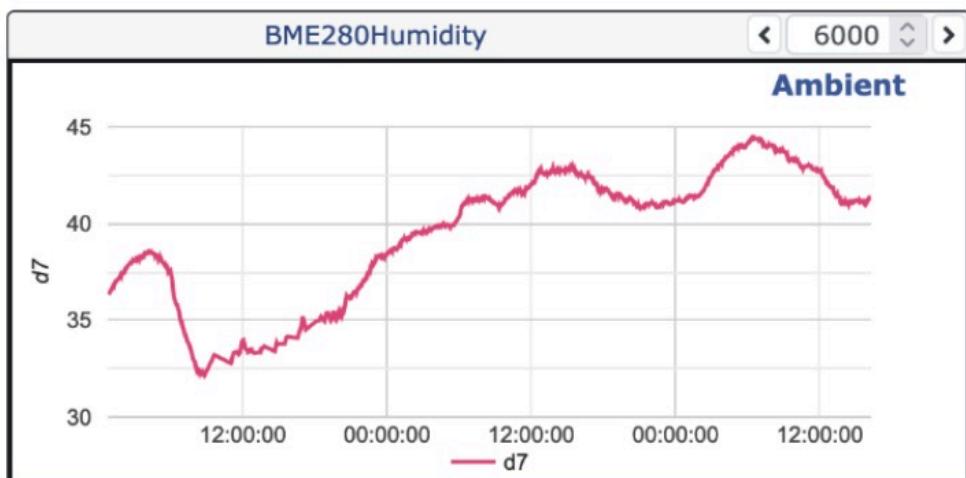


図 4.17: 湿度

4.7 断捨離(できなかったケース)

- 現場業務
 - 物理的に人を動かす必要がある場合には効率化に限界がある。
 - * (郵便物、宅配便、請求書、訪問者対応、など)
 - 頻度を減らす、まとめて対応する、などの対応方法になる。
- 法的規制
 - XX 法、施行規則、関連細則、安全衛生管理、消防法など、ルールを守る必要がある
 - * 業務は勝手に止められない。頻度・回数など。
 - 用語の解釈にも注意が必要。
 - * 「巡視」・「人が見て回ること」。ロボット化できない。
 - 政府によるアナログ規制撤廃^{*5}には期待しているが、対象・候補があまりなかった。
- 事務局の廃止(解散)

^{*5} デジタル原則を踏まえたアナログ規制の見直し (https://www.digital.go.jp/assets/contents/node/basic_page/field_ref_resources/c43e8643-e807-41f3-b929-94fb7054377e/1420dca1/20221221_meeting_administrative_research_outline_08.pdf)

第4章 業務断捨離のすすめ

4.8 悩み(悩んだこと)

- Google Apps Script の引き継ぎ(技術スキルが必要)
 - 背伸びをせず、自分たちの身の丈にあったスキルレベルで実装が求められます。
 - * 高いスキルを求めるに、後任者がメンテナンスできなくなり、技術的負債を生みやすくなります。
- 心構え
 - そもそも、その業務を無くせないか
 - 面倒なこと、苦痛なこと(つらみ)を無くす、緩和できないか
 - 自分の仕事が無くなる不安がある
 - * (別の仕事の割り当てを先に示す必要がある)

4.9 気づき

4.9.1 良かったこと

- 一緒に活動する仲間ができた(いろいろな人と知り合えた)
- チームワークが良くなった(団結力が向上した)
- 相談相手ができた(メンタル面の支えができた)
- 業務の中身を知ることができた
- 問題・課題を具体的に把握できた
- 集合知を生かすことができた(集合知の力を発揮できた)
- 上司/部下それぞれの視点・視野・観点を理解するようになった
 - (客観的・中立的に物事を見られるようになった)
- 事務/技術スキルが身についた
 - スプレッドシート、フォーム、GAS、など
- 技術部門なのでアナログ業務(巡回)よりデジタル(コードベース)の方が
 - 理解しやすい、引き継ぎしやすい
- 技術で解決する欲求が高まった(活動のモチベーションを維持できた)
 - 滞在人数可視化、職場環境のIoT化、計測(スマートマットもその一つ)

4.9.2 反省点

- ジョブディスクリプション(職務定義書)までは踏み込めなかった
- メンバーシップ型→ジョブ型雇用の夜明けが見られなかった

4.10 さいごに

- (差し迫った状況ではなく) 職場メンバーと(ふりかえりとして)座談会をやりたかった

4.9.3 副次的効果

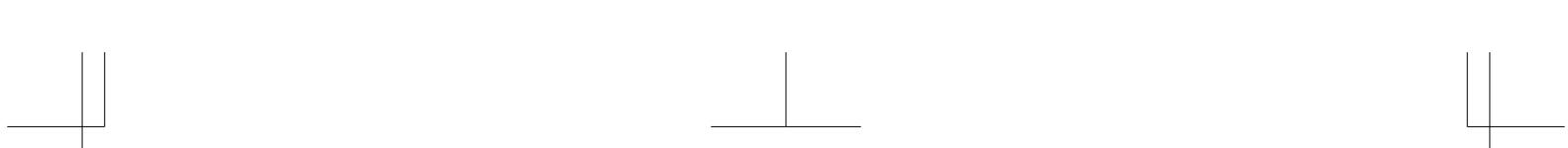
- ES サーベイ^{*6}の向上(責任者・担当者の明確化、ひとり IT 用務員の相談相手ができる)
- 心理的安全性の確保(庶務業務の一覧化、見える化(誰でも見られる場所へ公開)、数値による定量化・客觀化、負担の平準化(得手・不得手、適材適所の場合を除く))
- 担当者のローテーション(サイクル、任期、計画性が生まれた(我慢しよう、先が見えない不安から開放))

4.10 さいごに

二年間に渡る庶務業務整理がひと段落し、成功・失敗いろいろとありましたが、本書の執筆を機会にふりかえることができて良かったです。私が経験したベストプラクティスが、少しでもみなさまの「見える景色を変える」^{*7}お役に立てれば幸いです。

^{*6} ES(Employee Satisfaction) 従業員満足度調査。従業員エクスペリエンス、従業員エンゲージメントと表現されることもある

^{*7} 敬愛する沢渡あまね氏の著書から表現を借用



第5章

対象とする業務が決まつたら

小崎 肇

5.1 はじめに

対象とする業務が決まった段階では、業務の量、掛かった時間、難易度などの情報は得られても、業務の概要は見えていません。もし。この状況のままで作成を開始したら、開発しては疑問が発生し、疑問発生の都度質問をし、回答を待つ、その繰り返しに陥ります。これでは質問する方もされる方も時間が取られてしまいますので、まずは対象とする「業務の概要」をまとめていきましょう。「業務の概要」をあらかじめまとめておくことで、RPAに対してどのような機能を、どのように組み込むか設計することができます。設計せずに、RPAなどを作成する方もいらっしゃると思いますが、得てしてRPAとしてのゴールを見失いがちです。ゴールを決めるためにも、「業務の概要」として、「仕様書」を作成しておきましょう。

5.2 「仕様書」

仕様書は、業務を遂行する人と、RPAなどを開発する人との共通資料となるものです。まずは関係する人みんなが理解できる、統一した言葉を使ってまとめていきます。時として、部署外の人に分からぬ言葉があるかもしれません。その場合は、脚注などに説明を加えていくといいでしょう。略称、略号に関しては、資料の中で統一されてあれば使用しても構いません。

仕様書を書くには、少なくとも、4つの項目(業務情報、入力情報、出力情報、処理情報)が必要です。

1. 業務情報：業務名の他、主となる担当者、いつ、どのように業務を行っていたかの

第5章 対象とする業務が決まつたら

情報をまとめます。

2. 入力情報：業務を行うには、基となる情報（データ）があります。その情報がどのようなものかをまとめます
3. 出力情報：業務を行った結果、新たな情報（データ）が発生します。その発生した情報が、どのようなものなのかをまとめます。
4. 処理情報：入力情報上のデータを、どのように抽出、加工するかをまとめます。細かくまとめる必要性はなく、概要がわかる粒度に留めます。
5. 補助情報：業務を行った時に発生したことのある不具合、原因、その対応方法などを纏めます。

5.3 業務情報

業務名	○○部 ○○業務
業務概要	□□の情報を、●●し、△△を作成する
処理サイクル	□随時 □日次 ■週次 □月次 □四半期 □半期 □その他
処理開始条件	月曜日の10:00～
先行業務（担当部署）	◎◎部からの○○ファイル生成
後続業務（担当部署）	不明（△△部 △△業務担当者）
ロボット名	

入力情報	処理情報	出力情報
<ul style="list-style-type: none">・ ○○ファイル・ マスタファイル	<ul style="list-style-type: none">・ ○○ファイルから条件合致するデータを対象とする・ マスタからメールアドレスを参照する。・ メアド単位のファイルを生成する	<ul style="list-style-type: none">・ △△ファイル・ △△ファイル電子メール

図 5.1: 業務情報

5.3.1 必要な情報

仕様書を作成する際に、必要な情報（1. 業務情報）は以下のようない項目です。

- 業務名
 - 「売上報告」「売掛情報」など、部署で認識されている名称を記入します。
- 業務概要
 - その業務が何をしているのかを、数行で記入します。

5.3 業務情報

- 处理サイクル
 - その業務を毎日行うのか、週なのか、月なのか。それとも任意のタイミングで起動するのかなどを記入します。
- 处理開始条件
 - その業務を開始するキッカケ、条件を記入します。例えば時刻、ファイルの有無、メールの受信などです。
- 先行業務（担当部署）
 - 入力とする情報には、必ずその情報を出力した業務が存在します。開発に際して、直接影響する訳ではありませんが、内容を確認する窓口情報などがあると安心です。
- 後続業務（担当部署）
 - 出力する情報には、普通であればかならず後続業務に入力情報を扱う業務が存在します。開発に際して、直接影響する訳ではありませんが、例えば処理した出力ファイルの変更を打診するなど、連絡する窓口情報などがあると安心です。
- ロボット名
 - 仕様書を書く段階では不要ですが、開発の際のプロセス名、ファイル名などに使用します。開発資産に応じた名称を、記入します。

5.3.2 任意な情報

仕様書を作成する際に、あった方がよいと思われる情報は以下の通りです。

- 業務手順書の有無
 - 現在、業務を行っている人のメモが書き込まれたものではなく、部署として公式に作成されたものです。古い状態のものであっても、あるかないかを明確にしておきましょう。古い状態の場合には、この機に最新化してみましょう。
- 業務での異常発生時の対策
 - 入力とするファイルのデータ品質が悪く、業務遂行できない、有事の際の対策を記入します。例えば、今は実行しなくても大丈夫なのか、先の業務手順書をもとに、手作業で遂行するのかなどです。（※ これはRPAを改修、データを綺麗にするための猶予時間の確認のためであり、この対策があるからRPAの品質を下げてもよいというものではありません。）

第5章 対象とする業務が決まつたら

5.4 入力情報

正式名称（呼称）	○○ファイル
現在参照するメディア	<input checked="" type="checkbox"/> Windows Local <input type="checkbox"/> Windows Server <input type="checkbox"/> Google Drive <input type="checkbox"/> SharePoint <input type="checkbox"/> その他
現在参照するフォルダ 固定部／可変部 可変部命名ルール 生成方法	G:¥マイドライブ¥@個人事業主¥仕訳ツール 固定 <input checked="" type="radio"/> 既存 / 手動 / 自動
現在参照するファイル 固定部／可変部 可変部命名ルール 1処理辺りの数 1ファイル辺りのデータ数 第三者制限	202201_○○名簿.xlsx “202201” 処理翌月の年月 <input checked="" type="radio"/> 1 / n 平均 件 / 最大 件 無 / 条件付き / 提示不可

図 5.2: 入力情報 1

正式名称（呼称）	○○ファイル
ファイル形式 EXCEL	シート名： 固定 / <input checked="" type="radio"/> 最左端 / 不定（条件） 固定シート名「」 開始セル： A 2 列情報：別添
SHEET	
CSV	文字コード： SJIS / UTF-8 改行コード： LF / CRLF / 不定 セパレータ： カンマ / タブ / バイプ 列情報：別添
MAIL	アカウント：
WEB（起点URL）	別添遷移図参照
その他	

図 5.3: 入力情報 2

5.4.1 必要な情報

仕様書を作成する際に、必要な情報（2. 入力情報）は以下のようない項目です。

- 正式名称（呼称）
 - 正式名称の他に、部署で呼称する名称、略称も併記します。
- 現在参照するメディア
 - Windows 上のローカルフォルダ、ローカルネットワーク上の共有サーバ、Google Drive などのプラットフォームがあります。
- 現在参照するフォルダ
 - 入力情報が格納されている位置情報を記入します。Windows でしたらファイル名を除くフルパス、Google Drive ならフォルダ (FOLDERID) です。
 - 1. 固定部/可変部
 - * フォルダの情報の中で、固定部分、可変部分を見極めます。
 - 2. 可変部命名ルール
 - * 例ええば実行日を基準として、前月の年、月がフォルダ名になっている等のルールを記入します。
 - 3. 生成方法
 - * そのフォルダが、どのように作成されるのかを記入します。
 - * 例ええば先行業務の担当者が手で作成している等です。

■コラム： フォルダの命名

とあるお客様の開発現場で、現在使用しているフォルダ情報をサンプルとして共有していただき、そのフォルダ情報の可変部分をサンプルの通りにした RPA を開発しました。受入試験をしていただき、合格をいただいて、本運用も順調に動いていました。しばらくして、その RPA が異常終了したと連絡を受けました。原因調査を進めていくと、本来るべきフォルダの命名は "yyyy △ MM" (△は半角空白) だったのですが、その時のフォルダは、"yyyy △△ MM" だったのです。これは、先行業務の担当の方が手作業で作成した際のミスだった事が原因でした。また先行業務の担当の方と相談して、RPA 側で翌月のファイルを作成してあげ、極力自動化してしまうのも、異常終了リスクを減らす方法です。

- 現在参照するファイル
 - 入力情報のファイル情報を記入します。Windows でしたらファイル名、

第5章 対象とする業務が決まつたら

Google Drive ならファイル(FILEID)です。

- 1. 固定部／可変部

* ファイルの情報の中で、固定部分、可変部分を見極めます。

- 2. 可変部命名ルール

* 例えばファイル名の一部に社員番号が含まれる等のルールを記入します。

- 3. 1処理辺りの数

* 入力情報のフォルダに、そのファイルが何個格納されるかを記入します。

* フォルダに1つしかない場合、フルパスで狙い撃ちで読み込む処理を行えます。

* フォルダにファイルが複数ある場合、全てのファイルを対象にするのか、いずれかの一つを対象にするのか、確認が必要です。

- 4. 1ファイル辺りのデータ量

* 対象とする入力情報のデータ量を記入します。

* 例えば、1ファイル辺りのデータ量が数千だとしても、処理対象が複数の場合はそのファイル数倍のデータ量になります。

* 1ファイル辺りのデータ量が巨大であつたら、その入力情報のデータの持たせ方を変えるなど、検討が必要な場合も出てきます。

- 5. 第三者制限

* 部署で内製するのであれば、入力情報を第三者に開示する必要はないですが、外製とする場合は、個人情報などの扱いに気をつけなくてはならない情報が含まれている入力情報を開示しなければならない場合があります。そういった場合は、「条件付き」として、以下のような処置を施す事を検討する必要が出てきます。

・ 情報にマスクを掛ける

・ 情報量を減らし、正しいデータの桁数などに合わせたダミーデータを準備する

・ レイアウトはそのままで、処理で使う情報以外を空白にしてしまう

* 選択肢として「提示不可」も設定しましたが、このデータですと内製化も難しくなると思われますので、「条件付き」として扱える入力情報にする事を考えましょう。

- 6. ファイル形式

* ファイル形式によって、読み込むべき方法が変わります。

* 例えばEXCELには複数のシートが内包できるので、どのシートを対象とするのか確認する事が必要です。

* CSVですと、文字コード(SJIS/UTF-8)、区切り文字(カンマ、タブなど)、改行コード(Lf/CrLf)を確認する事が必要です。

5.5 出力情報

■コラム: 0で始まる数字のデータ

とあるお客様の開発現場で、現在使用しているファイルをサンプルとして共有していただき、EXCELファイルを入力にしたRPAを開発した事があります。EXCELファイル中に、従業員番号という項目があり、その桁がまちまちなのです。業務のご担当様に確認したところ、CSVファイルをEXCELで開いて、列を入れ替えたりしている事が判明。0で始まる数字のデータを含んだCSVファイルをEXCELで開くと、先頭の0はなくなる事は認識していたようですが、これまで手で対応されていたそうです。もちろん、CSVファイルを入力にしたRPAに改良し、人手を介することはなくなりました。

5.5 出力情報

正式名称（呼称）	△△ファイル
現在出力しているメディア	<input checked="" type="checkbox"/> Windows Local <input type="checkbox"/> Windows Server <input type="checkbox"/> Google Drive <input type="checkbox"/> SharePoint <input type="checkbox"/> その他
現在出力しているフォルダ	G:¥マイドライブ¥@個人事業主¥仕訳ツール¥202202¥
固定部／可変部	“202202”
可変部命名ルール	処理年月
生成方法	既存 / 手動 / <input checked="" type="radio"/> 自動
現在出力しているファイル	△△_202202.csv
固定部／可変部	“202202”
可変部命名ルール	処理年月
新規／更新	<input checked="" type="radio"/> 新規 / 更新
1処理辺りの数	<input checked="" type="radio"/> 1 / n
第三者制限	<input checked="" type="radio"/> 無 / 条件付き / 提示不可

図 5.4: 出力情報 1

第5章 対象とする業務が決まつたら

正式名称（呼称）	△△ファイル
ファイル形式 EXCEL	シート名： 固定 ／ 最左端 ／ 不定（条件） 固定シート名「」
SHEET	開始セル： A2 列情報：別添
CSV	文字コード： SJIS ／ UTF-8 改行コード： LF ／ CRLF ／ 不定 セパレータ： カンマ ／ タブ ／ パイプ 列情報：別添
MAIL	アカウント：
WEB（起点URL）	別添遷移図参照
その他	

図 5.5: 出力情報 2

5.5.1 必要な情報

仕様書を作成する際に、必要な情報（3. 出力情報）は以下のようない項目です。

- 正式名称（呼称）
 - 正式名称の他に、部署での呼称、略称も併記します。後続処理を扱う部署での呼称、略称もあるといいでしよう。
- 現在参照するメディア
 - Windows 上のローカルフォルダ、共有サーバ、Google などのプラットフォームがあります。
- 現在参照するフォルダ
 - 出力情報として保存する位置情報を記入します。Windows でしたらファイル名を除くフルパス、Google ならフォルダ (FOLDERID) です。
 - 1. 固定部／可変部
 - * フォルダの情報の中で、固定部分、可変部分を見極めます。
 - 2. 可変部命名ルール
 - * 例えば実行日を基準として、前月の年、月がフォルダ名になっている等のルールを記入します。
 - 3. 生成方法
 - * そのフォルダが、どのように作成されるのかを記入します。

5.5 出力情報

- * 多くは、出力情報を格納する際に作成する事が多いとは思います。部署間の場合には、後続の業務の担当者が、ご自身の手で作成する場合もあるかもしれません。
- 現在出力しているファイル
 - 出力情報のファイル情報を記入します。Windows でしたらファイル名、Google ならファイル(FILEID)です。
 - 1. 固定部／可変部
 - * 出力ファイルの情報の中で、固定部分、可変部分を見極めます。
 - 2. 可変部命名ルール
 - * 例えば出力ファイル名の一部を社員番号とする等のルールを記入します。
 - 3. 新規／更新
 - * 出力情報のフォルダに、出力ファイルを新規作成するのか、あるいは既存のファイルに追記するのかを記入します。
 - * 新規作成の場合は、出力ファイルに対して上書きする方法、あるいは一度出力ファイルを削除してから改めて新規に作成する方法があります。
 - * 追記の場合は、出力ファイルを事前に読み込む必要が発生します。
 - 4. 1処理辺りの数
 - * 出力情報のフォルダに、そのファイルが何個格納するかを記入します。
 - * 特に処理には影響しませんが、後続処理には大いに関係しますので、しっかりまとめ、確認しておきましょう。
 - 5. ファイル形式
 - * ファイル形式によって、書き込むべき方法が変わります。
 - * 例えば EXCEL には複数のシートが内包できるので、シート名を固定にするのか、可変でいいのか、
 - * 新規作成の場合、必ず存在する"Sheet1"にするかなど、後続処理の担当者と確認する事が必要です。
 - * CSV ですと、文字コード (SJIS/UTF-8)、区切り文字 (カンマ、タブなど) を確認する事が必要です。
 - 6. 第三者制限
 - * 出力情報には、第三者制限を受けた入力情報が転記されている場合があります。他にも、業務ロジックで新たに生成された第三者に開示できない情報が含まれている場合もあるかもしれません。
 - * 例えばメールに添付する場合の誤送信対策を確実に行う必要性などの確認項目になります。

5.6 処理情報

仕様書を作成する際に、必要な情報（4. 処理情報）は以下のようにまとめます。

- 入力情報の準備をする、使用する、出力ファイルを書き出す（追記する）など、情報の状態の変化を明示する。
 - 例)
 - * ○ 所定フォルダから、Tファイルを開く。
 - * × エクスプローラで所定フォルダを開き、中のTフォルダをダブルクリックして開く。
- 何の目的にしているのかを書き出す。その際に、手段・方法（どのように）は記述しない。
 - 例)
 - * ○ TファイルのX項目から、MファイルのY項目を得る。
 - * × TファイルのX項目から、VLOOKUP関数を使用して、MファイルのY項目を得る。
 - * ○ SシートのX項目から、値が100以上の情報を得る。
 - * × Sシートにオートフィルタを設定し、X項目から、値が100以上の情報を得る。
- 業務上の条件分岐を記述する。その際、「処理対象とする条件」と対を成す「処理対象外とする条件」を併記すると理解しやすい。
 - 例)
 - * ○ メールアドレスが設定されている情報から、電子メールを作成する。
 - * ○ メールアドレスが設定されていない情報は、その情報件数を計測し、処理の最後に、その情報件数を載せた電子メールを作成する。
- 業務上の繰り返し作業の範囲を明示する
 - 例)
 - * ○ 明細ファイルがなくなるまで、以下の処理を繰返す。明細ファイルのデータをサマリファイルに転記する。
 - * × 明細ファイルのデータをサマリファイルに転記する。(前述の入力情報で「1 処理ファイルの数」で分かりますが...)

5.7 補助情報

仕様書を作成する際に、必要な情報（5. 補助情報）は以下の通りです。

5.7 拡助情報

- 想定される不具合、原因と対策

- 先行する業務の出力情報が、本業務の入力情報になります。その時の出力情報の品質が悪いと、本業務での処理の妨げになります。ここでは過去に遭遇した不具合と原因、対策を記入します。

- 例)

- * 不具合：ゼロで始まる数字の箇が、先行するゼロがなくなっていて、それ故マスター上の値と合致せず、対象外となる。
 - * 原因：先行業務では、CSVファイルを扱うのであったが、値の修正するシーンがあり、その際の修正にEXCELを使用していたため。
 - * 対策：正しい情報を入手する。あるいは、所定の桁になるまでゼロを前置する。

- 例)

- * 不具合：A列から有効データが格納されているはずが、先行業務のメモ用に列が挿入され、有効データがB列からになっている。
 - * 原因：先行業務の担当が後で調査するための情報を残したまま、本処理が実行されてしまった。
 - * 対策：有効データの開始列を判断する文言を探し、その文字がある列から処理するようにする。

- 「設定ファイル」に関する設定方法、参照方法

- 例えば、メールの件名のような固定的な情報は、RPAプログラム自身に持たせるのではなく、EXCELのようなファイルに持たせ、そのファイルから情報を得る方法がいいでしょう。そうしておけば、開発が完了し運用フェーズに入ってから、もし業務の仕様が変わって、メールの件名を変更する場合にも、RPAプログラムを修正するのではなく、そのファイルを修正するだけで動きを変える事ができます。そのような役割のファイルを「設定ファイル」と言います。「コンフィグファイル」とか、「CONFINGファイル」と呼ばれる事もあります。「設定ファイル」を準備したら、その管理方法を決めます。

- 1. RPAプログラム内部の一部として管理する

- * 同じロボットを複数動かす時に、ロボットそれぞれに違う情報を与える事ができる。
 - * 逆も言え、同じロボットを複数動かす時に、同じ情報かどうかは保証できない。

- 2. RPAプログラム内部とは、別区画に管理する。別区画の情報は、RPAプログラム内部の一部として管理するファイルにて管理する。

- * 同じロボットを複数動かす時に、ロボットそれぞれに同じ情報を与える事ができる。

第5章 対象とする業務が決まつたら

* 別区画が使えなくなると、そこを参照する全てのロボットが実行できない。

5.8 結び

「業務の概要」をまとめるために必要な情報などを列挙してきました。無論、これが足りないとか、これは要らないとか、色々なご意見があると思いますが、その都度決めていただければと思います。

本書で紹介した様式は、システム設計における図式化手法のひとつである、HIPO (Hierarchy plus Input Process Output) の様式を模したものです。

HIPO は、「図式目次」と「IPO ダイアグラム」の 2 つから構成されます。機能の階層 (Hierarchy) を「図式目次」に記述し、図式目次の機能ごとに入力 (Input)、処理 (Process)、出力 (Output) の関係を「IPO ダイアグラム」に記述するのですが、機能の階層より、入力、処理、出力の情報をまとめるための「IPO ダイアグラム」を参考にしました。

この表はパワーポイントの「表」を挿入して作成しましたが、実際には、EXCEL ファイルなどにまとめた方が便利です。

- ファイル中の文字列を検索しやすい
- ハイパーリンクを設定して、展開されている情報へジャンプさせる
- 設定した印刷範囲外に、メモ情報を残せる
- シートを保護し、情報を保護する

「業務の概要」として作成した「仕様書」は、レビューなどを経て、正式な書類として残すようにしていきましょう。無論、業務の内容が変わったら改修も必要ですし、業務が不要になったら文書の抹消も必要です。

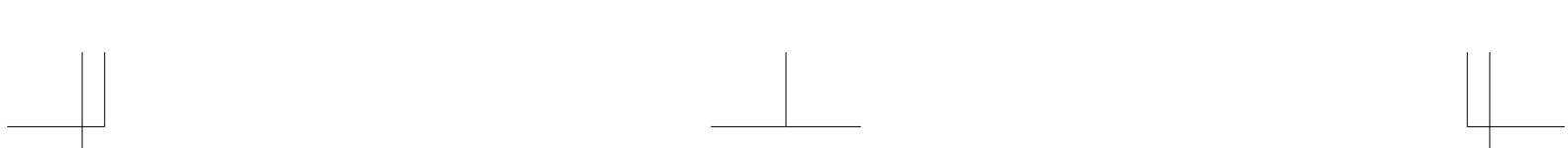
■コラム：無意味な作業

とあるお客様の開発現場で、Aロボットで使用しているプラットフォームを別のプラットフォームへと変更する改修を依頼されました。改修対象となるAロボットを共有され、中身を解説し始め、改修概要をまとめた資料を作成し、現場担当者様との打ち合わせに臨んだのですが、開口一番、「Aロボットは現在運用していない」と。その場でBロボットの改修に当たる事になったのですが、共有されたBロボットは、プラットフォーム変更以前に業務ルール改編による改修が進行中で、2週間ほど無意味な作業をしていたのでした。

最後に、入力とする情報は「ナマモノ」です。常に変化しています。そのような情報を

5.8 結び

扱う業務も「ナマモノ」と言えるでしょう。それを詳らかにしても、過去のものになっていきます。新しいパソコンを買った途端、古いパソコンと言われてしまうように。しかし、過去になるからといって、「業務の概要」をまとめない手はありません。改めて業務に向き合って、「業務の概要」にまとめてみてください。まとめてみる事で、本当に必要な業務なのか？他で同様な事をしている業務ではないか？が知る事ができる「キッカケ」になります。これも「自動化」への一歩になります。



第6章

スマートマットをハックしてさらに便利にする

北崎 恵凡

6.1 スマートマットとは

あらかじめ登録した商品の重さを定期的に計測し、指定した条件になると自動で発注するスマートデバイスです。製品としてスマートマットライト^{*1}とスマートマットクラウド^{*2}の2種類があります。スマートマットライトはAmazonの日用品(水やペーパータオルなど、約3,000の対象商品)の自動注文に特化しており、管理画面が用意されています。マットの価格^{*3}は安価(セール時～通常:1,980～2,980円)で、マット10枚まで利用可能です。スマートマットクラウドは法人向けサービスで、棚卸自動化、入出庫管理、自動注文など、フルスペック・フルサービスの在庫管理・発注プラットフォームです。違いの詳細は製品のヘルプセンター^{*4}に掲載されています。

*1 <https://service.lite.smartmat.io/>

*2 <https://www.smartmat.io/>

*3 <https://www.amazon.co.jp/dp/B08G8B2928>

*4 <https://smartshopping.my.site.com/help/s/article/000001605>

第6章 スマートマットをハックしてさらに便利にする



図 6.1: 戸棚のスマートマットライト

6.2 スマートマットライト



図 6.2: ペーパータオルの在庫

6.2 スマートマットライト

スマートマットライトはマット本体を購入すれば月額利用料は発生しません（買い切り）。マットは単三電池 4 本で動作し、Wi-Fi へ接続します。定期的に（デフォルトでは 1 日 4 回）重さを測定し、AWS へデータが送信されます。（通信をキャプチャーして確認）データは専用の管理画面^{*5}から、対象のマットを選択→詳細情報→消費レポートへ移動して確認することができます。

^{*5} <https://lite.smartmat.io/>

第6章 スマートマットをハックしてさらに便利にする

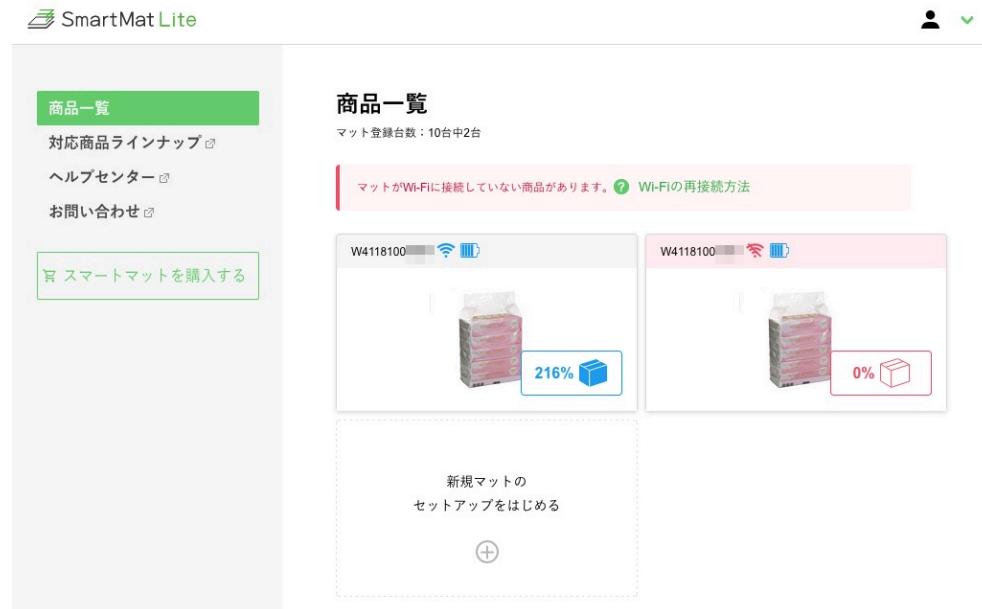


図 6.3: スマートマットライトの管理画面

6.2 スマートマットライト

詳細情報

商品情報



Ar ハンドタオル 200枚入 5個パック >

100%時の商品重量 2.07kg >

注文設定

注文方法 買いどき通知

注文タイミング 20%以下 >

重複注文防止機能 オフ

?

重複注文防止機能とは？

残量情報

先週は 67.1%
昨日は 33.3% 商品を消費しました

商品残量 216% >

消費ベース/日 8.1% >

最終計測日時 2023/03/29 12:49

計測頻度 1日8回 >

?

消費レポートを詳しく見る >

図 6.4: 詳細情報

消費レポート

先週は**67.1%**、昨日は**33.3%**商品を使いました

注文回数

2回

消費ベース/日

8.1 %

使い切るまでの
平均日数

12.3 日

現在の商品残量

?

計測誤差がある場合

216 %

残量グラフ

1週間

1ヶ月

1年

%

300

270

240

210

180

150

120

90

60

30

0

注文タイミング (20%)

3

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

図 6.5: 消費レポート

6.3 スマートマットライト 1 と 2 の違い

6.3 スマートマットライト 1 と 2 の違い

結論から書くと、機能上の違いはほとんどありません。スマートマットライトの裏側のシール(図 4 の矢印)を剥がすとネジが 1 本現れ、プラスドライバーで外すことができます。側面はガラス板とプラスチックの境界(図 5 の矢印)にマイナスドライバーを差し込んで開けることができます。スマートマットライト 1 は ESP-WROOM-02(ESP8266 チップ) 基板の Wi-Fi アンテナを使用していますが、スマートマットライト 2 は Wi-Fi アンテナ(図 6 の矢印)が外しとなり、ネットワークの接続性が向上しています。



図 6.6: スマートマットライトの裏側

第6章 スマートマットをハックしてさらに便利にする

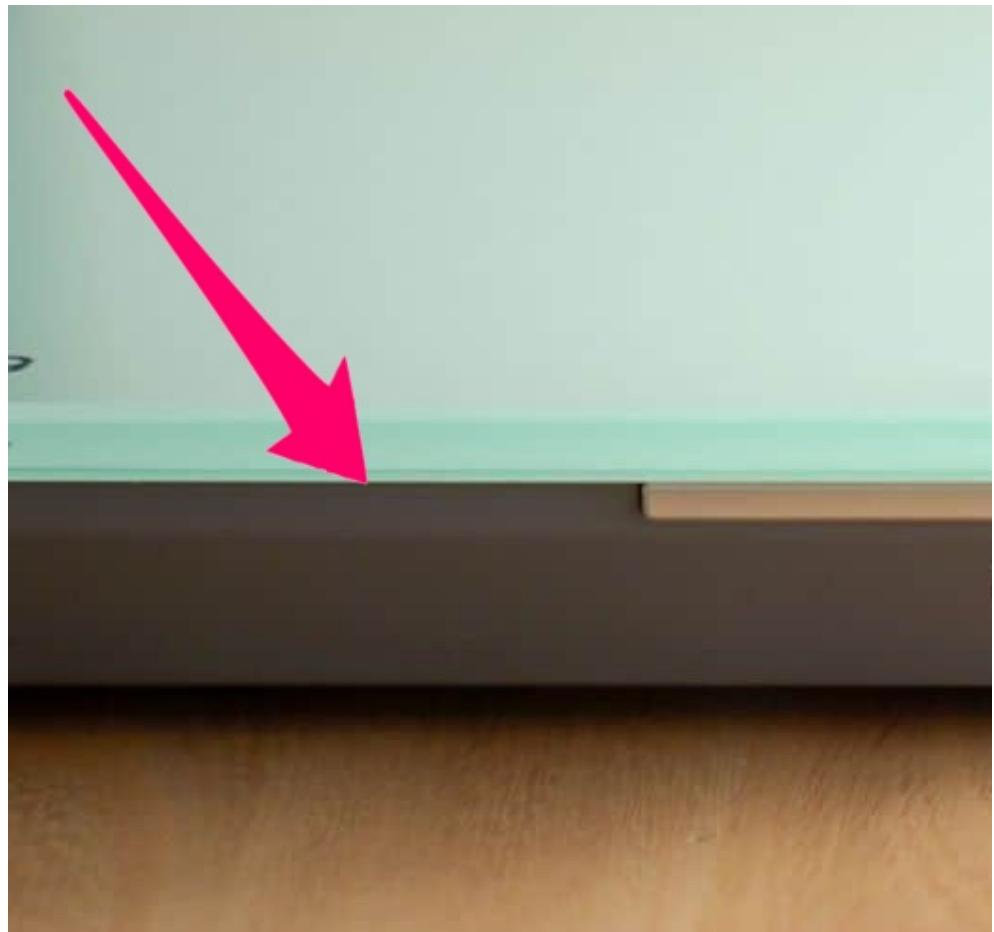


図 6.7: スマートマットライトの側面

6.3 スマートマットライト 1 と 2 の違い



図 6.8: 分解した中身 (バージョン 1)

第6章 スマートマットをハックしてさらに便利にする

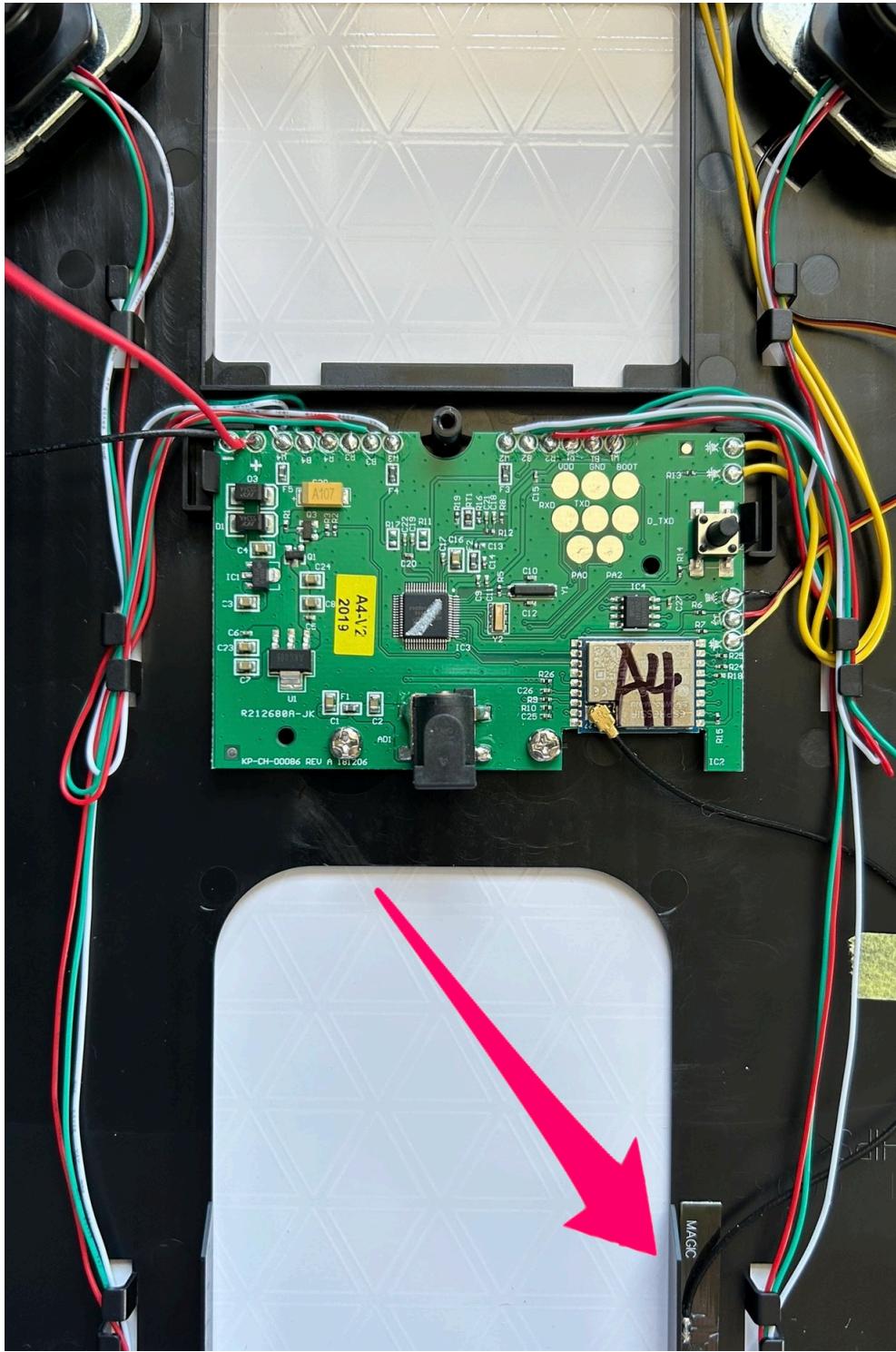


図 6.9: 分解した中身 (バージョン 2)

6.4 スマートマットライトをハックする

6.4 スマートマットライトをハックする

スマートマットライトをそのまま使っても、Amazon アカウントと連携して自動注文してくれたり、E メールや LINE(ID 連携が必要) で通知されるので非常に便利ですが、注文先を変更したり、消費データだけを利用(分析)したり、他の機能と連携させたい場合があります。これらのニーズを実現するためスマートマットクラウドは API 連携^{*6}が可能ですが、スマートマットライトは API が提供されていません。そこでこの記事では、Google Spreadsheet と GAS(Google Apps Script) を使用して管理画面の消費レポートからデータを取得し、他の機能と連携させる方法を説明します。

6.5 パケットキャプチャで通信の内容を確認

まず、macOS のインターネット共有の機能を利用して、スマートマットライトの通信内容を Wireshark^{*7}でパケットキャプチャします。



図 6.10: 接続構成

すると、以下の処理をしていることが確認できます。

1. DNS でネームを検索
2. HTTP POST で測定データを送信
3. HTTP GET でデータを取得

^{*6} <https://smartshopping.my.site.com/help/s/article/000001143>

^{*7} <https://www.wireshark.org/>

第6章 スマートマットをハックしてさらに便利にする

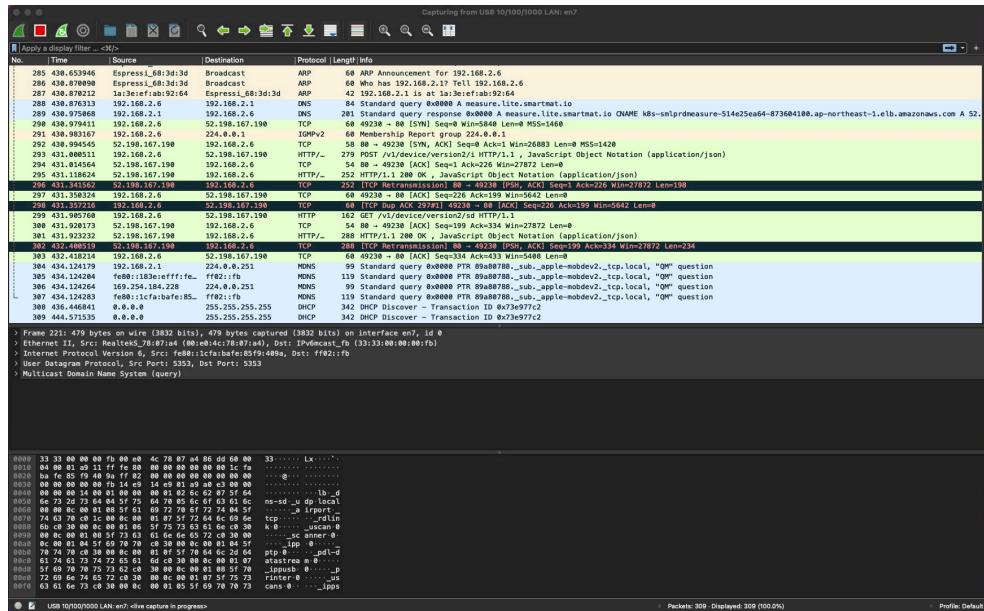


図 6.11: パケットキャプチャ

6.6 ブラウザの開発者ツールで通信の内容を確認

ブラウザの開発者ツールを使用して、スマートマットライトの管理画面にログインします。すると、以下の処理をしていることが確認できます。

1. サインインの URL(/api/bff/v1/signin) で ID(email) とパスワード (password) を送信
2. クッキー (cookie) でセッション ID(sid) を受信

6.7 API の仕様を推察

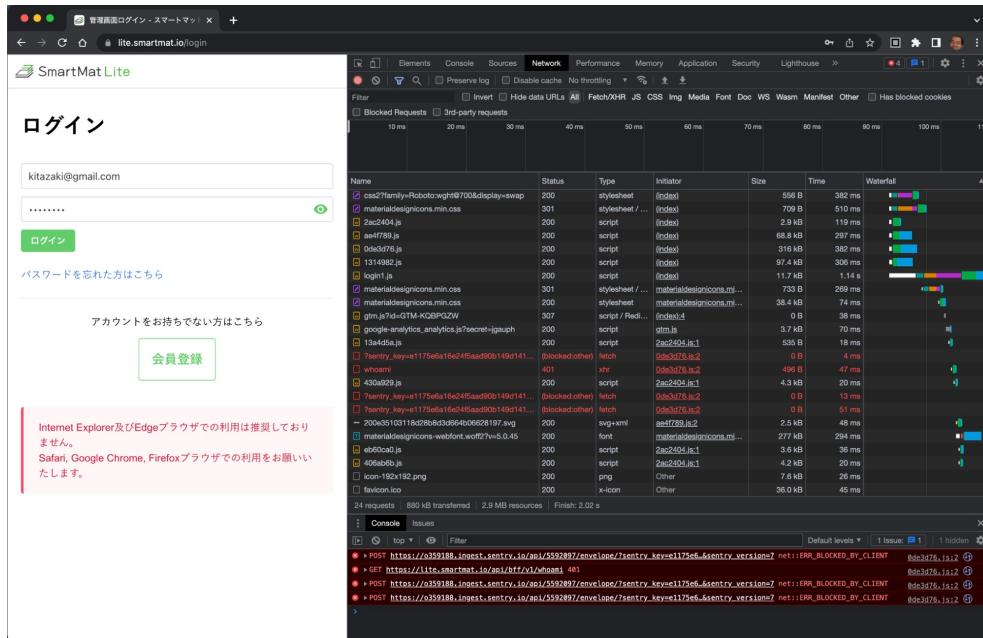


図 6.12: ブラウザの開発者ツール

6.7 API の仕様を推察

管理画面にログインした後、スマートマットのデータを確認します。詳細情報を確認すると、スマートマット毎に URL に管理番号 (subscriptionId) が付与されています。

第6章 スマートマットをハックしてさらに便利にする

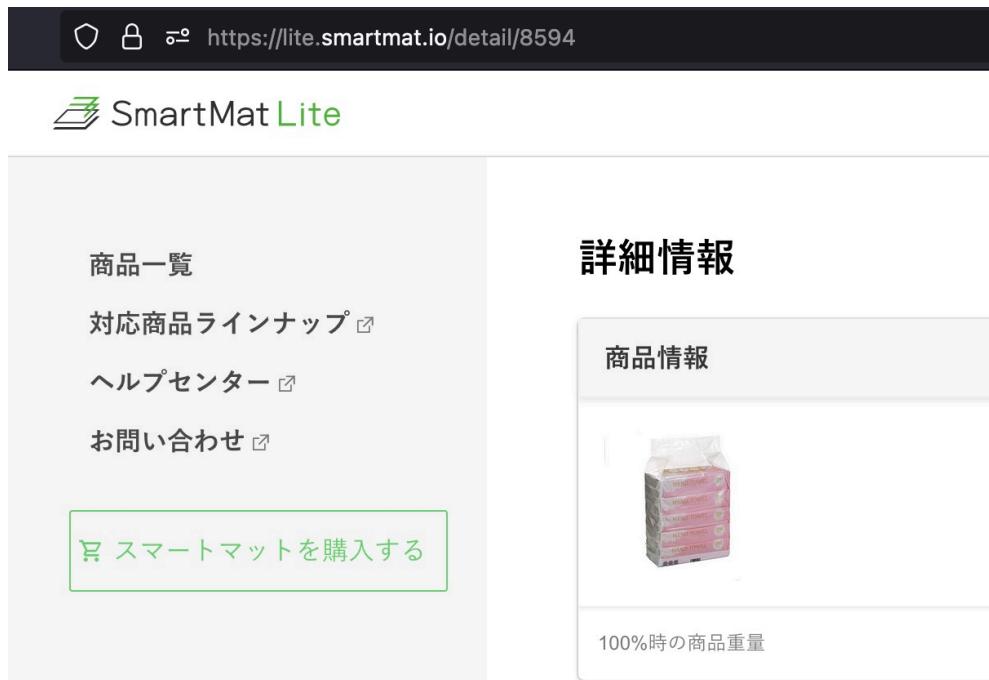


図 6.13: スマートマットの詳細情報

HTTP GET でデータ取得用の URL(/api/bff/v1/subscription/search_detail) にアクセスし、以下の JSON データを取得しています。

```
{  
    "deviceSerialNumber": "WXXXXXXXXXXXXX",  
    "isFirstConnected": false,  
    "isConnected": true,  
    "battery": 66,  
    "remainingPercent": 0,  
    "triggerRemainingPercent": 20,  
    "measuredAt": "2023-04-08T06:06:03+09:00",  
    "current": 0,  
    "frequency": 8,  
    "subscriptionId": 8594,  
    "title": "Ar ハンドタオル 200枚入 5個パック",  
    "full": 2070.21,  
    "productUrl": "https://www.amazon.co.jp/dp/B00KOZ0ZFU",  
    "imageUrl": "https://m.media-amazon.com/images/I/414WdyYdVuL._SL500_.jpg",  
    "outputType": 4,  
    "orderable": false,
```

6.8 GAS の実装

```
"dailyAverageConsumption": 6.5,  
"dailyConsumption": 0,  
"weeklyConsumption": 0,  
"totalOrderCount": 4  
}
```

これで、スマートマットクラウドの API から取得できるデータとフォーマットが一部共通であることを確認できました。

表 6.1: データフォーマット

戻り値	説明	具体例
deviceSerialNumber	シリアル番号	W42190800XXX
isFirstConnected	(不明)	
isConnected	Wi-Fi 接続状況	true
battery	電池残量	73
remainingPercent	最新計測時刻の残量 % 表示の場合のみ	57
triggerRemainingPercent	閾値 (%表示の場合)	20
measuredAt	最新計測時刻	
current	最新計測値 (単位はグラム)	11330
frequency	計測頻度 (1 日 n 回)	1
subscriptionId	(不明)	
title	商品名	お米
full	満タン重量 % 表示の場合のみ	20000
productUrl	(不明)	
imageUrl	商品画像 URL	
outputType	発注通知方法	メール通知
orderable	(不明)	
dailyAverageConsumption	(不明)	
dailyConsumption	(不明)	
weeklyConsumption	(不明)	
totalOrderCount	(不明)	

6.8 GAS の実装

通信の確認と API の解析結果をもとに、GAS(Google Apps Script) を作成します。作成するスクリプトは大きく 2 つの処理から成ります。

1. スマートマットライトの管理画面へログインし、スマートマットの測定データを取

第6章 スマートマットをハックしてさらに便利にする

得する

2. 取得したデータをスプレッドシートに記録する

GAS の中の変数に必要な情報を設定します。

- ・「email」 → 管理画面のログイン ID
- ・「password」 → 管理画面のパスワード
- ・「XXXX」 → スマートマットの管理番号
- ・「sheet_id」 → スpreadSheet ID
- ・「sheet_name」 → スpreadSheet のシート名

GitHub にサンプルコード^{*8}を載せましたので参考にしてみてください。

```
function fetchJSONData() {  
  
    const postdata = {  
        'email': '', // ID  
        'password': '' // Password  
    }  
  
    const loginUrl = 'https://lite.smartmat.io/api/bff/v1/signin';  
    // ログインURL  
    const dataUrl = 'https://lite.smartmat.io/api/bff/v1/subscription/search_detail?subscriptionId=XXXX'; // データURL  
    const sheet_id = ''; // SpreadSheetID  
    const sheet_name = '' // SpreadSheetName  
  
    const options = {  
        'method': 'post',  
        'Content-Type': 'application/json',  
        'payload': JSON.stringify(postdata),  
        'followRedirects': false  
    }  
  
    // ログインページにPOSTリクエストを送信して、Cookieを取得する  
    const response = UrlFetchApp.fetch(loginUrl, options);  
    const headers = response.getHeaders();  
    const cookie = headers['Set-Cookie'];  
  
    // Cookieを使用して、JSONデータを取得する  
    const jsonData = UrlFetchApp.fetch(dataUrl, {  
        'headers': {  
            'Cookie': cookie  
        }  
    })
```

^{*8} <https://github.com/kitazaki/smartmat/blob/main/smartmat.gs>

6.8 GAS の実装

```
}).getContentText();

// JSONデータを解析する
const parsedData = JSON.parse(jsonData);

// スマートマットがWi-Fiに接続されている場合: true → 1、切断されて→
//いる場合: false → 0
let connect = 0;
if (parsedData['isConnected']) {
    connect = 1;
}

// JSONデータをSpreadSheetに出力する
const MySheet = SpreadsheetApp.openById(sheet_id);
MySheet.getSheetByName(sheet_name).appendRow([
    [new Date(), connect, parsedData['battery'], parsedData[→
    'remainingPercent']]
]);
}
```

GAS を保存したら「実行」を押します。

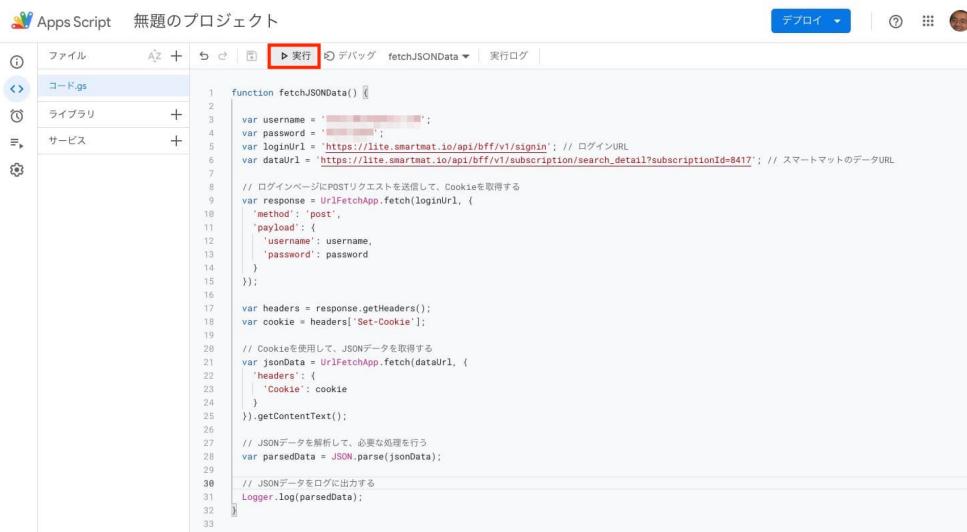


図 6.14: GAS の実行

初めて実行する場合、承認が必要ですので、「権限を確認」を押します。

第6章 スマートマットをハックしてさらに便利にする



図 6.15: 権限を確認

次に、アカウントを選択します。

6.8 GAS の実装

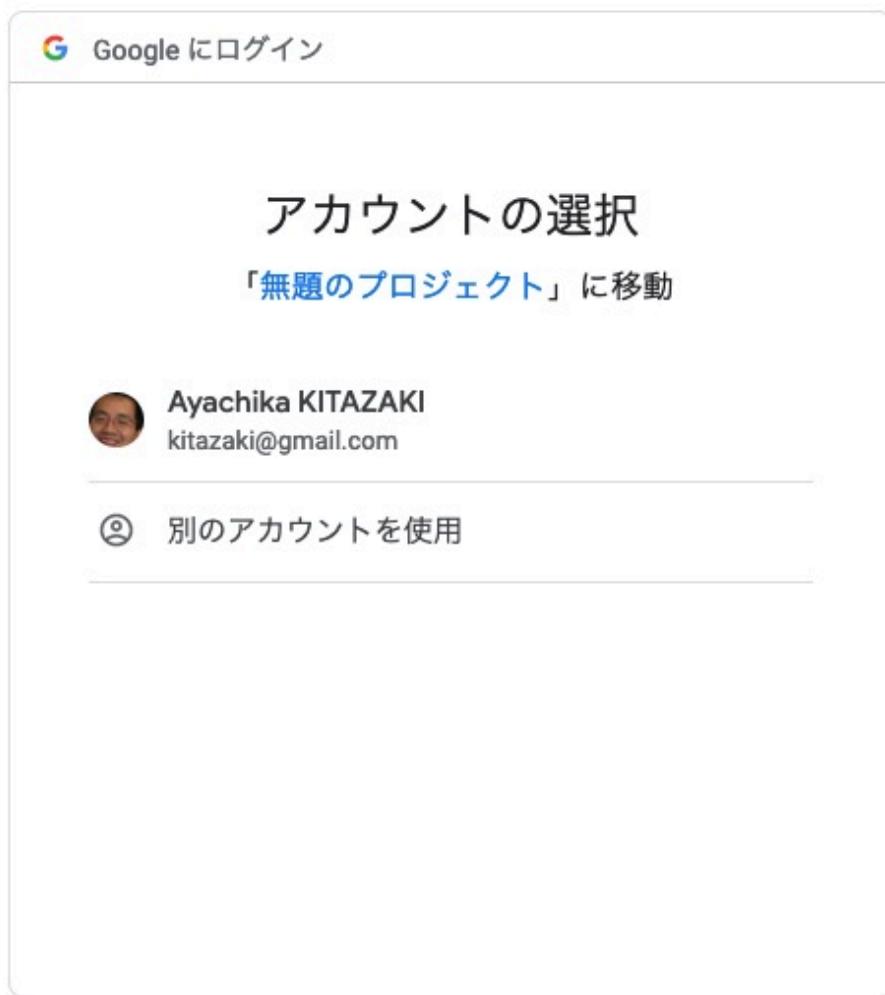


図 6.16: アカウントの選択

「詳細」を押したあと、「(安全ではないページ) に移動」を選択します。

第6章 スマートマットをハックしてさらに便利にする



このアプリは Google で確認されていません

アプリが、Google アカウントのプライベートな情報へのアクセスを求めてています。デベロッパー (kitazaki@gmail.com) と Google によって確認されるまで、このアプリを使用しないでください。

[詳細](#)

[安全なページに戻る](#)

図 6.17: 詳細の選択

リスクを理解し、デベロッパー (kitazaki@gmail.com) を信頼できる場合のみ、続行してください。

[無題のプロジェクト（安全ではないページ）に移動](#)

図 6.18: （安全ではないページ）に移動の選択

外部サービスへの接続で「許可」を押します。

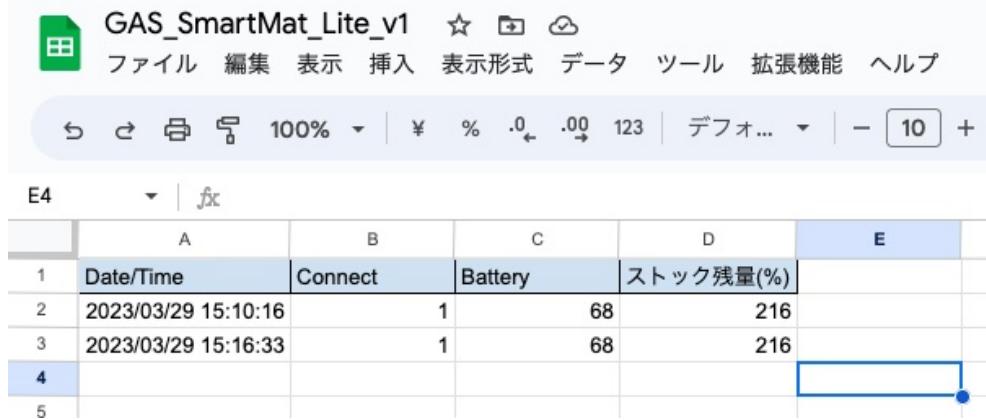
6.8 GAS の実装



図 6.19: 外部サービスへの接続

第6章 スマートマットをハックしてさらに便利にする

GAS の実行が正常に完了すると、スプレッドシートにスマートマットライトのデータが記録されるので、記録されたデータを用いてグラフを作成します。



1	Date/Time	Connect	Battery	ストック残量(%)
2	2023/03/29 15:10:16		1	68
3	2023/03/29 15:16:33		1	68
4				
5				

図 6.20: スプレッドシート

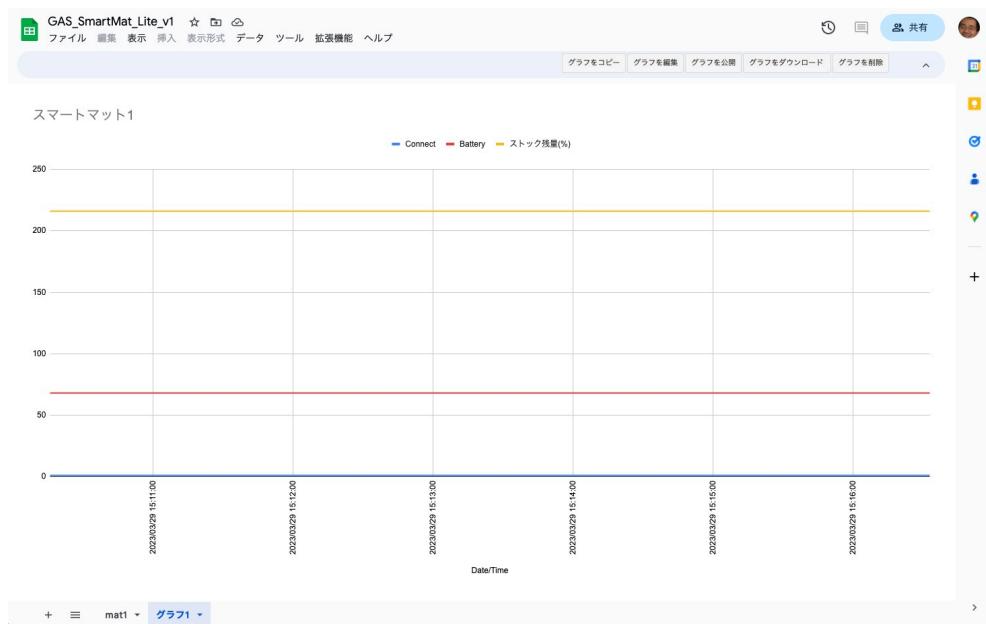


図 6.21: グラフ

GAS を定期実行する場合、トリガーを設定します。時計マークを押します。

6.8 GAS の実装



図 6.22: トリガーの設定

+ トリガーを追加

図 6.23: トリガーを追加

- ・「イベントのソースを選択」 → 時間主導型
- ・「時間ベースのトリガーのタイプを選択」 → 時間ベースのタイマー
- ・「時間の間隔を選択 (時間)」 → 1 時間おき

を選択し、「保存」を押します。

第6章 スマートマットをハックしてさらに便利にする



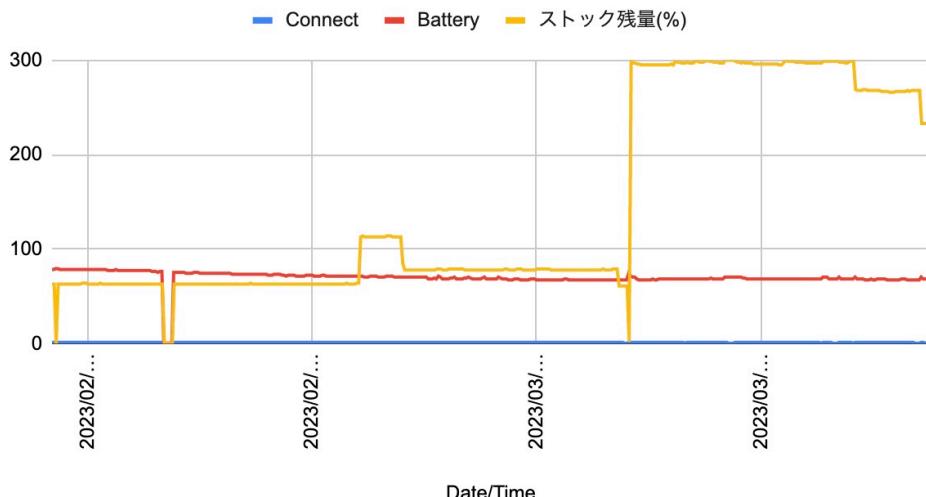
図 6.24: トリガーを保存

6.9 結果表示

在庫の消費傾向に加えて、Wi-Fi 接続状況と電池の消費傾向も把握できるようにしました。これで在庫が無くなる時期を予測したり、使い過ぎを警告することもできそうです。

6.9 結果表示

マット1



マット2

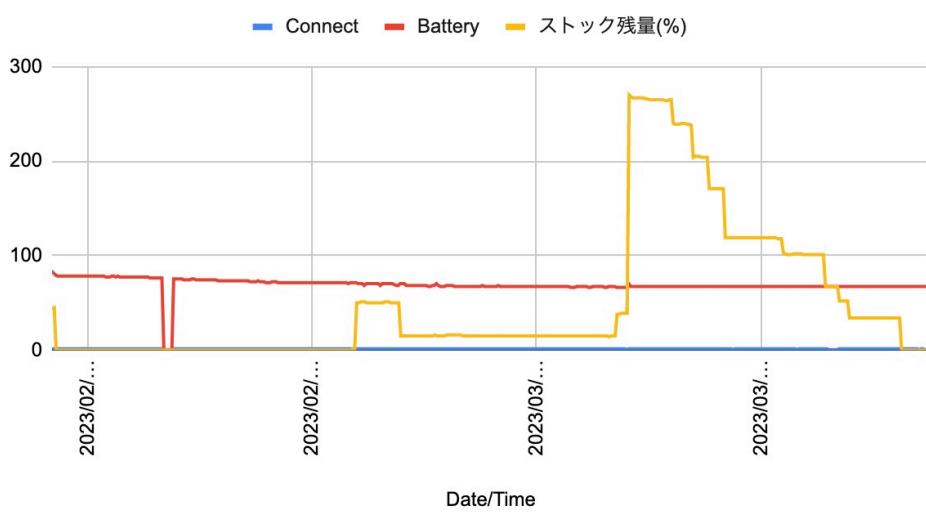


図 6.25: Wi-Fi 接続状況・バッテリー容量・ストック残量のグラフ

第6章 スマートマットをハックしてさらに便利にする

6.10 さいごに

スプレッドシートと GAS を使用して、スマートマットライトをハックしてみました。
みなさんもぜひ、いろんな IoT デバイスをハックして自動化を楽しんでみましょう！

第7章

RPAによるリアルタイム処理

鎌田 誠

7.1 RPA のメリット

RPA の導入目的として「生産性向上」と「品質向上」があります。24時間365日、文句も言わず動いてくれるRPAは、まさに生産性向上のために産まれてきた申し子のようなものですね。品質向上についても、ヒューマンエラーをしっかりと排除してくれて、常に一定の品質を確保できます。

本章では、その一方であまり注視されていないRPAによる「リアルタイム処理」について語りたいと思います。

リアルタイム処理を実装することで、これまで実施できなかった業務ができるようになるなど、大きなメリットもありますので、是非最後までお読みいただければ嬉しく感じます。また本章ではPower Automate for desktopで作成したサンプルフローも掲載致します。その他のRPA製品でも同様のフローは作れますので、よかつたら参考にしてください。

7.2 RPA のリアルタイム処理とは？

7.2.1 リアルタイム処理を知る

製造業でのPRA導入事例として「作業日報の入力作業自動化」など、よくあがる事例です。

工場では、工程1→工程2→工程3→完成品のように、いくつかの工程を通して製品が生産されます。例えば工程2のYさんが機種X1を100台、機種X2を200台、機種X3を35台の作業をしたとしたら、その実績を作業日報に記入します。

第7章 RPAによるリアルタイム処理

作業日報						
部署名	工程2					
作業者名	Y					
日付	2023/4/18					
承認	審査	作成				
項目	加工指示番号	機種名	オーダー数	開始時間	終了時間	実数
1	WO-202304170010	X1	100	8:00	12:00	100
2	WO-202304170011	X2	200	13:00	16:00	200
3	WO-202304170012	X3	100	16:00	17:00	35 残数65は、明日対応
4						
5						
6						
7						
8						

図 7.1: 作業日報

各工程ごとにその作業日報を取りまとめて（例えば、工程2ではYさん含めて10名の方が作業をしていれば、10名分の作業日報）、基幹システムに投入します。作業日報ができるのは作業が終ってからですので、投入できるのは夕方か夜になりますよね。

そうすると、基幹システム上正しい在庫が見えるのは全ての投入が終ってからになります。つまり、翌朝にならないと基幹システム上では正しい在庫が見れないのです。

これは正しい姿でしょうか？

常に最新の「在庫状況」が基幹システム上で見えることで、最適な判断ができます。最新の在庫が基幹システム上で見えない以上、現場に赴き在庫状況を確認しなければならず、非常に無駄な作業となります。

しかし、現実はそこに人手を割いてでも最新の情報にするのが難しく、結果、翌朝にならないと在庫が見えない状況になっている工場も多いです。

こういったケースにおいて、RPAによるリアルタイム処理が活きてきます。必要な時にシナリオを動かすのではなく、リアルタイム処理で常にシナリオが動き続け、必要な条件が揃った際に必要な処理をさせることができます。

先ほどの例では、Yさんが各機種の作業を完了した際にシナリオが自動的に動き始める仕組みです。これにより、常に最新の在庫状況が見れるようになります。

7.2 RPA のリアルタイム処理とは？

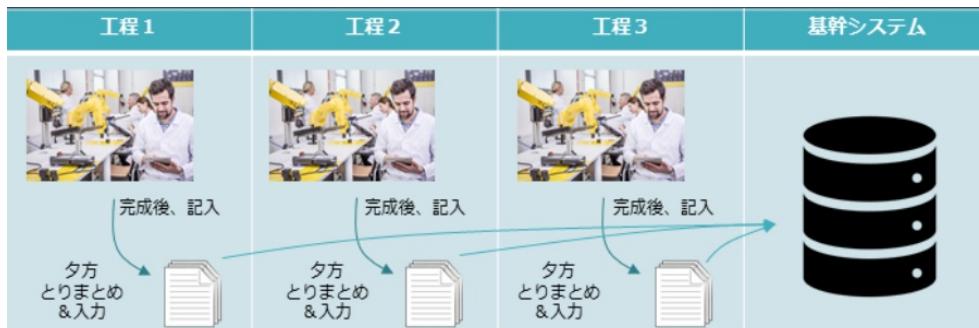


図 7.2: 工程移動フロー (改善前)

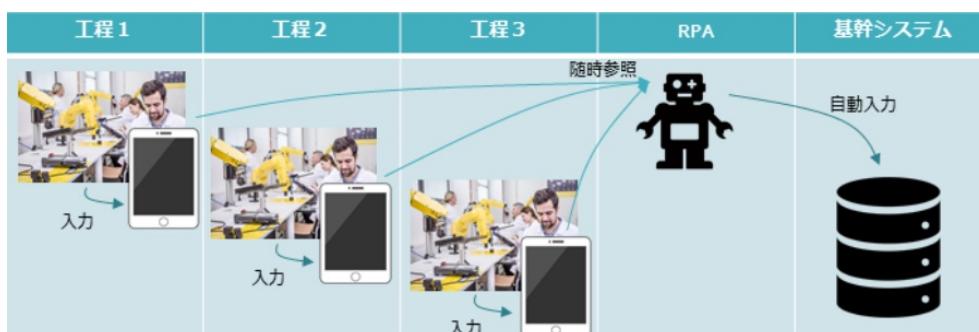


図 7.3: 工程移動フロー (改善後)

7.2.2 リアルタイム処理のメリット・デメリット

このように、本来実施したかったにも関わらず、人が実施するには現実的ではないような作業でも、RPA なら実現可能なのです。ただし、RPA でリアルタイム処理を実現させるためには、いくつかのメリット/デメリットがあります。

メリット

1. 常時動いているため、都度シナリオを実行する手間がない。
2. 必要なタイミングで自動で処理を実行してくれる
3. 処理 A, 処理 B 等複数の処理を埋め込むことで、1ライセンスでも複数処理が可能

デメリット

1. トリガーとなる電子的な仕組みが必要
2. シナリオが複雑化しやすく、エラー時の原因がわかりにくく

第7章 RPAによるリアルタイム処理

3. RPAでPCを占有してしまうため、RPA実行専用のPCが必要となる

このようにみると、良いことばかりではありません。特にシナリオが複雑化しやすく、トラブル時の復旧時間が長くなりがちです。その分、現場には負担を強いることになるので、可能な限りエラー原因が追跡できるように情報を残すようにし、復旧時間の短縮を意図した作りとしておきましょう。

7.3 リアルタイム処理の実装

リアルタイム処理を実装するには、処理が実行されるための条件が満たされるかどうかを常に監視する必要があります。そのため、無限ループを組んで、その中で発動条件を常に監視します。

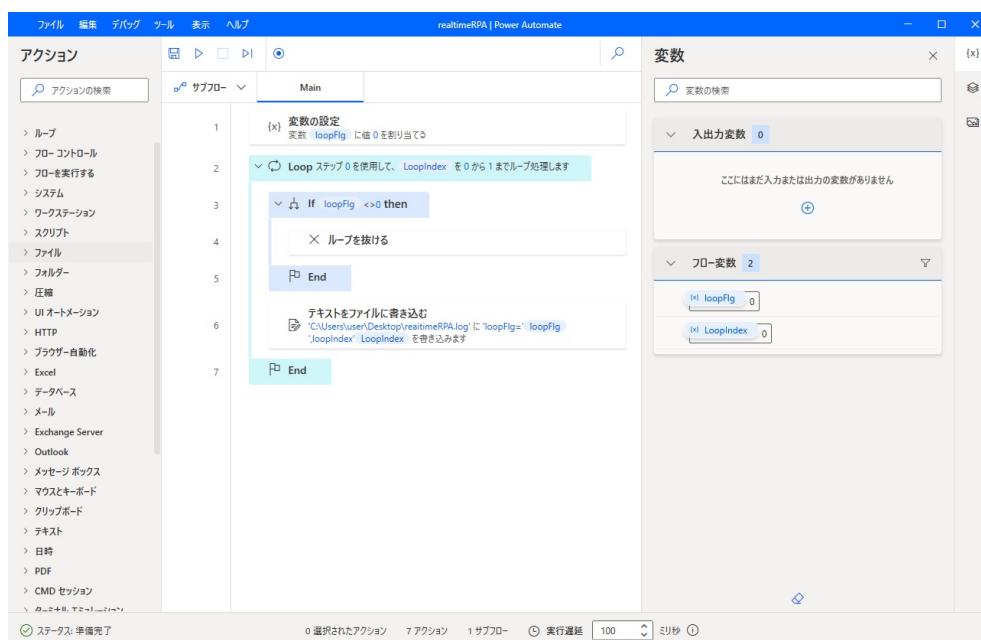


図 7.4: Power Automate for desktop による無限ループシナリオ

まず、loopFlg を初期化 (0 を代入) します。次に、loopFlg が 0 以外の値になったらループを抜けるように Loop 処理を追加します。Loop の中では、loopFlg の値を確認し、0 以外の時にループを抜けるようにしています。また、Loop 内では、ログファイルに loopFlg の値と、loop 回数 (loopIndex) を出力しています。

この Loop 処理の中で、特定の条件が満たされた際に、所定の処理を実行するシナリオを作成すれば良いわけですが、特定の条件とは、どのようなものがあるでしょうか？ た

7.3 リアルタイム処理の実装

とえば……

- 10時、12時、15時のような特定の時間を迎えた時
- 5分毎、10分間毎のような特定の周期を迎えた時
- あるファイルが生成された時

ぱっと思いつくのはこれぐらいでしょうか？

ここでは、「ファイルが生成された時」をトリガーとして処理するシナリオを考えていきましょう。

先ほどの作業日報の例で行くと、トリガーとしてはYさんの作業を完了させたタイミングになります。機種X1 100台分の作業が完了した際に、基幹システムにX1 100台と投入することで、在庫がリアルタイムで見えるようになります。そのためには、実績値を電子化する必要があります。筆者の例だと、データを入力するだけの簡単なフォームを作り、CSVファイルに出力するようにしました。今なら、Microsoft 365のFormsを使ったり、チャットツールから入力し、Power AutomateでCSVファイルを出力する、なんてやりかたもありそうですね。

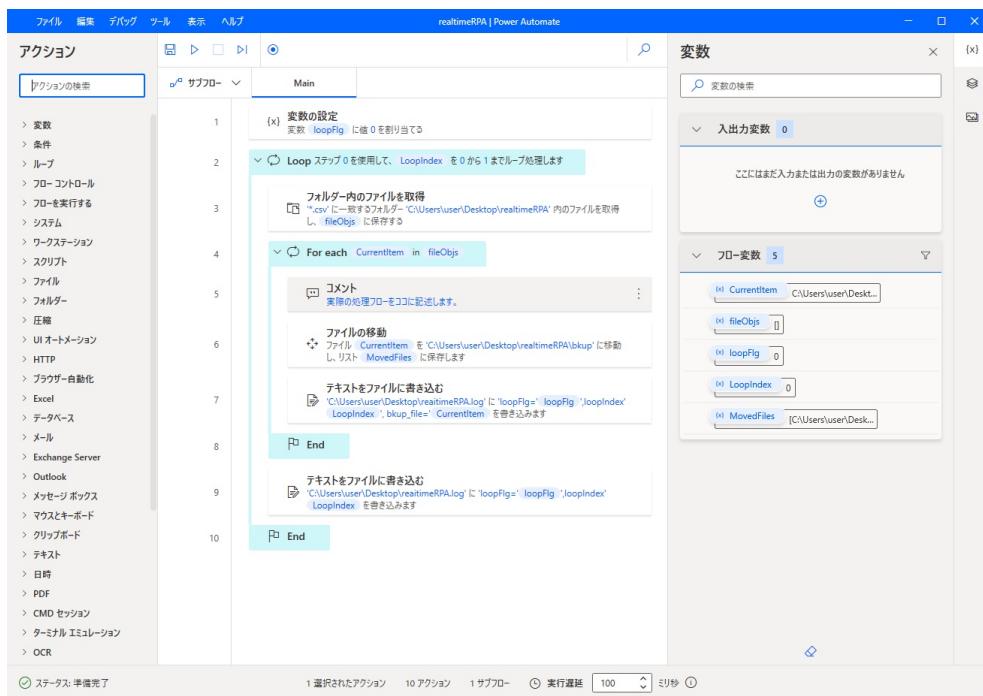


図 7.5: CSV ファイルが生成された際のロジック

ここでは、指定したフォルダ内に CSV ファイルがある場合に、コメント「実際の処理

第7章 RPAによるリアルタイム処理

フローをココに記述します。」の箇所に処理フローを登録していきます。例えば、これまでの例のように作業日報の登録であれば、CSV ファイルを読み取り、基幹システムを操作して転記と登録を行います。併せて処理済みのファイルは bkup フォルダに移動させます。これは同じファイルで二重登録を避ける目的と、実際に登録したデータのエビデンスを残す意味があります。またログファイルも出力しています。ログファイルについては後ほど説明致します。

このようなフローを組むことで、基幹システムへの登録作業を RPA が人知れずリアルタイムに行っててくれるようになり、これまでどうしても実現できなかったリアルタイムでの正しい在庫管理ができるようになりました。

7.4 リアルタイム処理の適用事例

RPA におけるリアルタイム処理の事例を見てみましょう。

7.4.1 製造業における工程間在庫移動

これまでの事例としてあげてきたものが、製造業における在庫移動の事例になります。実際に筆者が情シス時代に構築したシナリオで、数値もかなりリアルな値になっています。では改めて処理フローを見てみましょう。

7.4 リアルタイム処理の適用事例

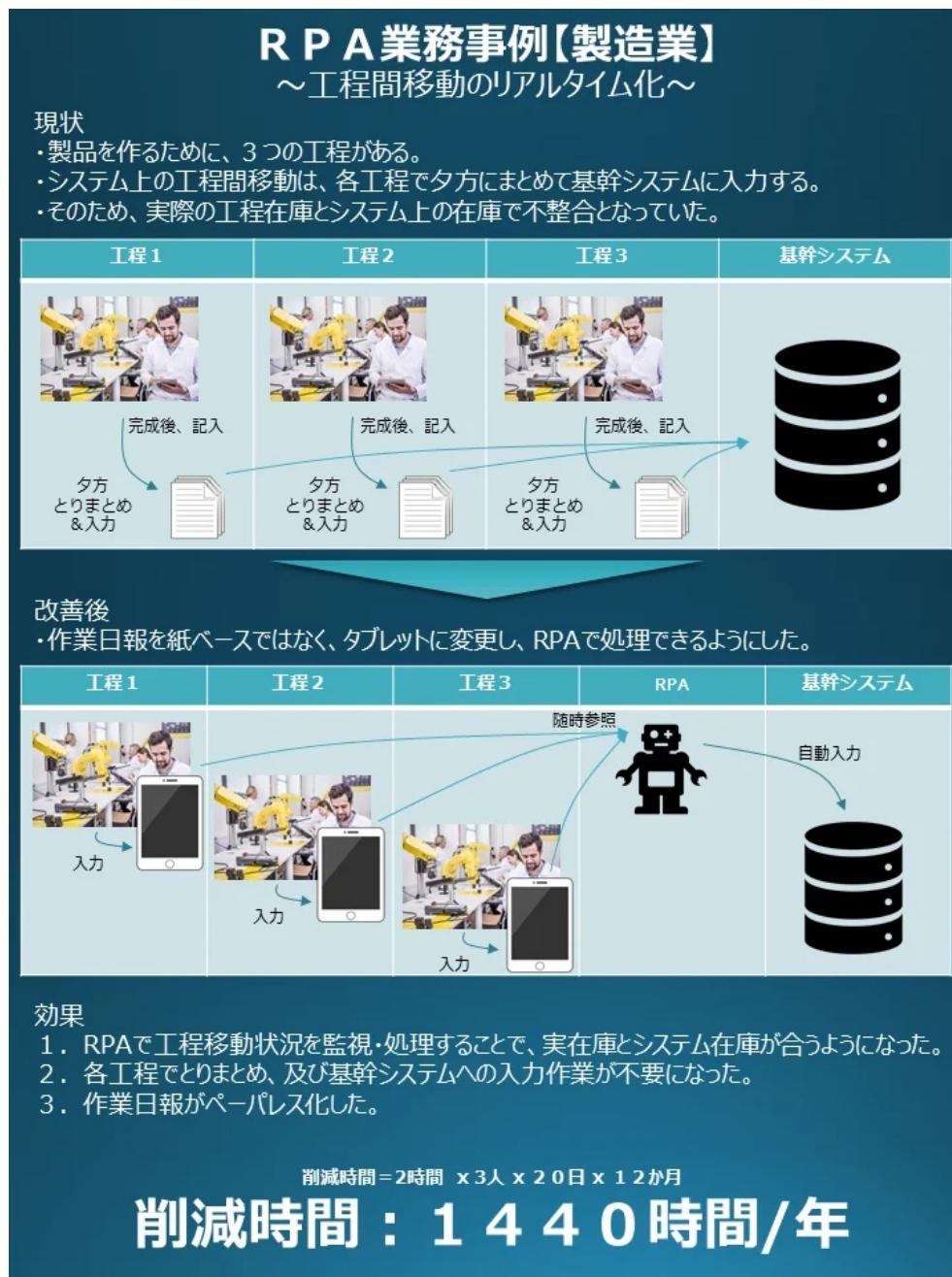


図 7.6: 工程間移動の実例

7.4.2 製造業における完成品入力

前項は在庫間移動でしたが、ここでは完成品入力になります。製造業においては、1つの機種を生産するのに、加工指示書と呼ばれる書類が発行されます。各工程は、この加工指示書に基づいて実際の生産を実施します。こちらも、私が情シス時代に実際に構築したシナリオです。完成品入力は、現場応援のため総務部門で入力することになっており、加工指示書と作業日報の物理的な移動（現場→総務）も発生していました。

7.4 リアルタイム処理の適用事例

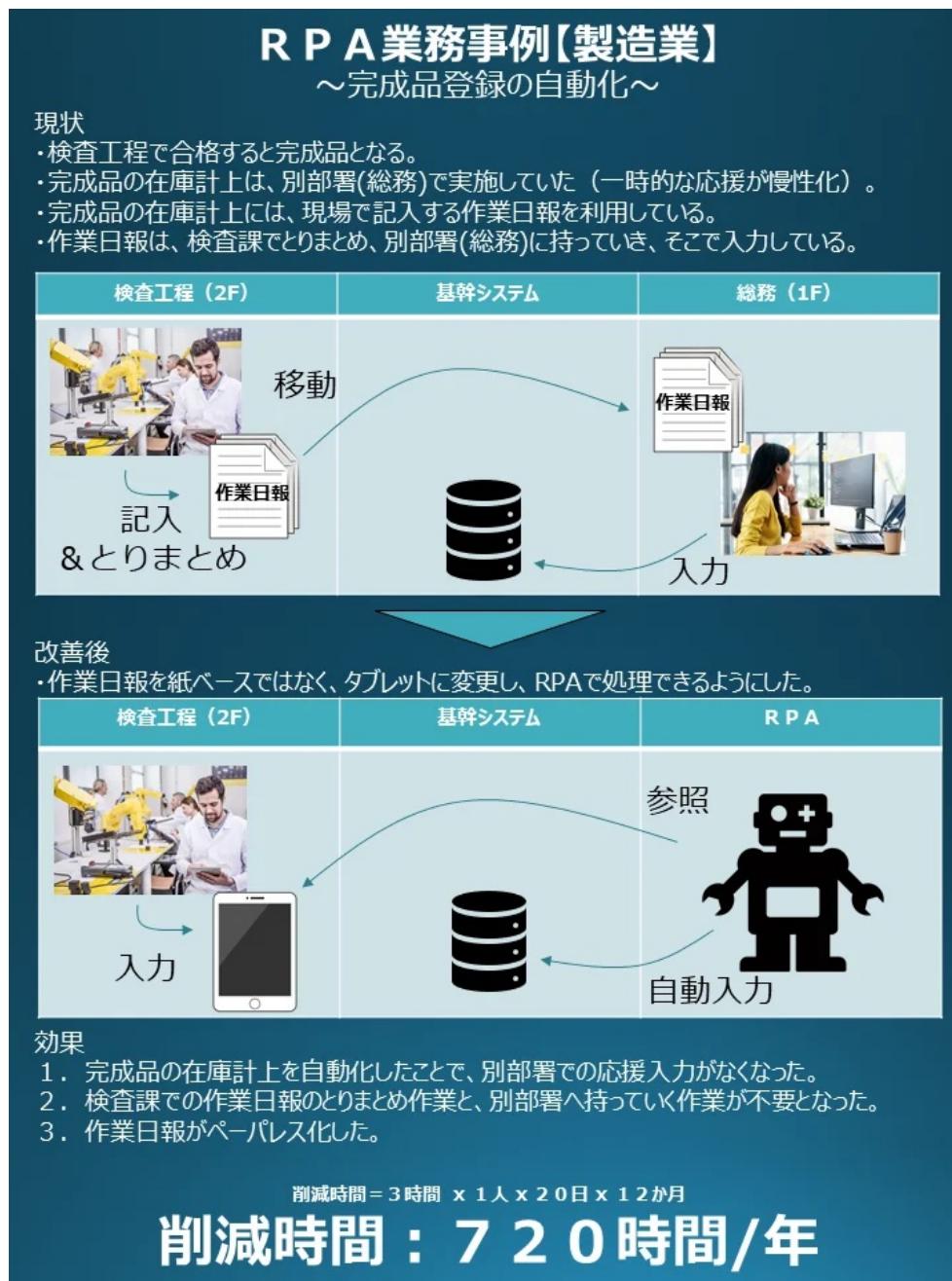


図 7.7: 完成品登録の事例

7.5 リアルタイム処理の課題と解決策

リアルタイム処理は、パソコンを占有してしまうため、常に人が見続ける（エラーなく処理が終わるか監視する）ことができません。そのため、エラーが発生した際に如何に早く気づくことができるか、またなぜエラーが発生したのかを知ることが大切です。

7.5.1 エラーを素早く知るには

リアルタイム処理で動くシナリオは、通常、人が監視をしていないところで動きます。よって、異常終了等でシナリオが止まっていても知ることができず、現場の方からの指摘で知ることになってしまいます。そうならないためには、エラー時はしっかりとエラーをトラップし、管理者へ通知することが大切です。また、通知も一人だけではなく、関係者複数人に送るようにしましょう。



図 7.8: エラー時の処理例

7.5 リアルタイム処理の課題と解決策

こちらのフローは、エラー発生時に処理されるフローになります。ログファイルへの書き込みと、画面キャプチャを取得し、Outlook を使ってメール送信をおこなっています。今回はメールとしましたが、Teams 等のチャットツールに送る方が、より早く気づいてもらえるメリットがあります。

7.5.2 エラーの原因を知るには

エラーでシナリオが止まってしまった場合を想定して、シナリオのどこまで進んだのか、その際の変数の状況はどうだったのかをファイルに保存しておくようにしましょう。また可能であれば、エラー時の画面キャプチャも保存しておくと、ログだけでは分からぬ状況が見え、解決のためのヒントとなる場合も多いのでオススメです。

実際にエラーで止まってしまった場合は、取得したログやキャプチャ画面等をヒントに原因を特定し対処していくことになります。そのために、ログファイルに記録すべき情報を検討します。一般的には、下記のような項目になります。

- ログの種類
 - シナリオ名
 - 発生日時
 - 直前の処理名（行番号など）
 - エラーメッセージ
 - 変数の値
 - (などなど)

図 7.9 の左側は、図 7.8 で示したフローにあるログ情報をログファイルに保存するノードの画面です。一方、右側はエラー発生時のスクリーンショットの記録です。

第7章 RPAによるリアルタイム処理

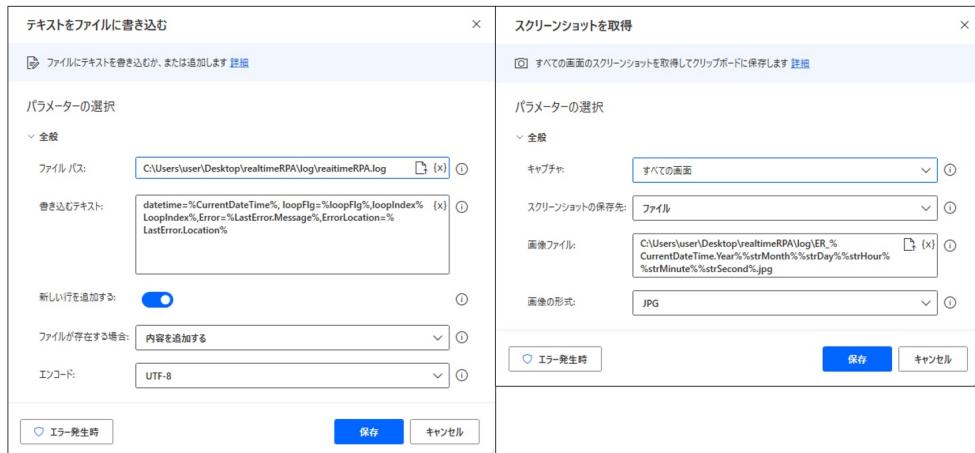


図 7.9: エラー時のログ取得

シナリオをリリースする前に、トラブル対応に必要な情報（ログ）が十分に出力されているか確認しておきましょう。

また、登録が目的のフローであれば、登録がどこまで済んだのかの確認も必要となります。実際の登録状況と、監視フォルダや bkup フォルダ、またログファイルの内容を見比べ、どこから処理を再開させれば良いのかを判断する必要があります。

7.6 リアルタイム処理の今後の展望

RPAにおいては、生産性向上、品質向上の2点に注目が集められていましたが、このようにRPAを活用することで、人手ではできなかつたことが実現でき、あるべき姿に近づくことができます。全てのシステムがクラウド化されれば、クラウドフローで事足りることかもしれません、まだまだオンプレミスの仕組みも多いですし、クラウドからオンプレミス回帰の流れもできつつあります。また、AIによる自動化等もRPAには追い風だと思っています。

7.7 最後に

今回記載した内容は、一部 note記事にしてあったり、過去のRPACommunityのライセンシングトーク大会でも発表した内容になりますので、併せてそちらも参照頂けると幸いです。

note

RPA事例集について

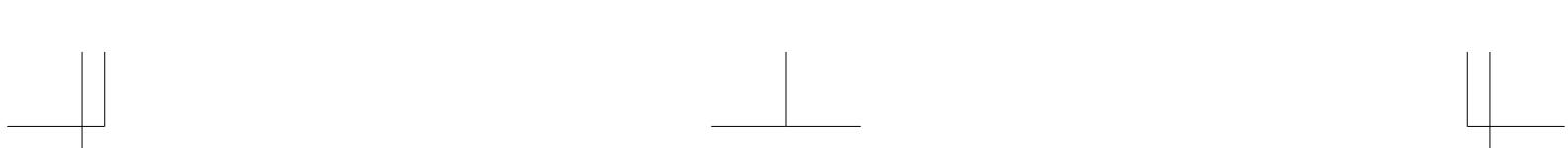
7.7 最後に

<https://note.com/kerdy/n/nd2f20dad64f2>

Youtube

RPA でリアルタイム処理

<https://youtu.be/MMfycoq9tsc?t=789>



第8章

Raspberry Pi と GCP を使って、 SMS の E2E 監視を実装してみた！

瀧川 大樹

8.1 はじめに

こんにちは！ 私は普段、某通信事業者のショートメッセージングサービス（以下 SMS と記載）を提供するシステムの保守業務をしています。本章では Raspberry Pi を使って SMS の送受信テストを自動化し、GCP の GAE と Cloud SQL で WebAPI を構築して、社内の監視システムに連携させた事例について紹介します。

8.2 E2E 監視の重要性と構築した経緯

なじみがない方のために、SMS について少し紹介させて頂きます。SMS は電話番号を送信先情報に指定し、テキストメッセージをやり取りするメッセージングサービスです。ユーザコミュニケーションの用途としての利用は減っていますが、Web サービスの 2 段階認証の際のワンタイムパスワードをユーザへ送付するような、システムからユーザに送られるケースについての利用は増えています。社内では SMS 関連のシステムは重要なインフラとして定義され、障害が発生した場合については、迅速な状況確認と復旧が要求されます。私たち運用担当は障害対応の際、まず、お客様が実際にサービスを使っているかどうかを最初に確認します。ここで、お客様が実際にサービスを使っているのか確認するために、End to End の監視（以後、E2E 監視と記載）は非常に重要です。

E2E 監視とは、システム外部からユーザと同様の方法でアクセスして、そのシステムが正常に稼働しているか確認するための監視です。障害が発生した際に手作業で端末を使って試験をしていると障害検知・復旧が遅くなり、お客様と約束している品質を守る事が難

第8章 Raspberry PiとGCPを使って、SMSのE2E監視を実装してみた!

しくなります。そのため、私たちの部署では、システムの導入の際には、E2E監視の実装をする事を必須の条件としています。E2E監視の実装の内容としては、サンプルとしてお客様のリクエストをエンドポイントに実行し、その応答結果や応答時間を監視しています。

以前まで、SMSのE2E監視は、SMSのE2E監視専用のサーバを構築して実現していました。SMSが利用するプロトコルは非常にニッチな事もあり、SMSのE2E監視専用のサーバはベンダーが構築したものを利用していました。最近、このSMSのE2E監視サーバのサポート期間が終了することになり、構築をしたベンダーからリプレイスの提案を頂きました。その提案頂いた費用が非常に高額であったため、自社内製でSMSのE2E監視を実装する検討を開始しました。

8.3 監視の構成

E2E監視システムは、SMSの送受信を行うSMS送受信シミュレータ、監視結果を表示する監視ダッシュボード、シミュレータと監視ダッシュボードを連携させるWeb APIの3つで構成されています。以下、それぞれについて少し詳しく説明します。

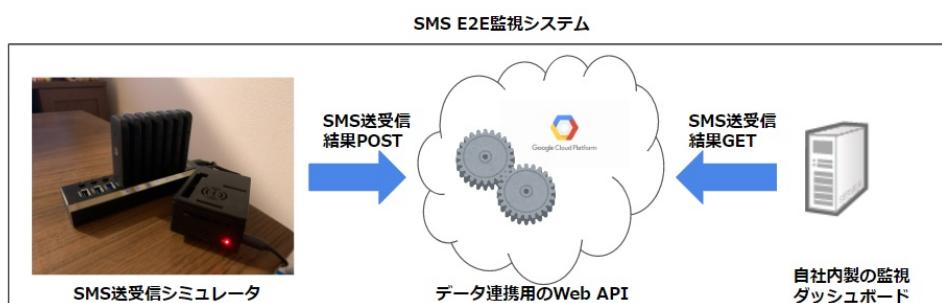


図 8.1: E2E監視システムの構成

8.3.1 SMS送受信シミュレータ

SMSのE2E監視を実現するために、SMSの送信・受信をシミュレートする方法を検討しネットで調べたところ、Raspberry Pi^{*1}とUSB接続のLTEドングルを使ってSMSを送受信を試しているネット記事を見つけました。Raspberry Piとは、教育目的に製造された小型コンピュータで、現在普及が進んでおり教育分野以外も様々な分野で利用され

^{*1} https://ja.wikipedia.org/wiki/Raspberry_Pi

8.3 監視の構成

ています。

また、USB 接続の LTE ドングルは、外出した際に Wifi 環境がない場合に利用する USB 型のモデムの事を指します。Raspberry Pi から AT コマンドと呼ばれるモデムを操作するコマンドをシリアル接続でドングルに流し込む事により、通話やインターネット接続や SMS の送受信などの携帯端末の操作を実行する事ができます。今回は、AT コマンドを使った SMS の送受信の処理をスクリプトとして実装する事によって、SMS 送受信シミュレータを実現しました。詳しくは実装パートで説明します。



図 8.2: SMS 送受信シミュレータ

8.3.2 監視ダッシュボード

私の属している部署で運用・保守しているサービスは、自社内製の監視ダッシュボードで E2E の監視の状態を確認できるようになっています。監視の方式は統一されている事が望ましいと考えたため、SMS 送受信シミュレータの送受信結果は監視ダッシュボード上に表示させる事にしました。ダッシュボードは社内の商用ネットワークに属しており、SMS 送受信シミュレータとダッシュボードを直接連携させるのは、セキュリティ上避

第8章 Raspberry Pi と GCP を使って、SMS の E2E 監視を実装してみた!

けたいと考えました。連携については、後述するデータ連携用の Web API を利用して連携するような方式としました。

8.3.3 データ連携用の Web API

シミュレータでの SMS の送受信結果をダッシュボードに表示させるために、部署で契約している Google Cloud Platform(GCP) 上に Web API を構築しました。社内でのクラウド利用については、2つの要件を満たせば利用が可能となります。Cloud 環境を社内で使ってみたいと考えている方は、実装の前に、会社の利用ルールや部門の予算の状況を確認する事をおすすめします。

8.3.4 予算の確保

クラウド利用に当たって、部門でクラウド利用のための予算を取っているため、利用に当たってどのくらいお金が当たるかを部門内で説明が必要でした。

8.3.5 社内のセキュリティガイドラインとセキュリティポリシーを満たす事

実装に当たっては社内のセキュリティ部門が提供するセキュリティガイドラインを満たすように実装する必要があります。また、実装後に、社内のセキュリティポリシー的に運用前には必ず脆弱性診断を受け、診断結果が問題ない事を確認して初めて、実装したものを使って運営開始する事ができます。また、運用を開始してからも年次で脆弱性診断を実施する必要と月次でセキュリティパッチの適用調査をする必要があります。

8.4 SMS 送受信シミュレータの実装

SMS 送受信シミュレータの要件は大きく 2 つあります。1 つ目は、SMS の送受信を実行できる事、2 つ目はインターネットに接続し、SMS の送受信結果を Web API を使って登録できることです。

8.5 SMS 送受信スクリプトの処理概要

1 つ目の要件を満たすために、SMS 送受信スクリプトを実装しました。本項では、SMS 送受信スクリプトの処理の概要について説明します。

8.5 SMS 送受信スクリプトの処理概要

8.5.1 AT コマンド投入用の USB デバイスファイルの認識

ドングルを Raspberry Pi に接続すると、デバイスファイルが複数作成されます。スクリプトからシリアル接続する際にライブラリでエラーが出力されるため、wvdial^{*2}用と AT コマンド登録用で USB デバイスファイルを分ける必要があります。そのためスクリプト上では、AT コマンド投入用のデバイスファイルのパスを定義しています。複数のドングルを接続すると、デバイスファイルとドングルの紐づけができなくなるため、デバイスファイルを固定化する対応が必要となります。具体的には、udev ルールに定義する事によってデバイスファイルを任意の名前に固定化することができます。USB の接続ポートやドングルのシリアルの情報で固定化できるのですが、利用しているドングルはシリアルの情報がなかったため、USB 接続ポートで固定化を実施しています。

8.5.2 SMS の送受信

SMS を送受信する前に、SMS の送受信できる状態であるかどうか、ネットワーク状態(以下 NW 状態と記載)の判定を実施しています。NW 状態の判定には、AT コマンドで電波強度確認、在圏状態の確認を実施しています。また、SMS の送受信を可能にするために、SMS モード設定コマンド等を実施しています。ドングルで SMS を送受信する際に注意が必要な点は 2 つあり、1 つ目は SMS の文字コードで、正しく設定しないと文字化けをします。実装したシミュレータでは GSM に設定しています。2 つ目は SMS の読み込み先、書き込み先の設定です。読み書きでそれぞれ端末か SIM カードで保存するかを設定する事ができますが、書き込み先と読み込み先を一致させる必要があります。今回の場合ドングル上か SIM カード上かになりますが、受信したら都度削除するシナリオになっているので、SIM 上を保存先として指定しています。

8.5.3 Web API で送受信結果を Post

SMS の送受信の試行を実施後、Web API のユーザとパスワードで認証トークンを取得し、認証トークンで SMS の送受信結果を POST します。工夫したところは WebAPI のユーザとパスワードの秘匿性を上げるために、パスワード管理モジュールを使って、スクリプト内に認証情報を直接埋め込まないようにしています。スクリプト内にパスワード等の情報を埋め込んでしまうとスクリプトが誤って外部に公開された場合、認証が危険化するリスクがあると考えたため、実装はやや煩雑になりましたが対策を実施しました。

^{*2} <https://wiki.archlinux.jp/index.php/Wvdial>

8.6 インターネット接続処理の概要

2つめの要件を満たすために wvdial と呼ばれる Point-to-Point Protocol ダイアラをインストールしてインターネットへの接続を可能としています。インターネットへの接続には、wvdial の設定ファイルの編集と AT コマンドを使ってドングルへの設定投入が必要でした。設定ファイルには、ドングルのデバイスファイルのパス、ドングルに搭載した SIM の APN の情報を設定します。次に、ドングルへの設定についてですが、ドングルに AT コマンドを投入して通信事業者のネットワークに接続します。設定に必要な AT コマンドを「8.7.1 APN の設定」に記載しています。シミュレータは Quectel 製の LTE チップが実装された、SORACOM Onyx LTE USB ドングル^{*3}を利用しておらず、「8.7.2 APN のユーザとパスワードの設定」と「8.7.3 Packet Data Protocol(PDP) 有効化」については、Quectel 製のオリジナルのコマンドとなります。

8.7 ドングルに設定必要な AT コマンド

8.7.1 APN の設定

AT+CGDCOUNT=1,"IP", "利用する APN 名"

8.7.2 APN のユーザとパスワードの設定

AT+QICSGP=1,1,"APN 名", "ユーザ", "パスワード", 0

8.7.3 Packet Data Protocol(PDP) 有効化

AT+QIACT=1

8.8 Web API の実装

Web API は GCP 上の Google App Engine(GAE)^{*4}と Cloud SQL^{*5}を使って実装しました。また、運用の負荷をかけずに、実装は柔軟にしたいという考えで PaaS を利用して実装しており、GAE は認証 API と登録 API と結果取得 API を提供しています。また、Cloud SQL にはシミュレータで実行した SMS の配信結果情報が登録 API で保存されています。また、この Web API はインターネット上に公開しているため、以下のよう

^{*3} <https://soracom.jp/store/7326/>

^{*4} <https://cloud.google.com/appengine?hl=ja>

^{*5} <https://cloud.google.com/sql?hl=ja>

8.8 Web API の実装

なセキュリティの対策を行っています。

8.8.1 ファイアウォールで接続 IP を制限

GAE はファイアウォールの機能を提供しており、GAE 上にアクセスする IP を制限することができます。今回接続するシミュレータの IP とダッシュボードの IP に接続を制限しています。接続するシミュレータの IP は事業者毎によって振られるものとなるため、広いレンジで制限をかけています。

The screenshot shows the 'Firewall Rules' section of the App Engine dashboard. On the left, there's a sidebar with links like Dashboard, Services, Versions, Instances, Task Queue, cron jobs, Security Scan, and Firewall Rules. The Firewall Rules link is highlighted. The main area has tabs for 'Firewall Rules' (selected), 'Logs', and 'Metrics'. It includes buttons for 'Create Rule' and 'Test IP Address'. A table lists rules with columns for Priority (升順), Action (許可 or 拒否), IP Range, and Description (省略). The rules listed are: 500, 800, 900, 1000, 1200, and Default (拒否).

優先度 ↑	アクション	IP範囲	説明
○ 500	許可		⋮
○ 800	許可		⋮
○ 900	許可		⋮
○ 1000	許可		⋮
○ 1200	許可		⋮
○ デフォルト	拒否		⋮

図 8.3: ファイアウォールルール画面

8.8.2 認証キーの提供と有効期限の短時間化

結果取得 API、認証 API を実行するためには、認証 API で取得した認証キーを必要とするように実装しています。また、そのキーの有効期限については非常に短い期間に設定しています。有効期限を長くすると、キーが第三者に漏洩した場合、情報漏洩や不正アクセスのリスクが高まるためです。

8.8.3 入力値のバリデーション機能とプレースホルダの利用

SQL インジェクションの対策として、スクリプトではバリデーション機能とプレースホルダを実装しています。バリデーションは公開されているオープンソースのライブラリを利用して実装しています。API の入力値についてルールを設定し、ルールに違反する入力値に関しては、エラー応答を返すようにしています。プレースホルダについても公開さ

第8章 Raspberry PiとGCPを使って、SMSのE2E監視を実装してみた!

れているオープンソースのライブラリを利用して実装しています。入力値内に特殊文字があればエスケープを実施し、想定外のSQLが実行される事を防ぎます。

8.8.4 リクエスト数とリクエストエラーの監視

GCPが提供するCloud Monitoring^{*6}でリクエストエラーとリクエスト数を監視しています。Cloud MonitoringではAPIの実行ログやメトリクスを確認できます。また、Cloud Monitoringが提供するアラート機能によって、メトリクス等に閾値を設定し、リクエストのエラーやリクエスト数が設定した閾値を超過した場合は、メール等で異常を知ることができます。



図 8.4: アラート機能画面

8.9 自社内製の監視ダッシュボード側の実装

シミュレータとWebAPIについては自身で実装をしましたが、自社内製の監視ダッシュボードの実装については、実装をする専門の部隊がいたため、実装を依頼しました。監視ダッシュボードに実装が必要な機能は、定期的に結果取得APIでSMSの配信結果を取得し、ダッシュボード上にOK or NGを表示させるというものです。実装を専門部隊にお願いするにあたって、E2E監視の構成やAPIの仕様書を起こして、要件を実装部隊に伝える必要がありました。E2E監視の構成やAPIの入出力の情報や期待する処理フロー等をパワポで作成して要件を伝えて、無事に期限内に実装がされました。今回、実装を依頼する際にAPIの仕様をパワポにわざわざ起こして実装を依頼したのですが、

^{*6} <https://cloud.google.com/monitoring?hl=ja>

8.10 初期費用とランニング費用

Google 先生に聞いてみると Swagger^{*7}と呼ばれる、Web API 開発環境がある事を知りました。Swagger を使えば Web API の設計、ドキュメントの自動生成、テスト等が効率良くできそうなので、今後 Web API を構築する機会があれば使ってみたいと思います。

8.10 初期費用とランニング費用

構築と運用にかかる費用としては、以下の初期費用とランニングコストが発生します。基本料と SMS 送信料金のランニングコストについては月 6 万円程かかってます。現状、ランニングコストに月 10 万円程かかっていますが、ランニングコストの累計額が、ベンダーから提示されたリプレイスの見積もりに達するのは、何百年先になるので費用効果については十分あったと考えています。

- 初期費用
 - シミュレータの実装費用
 - SIM 初期契約料
- ランニング費用
 - GAE と Cloud SQL の費用
 - 通信事業者の基本料 + データ使用料
 - SMS の送信費用

以下、それぞれの詳細について紹介いたします。

8.11 初期費用について

シミュレータは Raspberry Pi とドングルの費用で 1 台大体 2 万円ぐらいかかります。半導体不足の影響で Raspberry Pi の値段が上がっているようなので、今はもう少しかかるかもしれません。SIM の初期契約料については、MVNO のソラコムさんが提供している特定地域向け IoT SIM を契約していて、初期費用は DCM の SIM で 3,300 円、AU の SIM で 1,650 円でした。

^{*7} <https://swagger.io/>

8.12 ランニング費用について

8.12.1 GAE と Cloud SQL の費用

GAE については、安定性と不要に料金追加を防ぐためにインスタンスについてはオーバースケールではなく、固定化していて、念のためインスタンス数は 2 つにして冗長化しています。

Cloud SQL に関しては処理も多くなく、ストレージの容量も必要ないので一番低いスペックとしました。クラウド側のランニングコストに関しては、1 カ月に GAE が 2 万円、Cloud SQL が 8 千円程となっています。GAE と Cloud SQL のスペックについては以下の通りです。約 0.5rec/s を処理しています。

- GAE
 - 環境:Standard
 - リージョン:asia-northeast1
 - インスタンス数:2
 - インスタンスクラス:B1
- Cloud SQL
 - リージョン : asia-northeast1
 - DB : MySQL
 - マシンタイプ : 標準
 - vCPU : 1
 - メモリ : 3.75
 - ストレージ : 10G

8.12.2 通信事業者の基本料 + データ使用料

通信事業者の基本料とデータ使用料については、月 300MB 分のデータ使用量が基本料に含まれ、超過した際は追加でデータ料を支払う料金体系となっていて、契約した SIM につき数百円/月の料金が発生します。

8.12.3 SMS の送信費用

SMS の送信費用については、自事業者同士の SMS 送信に関しては無料ですが、異なる事業者同士の SMS 送信については、費用が発生してしまいます。SMS の送信については 1 通 3 円かかり、サービス仕様上の 1 日に送信できる通数の制限が 200 通のため、上限に

8.13 運用について

かからないように送信をしています。

8.13 運用について

SMS E2E 監視の運用について説明します。監視システムに関わるコンポーネントがシミュレータ、GCP、監視ダッシュボードが多い事から誤検知が多くなり、従来より監視品質が低下する事が想定されました。そのため、以下の 3 つの対処を行う事によって、誤検知か障害かを切り分けています。

8.13.1 対処 1: 複数の監視シナリオで監視する。

シミュレータに異常が発生するケースに対する対処です。2 台のシミュレータで監視シナリオを 2 つ用意しておき、2 つの監視シナリオが NG となった際は障害と扱うように運用フローとして立てつけました。この対処により、シミュレータ故障による障害の切り分けが可能となります。

8.13.2 対処 2: 複数の監視手段を用意する

Web API 側に異常が発生するケースに対する対処です。シミュレータ ⇒ Web API ⇒ 監視ダッシュボードとは別の監視手段として、シミュレータからの SMS の送信・受信のログを監視しシステムのアラームとして発報させるようにしています。監視ダッシュボード上 NG かつ上記のログ監視のアラームが発報した場合は障害と扱うような運用フローを立てつけています。

8.13.3 対処 3: 複数のネットワークから API を叩く

監視ダッシュボードに異常が発生するケースに対する対処です。監視ダッシュボードとは別に誤検知確認ツールを Google Apps Script で実装しています。誤検知検知ツールは監視ダッシュボードと仕様は同じで SMS の配信結果取得 API を定期的に叩いて、その結果をスプレッドシート上に表示させるようにしています。監視ダッシュボードと誤検知確認ツール双方 NG の場合障害と扱うようにしています。

運用は少し煩雑となりますが、複数のパターンを用意しておき、組み合わせによって監視の確度を上げる対策をしています。E2E 監視を入れていない場合は、システム監視担当が実端末を使って手動で SMS 送受信のテストを実施していましたが、E2E 監視を導入する事によって、手動での SMS 送受信テストを実施せずに良くなりましたが、現場の工数削減にも寄与できたと考えています。

第8章 Raspberry Pi と GCP を使って、SMS の E2E 監視を実装してみた!

8.14 まとめ

今回、Raspberry Pi を利用した SMS サービスの E2E 監視を実装した経緯、システム構成、システムの実装、運用についてご紹介させて頂きました。シミュレータの実装、GCP の機能、構築の費用、監視・運用等の雑多な内容になってしましましたが、何か一つでも困りごとを解決する糸口になれば幸いです。

第9章

現場で使える Python 自動化入門

青木 敬樹

9.1 はじめに

これを手に取られた方は実務で使用する自動化に興味がある方でしょうか？この章は Python 自動化入門としています。しかし、一般書でも様々な自動化の手法やライブラリの紹介が豊富にされています。そのため本章では、Python を実務で使用した際の私の経験に基づく Tips を詰め込みたいと思います。Python 限定の話もありますが、言語に特定されず別のもので読み変えていただく事もできると思います。本章は

1. 頭をすっきりテスト駆動開発
2. みんなで読もう Sphinx
3. 最後の救い log 取得

の 3 つで構成されています。

9.2 頭をすっきりテスト駆動開発

9.2.1 はじめに

皆さんはテスト駆動開発という言葉をご存知でしょうか？アジャイル開発で有名なこちらのメソッド、提唱したのは Kent Beck です。

まず初めにテスト駆動開発を知る前のこのメソッドに対するイメージは「面倒くさそう」というものでした。これは個人開発程度であればテストという行為そのものが必要ではなかった為であり、また成果物の品質を担保するためのテストとテスト駆動開発を混同していた為です。そのため開発後の工程を取り入れる必要性が判りませんでした。しか

第9章 現場で使える Python 自動化入門

し、今ではその恩恵にあづかっておりそのイメージは「多少手間はかかるが便利で有効」と変わっています。この説ではテスト駆動のメリットとその手法の紹介を行ないたいと思います。

9.2.2 テスト駆動とは？

まずテスト駆動開発自体の紹介を行なう前に、始めに私がなぜテスト駆動開発を必要としたのかの背景を説明したいと思います。皆さんの中で近しい状況があればこの節を読み進めるモチベーションになりますし、逆に読み飛ばす判断材料にもなると思います。（もちろん、是非読んでいただきたい所ですが……笑）

事の始まりは業務自動化用のスクリプトを書いている所からでした。以下のような出力された文字列情報を解析して異常がないか確認するためのスクリプトです。

```
[hoge@hogehoge ~]$ df -h
ファイルシステム      サイズ    使用量   残り   ...
○○○                80G      5G       74. 3G
○○○○               42M      20M      21. 5M
○○○○○○○          112M     5M       96. 4M
○○○○               31M      6M       24M
○○○○               2. 4G     2. 1G     300M
```

図 9.1: 解析対象例

さらっと書けてしまいそうですが、上記の図のような物を判定するときいくつも考えなければならぬことがあります。例えば、

- 入力が想定外の物だったら？
- 予期せぬ内容だったら？
- 条件分岐はあってる？（組み合わせに問題はない？）
- 型変換時ミスはしていない？

とくに Python のように型を明示的に指定しない言語では比較的潜在する想定外のバグが増えやすい傾向にあると思います。もちろん一つずつ抜け漏れなく考えることができれば問題はありません。しかし気をつける事が多いため、実装している間は複雑なイライラ棒をしているような気持ちになりました。さらに弊害として1つの機能を実装するだけでも疲れてしまい、中々思うように開発も進まずフラストレーションのたまる日々でした。

ここまで背景をお話ししましたが、この悩みを要約すると

9.2 頭をすっきりテスト駆動開発

- 沢山注意しながら開発をするのが大変

となると思います。この悩みを解決するのがテスト駆動開発でした。

テスト駆動開発のステップ^{*1}は Kent Beck によると

1. まずテストを 1 つ書く
2. すべてのテストを走らせ、すべて成功することを確認する
3. 小さな変更を行う
4. すべてのテストを走らせ、全て成功することを確認する
5. リファクタリングを行なって重複を除去する

とされています。

ここでの大きなポイントはまずテストを書くという事です。小さく機能を開発しテストを実施するというステップを繰り返せば、ミスをした瞬間にテストが指摘してくれます。始めはテストに引っかかるたびに小言を言われているような気がして億劫でしたが、機能の実装を完了してみると思ったよりも時間がかかっておらず疲れも少なかった事を今でも覚えています。これは頭のリソースを機能開発にだけ使う事ができ、バグへの懸念はテストに指摘されたタイミングで修正するだけで良いというのが理由だと考えています。もちろん小さなスクリプトであればテストを最初に書くため書かない場合より時間がかかることは否めません。しかしそのコードが長くなればなるほど複雑さが増すため、頭のリソースを機能開発にだけ向けることができるテスト駆動開発の方が、効率が上がり開発時間が短くなる可能性があります。テスト駆動開発は言わば長距離走向けの手法と言えるでしょう。

あくまで私の体験談でサンプル 1 のお話でしたが、ここまでテスト駆動開発が広まっている事を考えると試す価値はあると思います。この説ではテスト駆動開発の紹介のみにどまりますが興味を持たれた方は、Kent Beck の「テスト駆動開発」もしくは Robert C Martin の Clean Code 第 5 章を参考にしてみてください。

9.2.3 誰が為のテスト駆動？

さてここまで素晴らしい手法としてテスト駆動開発として説明してきましたが、私はテスト駆動開発は常に使うべきとは考えていません。理由としてはプロジェクトで開発する以外の場面ではテストを成果としづらいと考えている為です。もちろん、チーム内でテストを残す文化があれば別ですが、チームで成果と見なされないものに工数を掛ける行為は中々継続しないものです。

そもそもの話となりますと、テスト駆動開発で作成するテストと品質の為のテストでは

^{*1} Kent Beck (2017) 『テスト駆動開発』 和田 卓人訳, P1, オーム社.

第9章 現場で使える Python 自動化入門

目的が異なります。テスト駆動開発で書くテストは開発者の注意点を網羅していれば良く、カバレッジをそこまで意識しなくても問題はないと思われます。品質担保の為の成果物としてテストを残す為にはカバレッジを上げる必要がありますが、そのカバレッジ向上と成果物の品質の向上は対数曲線的な関係にあると考えており幾ら細かいバグをテストしていてもスクリプトの機能自体は向上しません。

趣味ではなく現場で活用することを考えると工数も切り離せないファクターの一つです。成果に対して労力を必要以上にかける事は無駄であると考えて割り切ることは必要だと考えています。逆に言えば、ある程度品質が求められるのであればそれなりの工数が必要となる為、テストする工程を別で考えた方が良いと思います。

以上の事から、私は求められる品質を考慮したうえで開発の為のメソッドの一つとして効率を上げる為だけにテスト駆動開発を利用する事を心掛けています。

9.3 みんなで読もう Sphinx

9.3.1 はじめに

いきなり恐縮ですが、ツールを作成する事の恩恵は自分で使う場合のみにとどまらないと考えています。そしてむしろその多くは他の方に使ってもらう事により得られるものだと思っています。もちろんツールの種別によるとは思いますが、適切に抽象化されたメソッドは他の人に利用、継承してもらう事で複利的にその真価を發揮し続けます。

しかし、このような文化を形成していくためには誰かにその仕様を適切に伝える必要があります。そのためにはドキュメントが必要となるでしょう。皆さんはツールを作成するときにどのようなドキュメントを作成していますか？ README.md を作成する、エクセルで作成する、テキストで作成するなど様々だと思います。

本節ではドキュメントを作成する為に、Sphinx というドキュメントジェネレータを紹介したいと思います。

ドキュメントジェネレータとはその名の通りドキュメントを作成するためのツールです。百聞は一見にしかずという事で、まずは Sphinx の活用事例を見てみましょう。以下は Open3D という点群処理用のライブラリのドキュメント^{*2}です。

見ていただくとわかる通り html 形式の web サイトのようなドキュメントです。

^{*2} <http://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html>

9.3 みんなで読もう Sphinx

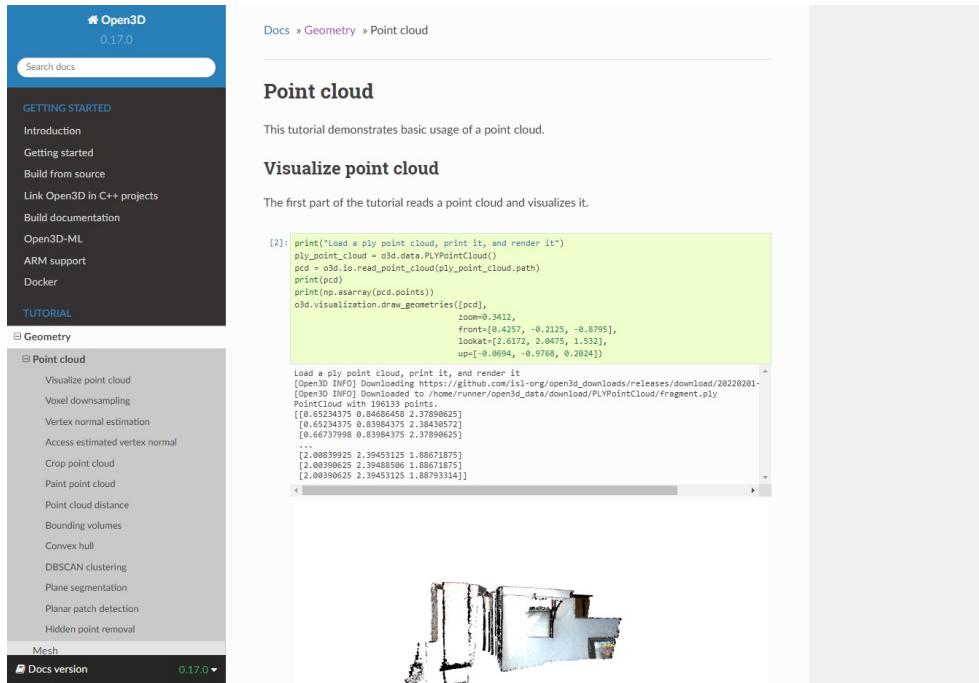


図 9.2: Open3D ドキュメント

一言で言ってしまえば、Sphinx はこの html ドキュメントを作成するためのツールです。（もちろん、html に限らず Latex や ePub, Texinfo, manual pages, plain tex といったフォーマットにも対応しています。^{*3)} 類似のツールとしては Doxygen などが有名です。

Sphinx のようなドキュメント生成ツールでドキュメントを作成した場合のメリットをいくつか列挙してみると、

1. ドキュメントがきれい
2. コメントが残る
3. 検索ができる
4. レイアウトを統一しやすい

といったところでしょうか。

まずドキュメントが綺麗である事は明白です。ただのテキストよりはこちらの方が読みやすいです。次にコメントが残るという点について、詳しくは後述しますが、実はこの Sphinx で生成する為のドキュメントの情報はプログラムにコメントとして残すのです。

^{*3} <https://www.sphinx-doc.org/ja/master/>

第9章 現場で使える Python 自動化入門

またプログラムにコメントとして残されている為に、テキストとして残ることもメリットとなると思います。エンジニアとしてはデータとなるべく環境に依存しない方法で記載をしたいものです。この点においてプログラムのコメントとして残っていれば安心です。

さらに、この Sphinx では検索機能もついてきます。ユーザーフレンドリーな UI があれば他の方にも使ってもらいやすいです。最後にレイアウトを統一しやすいという点も上げられます。いろんな方がテキストでドキュメントを作成するとそのレイアウトは多種多様になりその読みやすさも書き手によってまちまちになります。

最終的にドキュメントの質はその内容に依存しますが、レイアウトが整えられている事で読み手はコンテキストを把握しやすくなるためレイアウトを統一しておく方がベターだと思います。

9.3.2 Sphinx 入門

Sphinx のドキュメント生成方法の基本的なステップは以下のとおりです。

1. コメントを記載する
2. rst ファイルを生成する
3. ドキュメントを生成する

ここからは、実際に Sphinx でドキュメントを生成する流れをご紹介します。

Sphinx はまずコメントを書きます。今回は Google の Docstring 形式^{*4}で記述します。このほか、numpy 形式などもあるので好みのスタイルを使いましょう。

< test.py(Google Docstring 形式コメント入り) >

```
class test:  
    """  
    testメソッド  
    """  
    def __init__(self, n):  
        self.number = n  
  
    def add(self, a):  
        """  
        内部変数nにaを足す関数  
  
        Args:  
            a(int): 内部変数nにaを足す  
  
        Returns:  
    """
```

^{*4} <https://google.github.io/styleguide/pyguide.html> 3.8.2 Modules

9.3 みんなで読もう Sphinx

```
int : 足し算後の内部変数
"""
self.number += a
return self.number

def out(self):
    """
    内部変数を出力するだけの関数

    Returns:
        None
    """
    print(self.number)
```

次にコメントから reStructuredText (rst ファイル) を作成後、ドキュメントを生成します。環境依存のある config の記述など詳細部分は省きましたが、ほぼこれだけで以下のようなドキュメントが生成されます。



図 9.3: ドキュメント例

また、Sphinx では html テーマがいくつもある為、自分好みに様々なカスタマイズを行なう事ができます。以下でいくつかのテーマを紹介いたします。(個人的には Open3D でも使用されているサードパーティー製の Read the Docs が好みです。)

第9章 現場で使える Python 自動化入門



図 9.4: classic

Welcome to test's documentation!

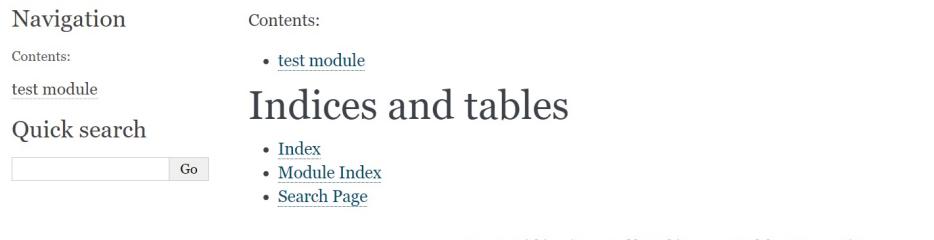


図 9.5: Alabaster

CONTENTS: / Welcome to test's documentation! View page source

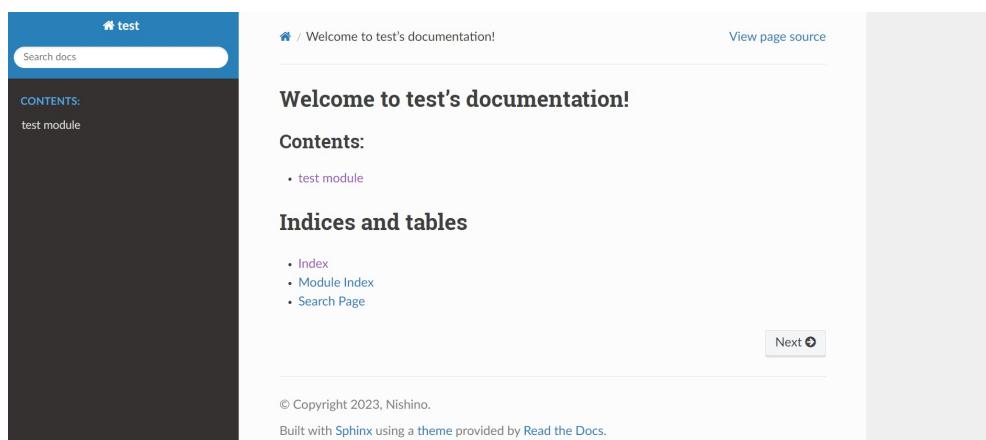


図 9.6: Read the Docs

9.3 みんなで読もう Sphinx

9.3.3 型アノテーションのすすめ

さて少し Sphinx とは離れますぐ、コードに仕様を記載するという点で型アノテーションについてご紹介したいと思います。型アノテーションは Python3.5 から追加された機能で Python の引数の型と返り値の型を記述する方法です。あくまでもアノテーションの為 Python 自体の挙動には関係がありません。以下にアノテーションの例を示します。

例えば add メソッドには引数の隣に int 型と記載してあり、返り値の型は” -> ”で記述されています。詳しくは Python.org のドキュメントを参照ください。^{*5}

```
< test.py >
```

```
class test:
    def __init__(self, n:int):
        self.number = n

    def add(self, a:int) -> int:
        self.number += a
        return self.number

    def out(self) -> None:
        print(self.number)
```

型アノテーションは Python で定義されている為、通常のコメントとは異なり IDE などを使うと自動で読み取ってくれる場合もあります。例えば Visual Studio Code で add 関数を呼び出すと以下のように型ヒントを自動で示してくれます。

^{*5} <https://docs.python.org/ja/3/library/typing.html>

第9章 現場で使える Python 自動化入門

```
1  from test import test
2
3  def main():
4      n=10
5      test_instance = test()
6      test_instance.out((a: int) -> int)
7      test_instance.add()
8      test_instance.out()
9  if __name__ == '__main__':
10     main()
```

図 9.7: Visual Studio Code による型ヒント

その他サードパーティーライブリの mypy^{*6}などを使用すると、型ヒントから型チェックを行なう事ができます。例えば以下のように、int で定義した変数に対して文字列を代入するようなコードを書いたとします。ここで実行すると後から代入した文字列が表示されます。

しかし、mypy を実行すると以下のようにエラーが出力されます。エラー文を読むと 2 行目で定義されている型と一致しないという事のようです。C 言語等のコンパイルチェックのようですね。これにより事前の型チェックを行なう事ができるようになり、型誤りによるバグの混入を未然に防ぐことができます。

< mypy_usecase.py >

```
def main():
    a: int = 3
    a = "bug"
    print(a)

if __name__ == '__main__':
    main()
```

<実行結果>

^{*6} <https://github.com/python/mypy>

9.4 最後の救い log 取得

bug

<mypy 実行結果>

```
(huga)hoge@hogehoge $ mypy mypy_usecase.py
mypy_usecase.py:2: error: Incompatible types in assignment (expr→
expression has type "str", variable has type "int") [assignment]
Found 1 error in 1 file (checked 1 source file)
```

9.3.4 現場での活用方法

現場で使用する場合には、生成されたドキュメント群を共有ディレクトリに置いています。エクスプローラーでパスがたどれるのであればドキュメント用に新規でサーバーを建てずともブラウザから確認することが可能です。

9.4 最後の救い log 取得

9.4.1 はじめに

みなさんは log を残しているでしょうか？先ほどの「2.1 はじめに」で述べたように私はなるべく多くの方に使ってもらう事でツールは真価を発揮し続けるという考えをもっています。沢山の方が使えば使うほど想定外の問題が起きる可能性は上がります。仕様による動作から想定しないバグまでできる事であれば全て網羅したいところですが、ツール作成時には必要以上の仕様を入れ込むことはコストパフォーマンスの低下を招きます。また、必要でない機能は仕様を決める時の解像度が低い為、良い仕様にすることは難しいものです。

そこでツール使用者に適切な情報を与えるために log 機能が必要になってきます。本節では、ツールの使用者に適切な情報を与えるための log についてお話ししたいと思います。

9.4.2 log を設定しよう

まず、前提としてどのような log が必要となるでしょうか？これは目的によって様々となります。例えばレポートログを作成するツールであれば形式は人が読みやすい書式が

第9章 現場で使える Python 自動化入門

良いでしょう。今回はツールのデバッグ用のログとして考えてみましょう。

ログにはロギングレベルというものがあります。例えば Python の logging モジュールでは以下の 6 つが定義されています。^{*7}

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG
- NOTSET

なぜこんなにも分れているかというと、常に出してほしい情報と必要な時に出してほしい情報のレベルは異なる為です。

例えば、セキュリティソフトをイメージしてみましょう。デバッグ用に必要な情報はなるべく多く出したいのですが、常に今どのファイルを検査していて、どのようなアップデートファイルをインストールしていて、バックグラウンドでどのシステムが動いているといった通知が常に出ていると必要な情報が埋もれてしまい逆に判りづらくなってしまいます。

一方で、攻撃を受けた場合は即座に通知をしてほしいものです。このように、その必要性に応じて適切にロギングレベルを適切に管理する必要があります。これだけの種類が用意されているとはいえ私が現場で使う際は CRITICAL、WARNING、DEBUG 程度でしか使い分けをしていません。理由としては Python で作成するツールの多くはそこまで規模が大きくならない為です。粒度は必要に応じて定めてよいと思います。

Python であれば、logging モジュールで必要なレベルを設定しておけば import 先で logging モジュールを使っていれば log に情報が吐き出されます。適切に設定しておくことでユーザーフレンドリーなツールとなるでしょう。また使用者側で確認ができればツールから手を離すことができます。現場ではツール開発者がメンテナンスを行なうことが多いと思います。ログを残してなるべく使用者側の手で解決してもらえるようにしておくと良いと思います。

9.4.3 DEBUG は DEBUG

DEBUG ログはバグ調査の最後の手段としてあるべきだと考えています。そのため、なるべく DEBUG では様々な情報があったほうが良いのですが、それでも必要な情報は精査すべきです。

^{*7} <https://docs.python.org/ja/3/library/logging.html?highlight=log>

9.5 さいごに

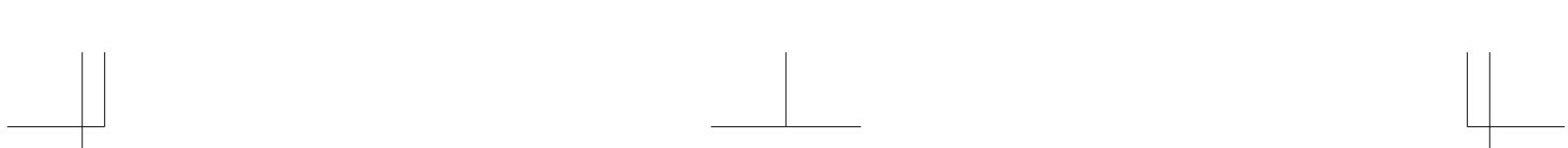
これは、私自身の体験なのですが何度も呼び出されるモジュールに対して細かく DEBUG を吐き出すように設定していました。何度かテストを行なってログを確認しようとするとテキストエディタでは開けなくなってしまいました。これは、あまりにもログを細かく設定しすぎたために一度の呼び出しで沢山のログが吐き出され（一度で約 500MB 程度）いつの間にかログが肥大化していたためです。DEBUG の設定はデバッグの為に必要な情報を精査すべきでした。

この例は、あくまでも私の場合であり状況に応じてログを残すべきですが、サーバー上に置くものであればログがディスクの圧迫を招く可能性があるという事も考慮に入るべきでしょう。また事前にログの量をある程度見積もることも重要なと思います。

9.5 さいごに

本章の裏テーマは「開発者の負担を減らす」というものでした。Python では様々なモジュールがでており、また書籍やテックブログも潤沢である事でツールを開発しやすくなっていると思います。

一方で現場ではその多くが開発者自身で運用やメンテナンスにも時間をさく必要があり、ツールを開発すればするだけメンテナンスコストが高まるという負のサイクルに課題を感じています。そのため本章の 2/3 がツール利用者に自力で解決してもらう為の同線を敷く為の Tips となっています。ご紹介した内容が少しでも有益なものとなれば幸いです。本章をお読みいただきありがとうございました。



第 10 章

GAS と AWS を使って Web 操作 をハックしてみた!

大野 泰歩

10.1 はじめに

はじめて！ 筆者は普段通信事業者として商用設備の運用保守業務を担当しております。運用保守業務を実施されたことのある方であれば経験があるかと思いますが、保守しているシステムの不具合や故障（これらを以下、インシデントと記載）の対処を日々実施していく必要があります。この業務を滞りなく遂行していくには情報の扱いを出来る限り自動にしていく事が望ましいです。

私は Google Apps Script^{*1}(GAS) や Amazon Web Services^{*2}(AWS) を用いて Web ページの操作を自動化の部分を実施しましたが、こちらをご説明する前に既存で作成されていた情報収集の自動化部分からも記載していこうかと思います。

GAS や AWS などの技術については独学で学んだモノになりますので、拙い部分も多々あるかと思いますが、あくまで自動化の一例として温かい目で拝読頂けると幸いです。

10.2 情報収集の自動化 (GAS)

「はじめに」で少し触れたように、私は運用保守業務を実施するうえで様々なインシデントを管理しており、こちらを管理するのにあたりスプレッドシートを用いております。毎度手動で入力していくのは大変なため、Gmail で通知されるインシデントから必要な情

^{*1} <https://developers.google.com/apps-script?hl=ja>

^{*2} https://docs.aws.amazon.com/ja_jp/

第10章 GASとAWSを使ってWeb操作をハックしてみた!

報を抜き出し、スプレッドシートに転記されるようにGASを用いて自動化しています。

自動化の方法としては通知されるインシデントメールのテンプレートを定義することでGASを用いて必要項目を抜き出しへスプレッドシートに転記するというものになります。

メールの例文を2種類程紹介します。

- ・メールサンプル1

```
■ID  
1111111  
  
■ホストグループ名  
service_ABC  
  
■ホスト名  
hostname_ABC  
  
■アラームテキスト  
• Error message
```

- ・メールサンプル2

```
[Linuxtime] 1111111  
[Hostname] hostname_ABC  
[Serial] abc123
```

ご紹介した様にテンプレートとしてメール本文の内容を決めることが出来た場合は、GASを用いてメール本文から情報を抜き出し、スプレッドシートに転記することが出来ます。

今回は正規表現を用いて情報を抜き出す際の例を記載いたします。

- ・メール本文の抜き出し(GAS)

```
var myThreads = GmailApp.search('メール件名', 0, 50);  
var myMsgs = GmailApp.getMessagesForThreads(myThreads);  
  
for ( var threadIndex = 0 ; threadIndex < myThreads.length ; threadIndex++ ) {  
    var mailBody = myMsgs[threadIndex][0].getPlainBody();
```

10.3 Web 操作の自動化 (AWS)

- メールサンプル 1_抜き出し方法例 (GAS)

```
var myRegexp = new RegExp('■ID' + '[\\s\\S]*?' + '■');
if (mailBody.match(myRegexp)) {
    var incidentID = mailBody.match(myRegexp)[0].split('\\n')[→
1];
}
else {
    continue;
}
```

- メールサンプル 2_抜き出し方法例 (GAS)

```
var myRegexp = new RegExp('\\[Linuxtime\\]' + '.*?' + '\\r');
if (mailBody.match(myRegexp)) {
    var incidentID = mailBody.match(myRegexp)[0].replace('[Linux→
time]', '');
}
else {
    incidentID = "Linuxtime未記載";
}
```

このようにメール本文から情報を変数として取得することができましたので、後はスプレッドシートの中で転記したい位置に取得した情報を記載するようにすれば情報の取得について自動化することが出来ました。

その他に補足情報が必要な場合は、他のスプレッドシートに記載している情報を同じ様に GAS を用いたり関数を使用することで更に多くの情報を集めることもできます。

10.3 Web 操作の自動化 (AWS)

情報の取得について自動にすることが出来たので、集めた情報を元にどの様に Web 操作を自動にしていくかを記載いたします。

元々は GAS を用いて Web 操作を実施しようと思っておりましたが、GAS のみで Web 操作を可能とするドキュメントを見つけることが出来なかったため、AWS を用いて自動化していく事にしました。

またこの後説明していく処理の流れがイメージしやすいように概要を添付いたします。

第 10 章 GAS と AWS を使って Web 操作をハックしてみた!

概要図

GASで必要な情報を「API Gateway」に渡し、
その後「Lambda」で処理の実行（pythonでselenium）

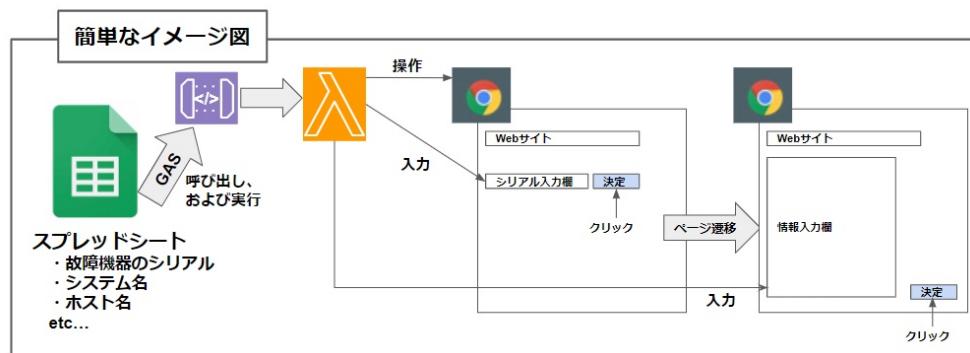


図 10.1: 処理概要

添付した「処理概要」の通りになりますが、下記の流れで処理を行っていきます。

1. スプレッドシートに情報収集
2. GAS から API Gateway に Lambda の呼び出し処理
3. Lambda にて Web 画面の自動操作
4. 処理結果を GAS に返信

まずは Lambda での処理を記載していきます。今回 Lambda では Python を使用していきます。Python を選択した理由としては Web 画面の操作を実施するのに有名な Selenium というモジュールを使用するためとなります。

自動化をするにあたって、このタイミングで 1 度躊躇しました。躊躇した理由としては Lambda の中には標準的な関数しか搭載されておらず Selenium などの拡張モジュールを使用することが出来なかったからとなります。そのため拡張モジュールを使用するために Lambda の中のレイヤーという機能として使用したいモジュールの追加を行いツールを実行出来るようにしました。

今回追加したモジュールは 4 つになります。

- Chromedriver: Selenium の処理を行う土台のサイトとして使用するため
- headless-chromium: Web 画面を表示しない状態で使用するため
- Selenium: Web 画面操作の自動化を行うため
- headless-selenium: ヘッドレスモードで Web 画面操作を行うため

10.3 Web 操作の自動化 (AWS)



名前	バージョン	互換性のあるランタイム	互換性のあるアーキテクチャ
chromedriver	2	python3.7	-
headless	1	python3.7	-
headless-chromium	2	python3.7	-
selenium	2	python3.7	-

図 10.2: レイヤー情報

ここまで準備出来たら後は python を用いて Selenium の処理を記載することで Web 画面の操作が出来るようになります。

- import 内容 (Lambda)

```
import json
import time
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.select import Select
```

- API Gateway と連携するための関数 (Lambda)

```
def lambda_handler(event, context):
    # TODO implement
    body = json.loads(event['body'])
    sample_value = sample_function(body)

    return {
        'statusCode': 200,
        'body': json.dumps(sample_value)
    }
```

- Chrome ドライバの設定 (Lambda)

第10章 GASとAWSを使ってWeb操作をハックしてみた!

```
def headless_chrome():
    ## Seleniumが使用するWebDriver作成時のオプションを作る
    options = webdriver.ChromeOptions()
    ## オプションのバイナリロケーションにLayerで用意したheadless-chromiumのパスを指定
    options.binary_location = '/opt/headless/bin/headless-chromium'
    ## オプションにヘッドレスモードで実行させる記述を書く
    options.add_argument("--headless")
    options.add_argument("--no-sandbox")
    options.add_argument("--single-process")
    options.add_argument("--disable-gpu")
    options.add_argument("--homedir=/tmp")

    driver = webdriver.Chrome(
        ## Layerで用意したchromedriverのパスを指定
        executable_path="/opt/headless/bin/chromedriver",
        options = options
    )
    return driver
```

- Selenium 使用例 (Lambda)

```
def sample_function(body):
    driver = headless_chrome()
    driver.delete_all_cookies()
    driver.get('https://www.google.co.jp/')
    search_box = driver.find_element_by_xpath('//*[@id="APjFqb"]')
    search_box.send_keys(body['sample'])
```

この4つの内容を記載することでプログラムの基本的な概要は完成しております。後は、実際に使用する場合に合わせて必要なオプションを追加したり Selenium の処理を追記していく事で実行したい処理を完成させていく事が出来ます。

Selenium の使い方や関数の詳細については色々と書籍や Web ページでの説明があるので、今回の自動化をする際に良く使用していた関数を代表として紹介します。

- find_element_by_xpath: Web ページの xpath を指定して要素を取得する
- send_keys: テキストボックスなどに文字列を入力する
- select_by_value: ドロップダウンなどのセレクト要素を選択する
- click: 選択している要素をクリックする

10.4 GAS から Lambda の呼び出し (GAS)

これにて Web 画面の操作を自動にする処理を Lambda を用いて作成することが出来ました。そのため後は API Gateway を作成し、今回作った Lambda と連携させることで GAS から呼び出せるようになります。

10.4 GAS から Lambda の呼び出し (GAS)

GAS から Lambda に連携するにあたって API Gateway で生成した連携用の URL と収集した情報を json の形で整えて使用する必要があります。その整えた情報を UrlFetchApp.fetch を使用することで Lambda に渡すことが出来ます。

- GAS から Lambda への連携 (GAS)

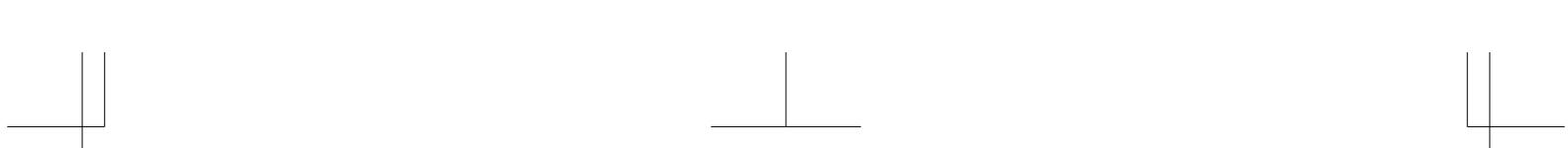
```
const aws_url = 'API GatewayのURL';
const params = {
  'method' : 'post', //get or post
  'contentType': 'application/json',
  'payload' : JSON.stringify(
    {"sample_value1" : sample_value1,
     "sample_value2" : sample_value2,
     "sample_value3" : sample_value3})
};

const req = UrlFetchApp.fetch(aws_url, params);
```

これにて GAS と Lambda の連携が完了しましたので、後は処理が完了した旨のポップアップを表示させたりメールを送るなどお好みで設定するかたちとなります。

10.5 さいごに

本章においては、定常的な業務の効率化を行うための方法について記載させて頂きました。AWS の使い方や GAS の使用方法について他にも最適な方法はあるかと思いますが、この章をお読みいただいた方の業務が少しでも楽になったり、効率化への閃きに寄与することができますれば幸いです。最後になりますが本章をお読みいただきましてありがとうございました。



第 11 章

フローで思い通りの結果を得るために大切なこと

高井 美佑

11.1 はじめに

みなさま、ごきげんよう。突然ですが、Power Automate でフローを組んで、テストして、無事に成功した時、安心してそのままウィンドウを閉じていませんか？

「フローが成功しました」という表示は「エラーなしで正しく実行されました」という意味で、「作成者であるあなたの想定通りに実行されました」という意味ではありませんよ？

さて、あなたが今閉じてしまったそのフロー、本当に「想定通り」の動きをしているでしょうか？ 不安になっていませんか？ 本章では、フローが「想定通り」に動いているかどうかの確認方法を実際のユースケースを見ながら、解説していきます。ぜひ最後まで一読頂き、想定通りの処理が行われているかどうかの確認方法を身に着けて頂きたいと思います。フローの処理に不備があったがために、偉い人に「手作業憐みの令」を出されないように。

11.2 テストは、2段階に分けられる

IT を本職とされている人よりも知識が薄い市民開発者が忘れがちなのは、テストは次の 2 段階に分けられるということです。

1. テスト機能を使用したフローの動作テスト
2. 作成者が期待する「正常な動作」が行われているかの確認

1. は、ほとんどの人が当たり前のように行っていることだと思います。問題は 2。作成者

第 11 章 フローで思い通りの結果を得るために大切なこと

が期待する「正常な動作」が行われているかの確認です。これは、フローのテスト結果の中身を確認したり、フローの外に出力された結果 (EX: フローで投稿した Teams のメッセージ) を確認したりすることが必要です。

フローが成功していても、作成者が期待する「正常な動作」をしていないこともよくあります。市民開発者は、Power Platform での開発が初めての開発経験という方も多く、1. のテスト機能を使用したフローの動作テストが成功すると安心してテストをやめてしまうこともあるのです (かつての私がそうでした) そのため、実際のユースケースを通して、2. の確認方法を学んでいきたいと思います。

11.3 Case アップロードしたはずのファイルが開かない

11.3 Case アップロードしたはずのファイルが開かない

11.3.1 どんなフローか

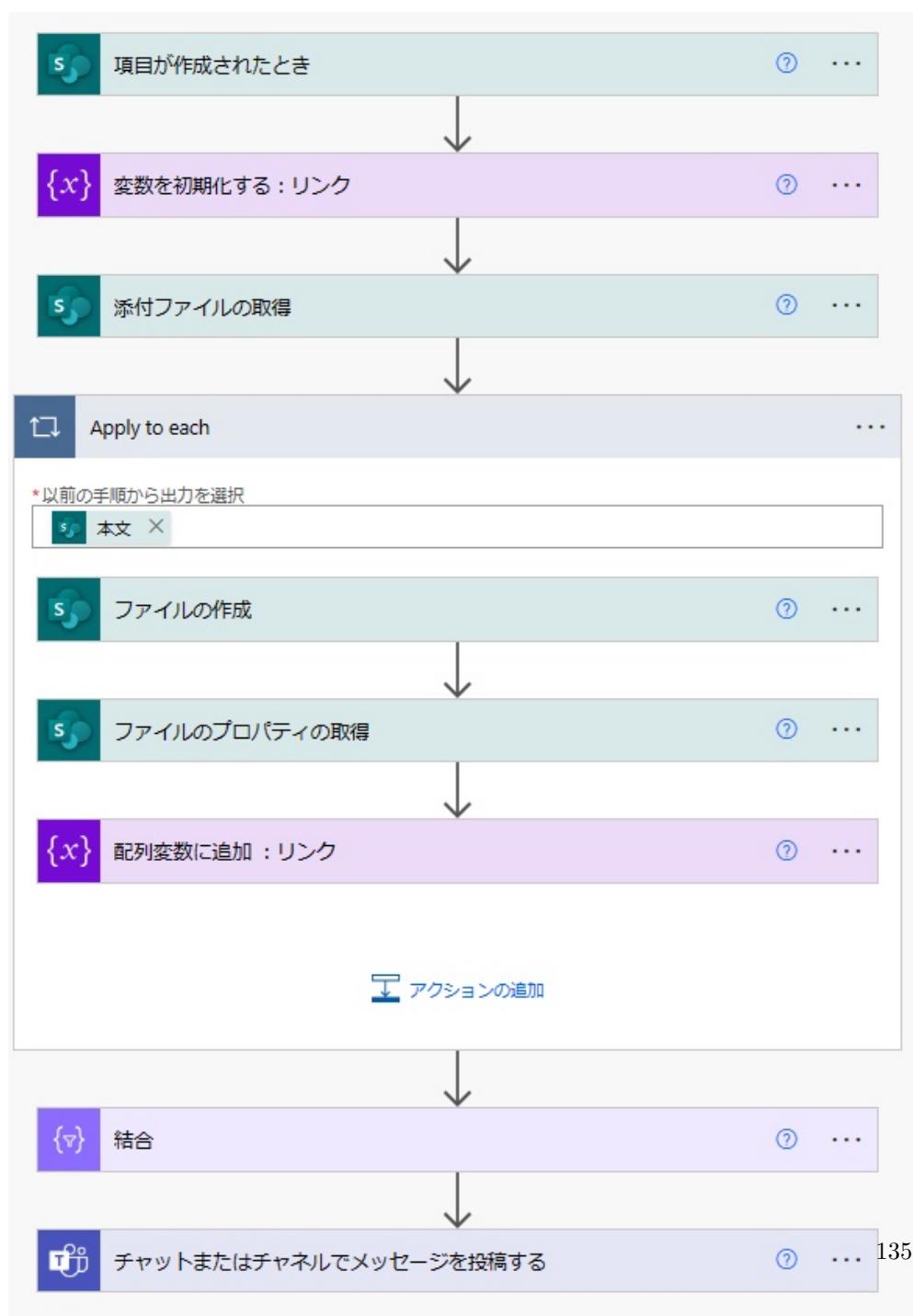


図 11.1: どんなフローか

第 11 章 フローで思い通りの結果を得るために大切なこと

このフローは、指定された SharePoint リストにレコードが作成された時にそのレコードについている添付ファイルを取得し、その添付ファイルを指定されたドキュメントライブラリにアップロードし、Teams にドキュメントライブラリへのリンク付きメッセージを投稿するという動作をします。

11.3.2 このフローで期待される正常な動作はどういうものか

- 新しいレコードが作成された時にフローが実行されること



図 11.2: 新しいレコードが作成された時にフローが実行される

- Teams にファイルのリンク付きメッセージが投稿されること

11.3 Case アップロードしたはずのファイルが開かない

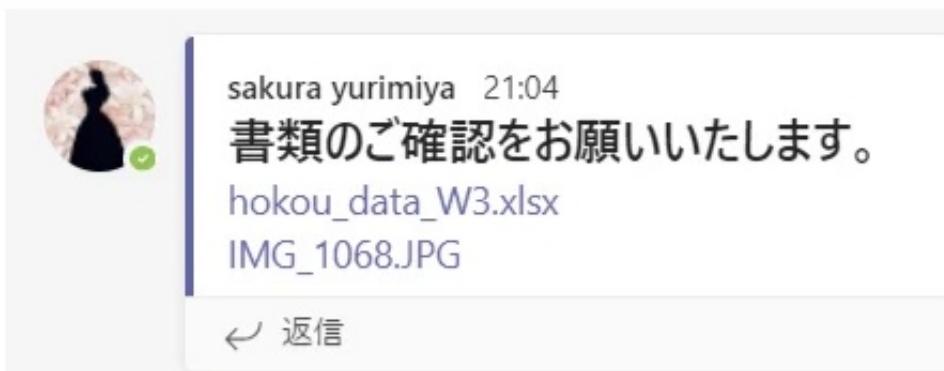


図 11.3: Teams にファイルのリンク付きメッセージが投稿

- メッセージのリンクをクリックしたら、ファイルが表示されること

以上 3 点が作成者である私が期待する正常な動作です。

11.3.3 テスト機能を使って、フローの動作テストを行う

フローを作成したら、まずはテスト機能を使って、動作テストを行います。動作テストは、画面右上の「テスト」ボタンをクリックすると行うことができます。

第 11 章 フローで思い通りの結果を得るために大切なこと

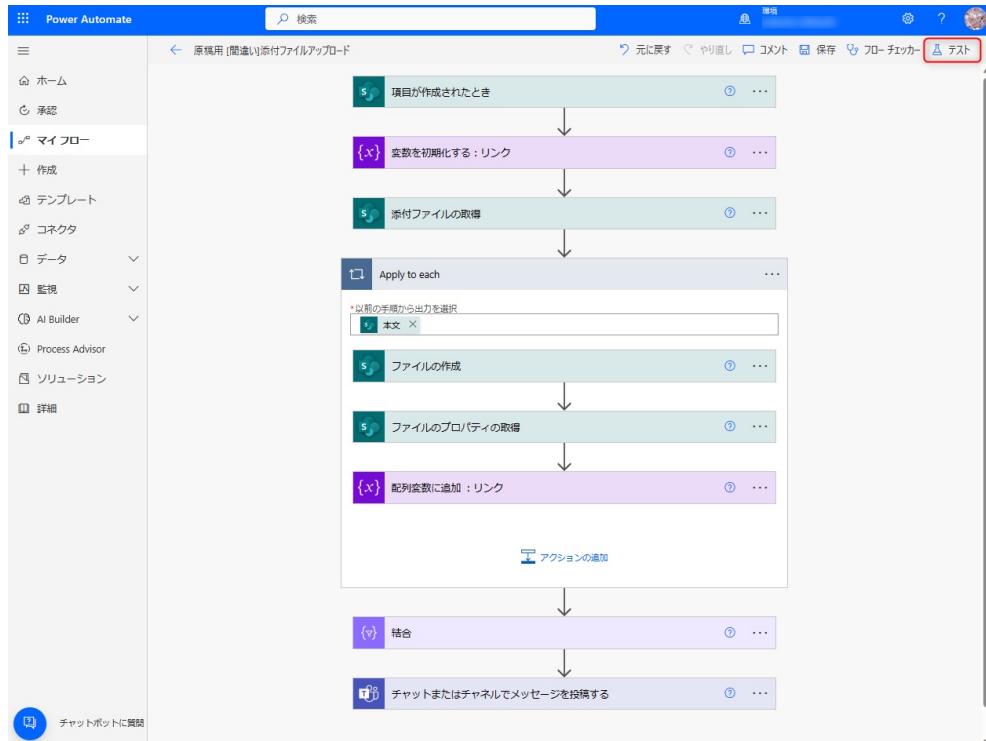


図 11.4: テスト機能 1

11.3 Case アップロードしたはずのファイルが開かない

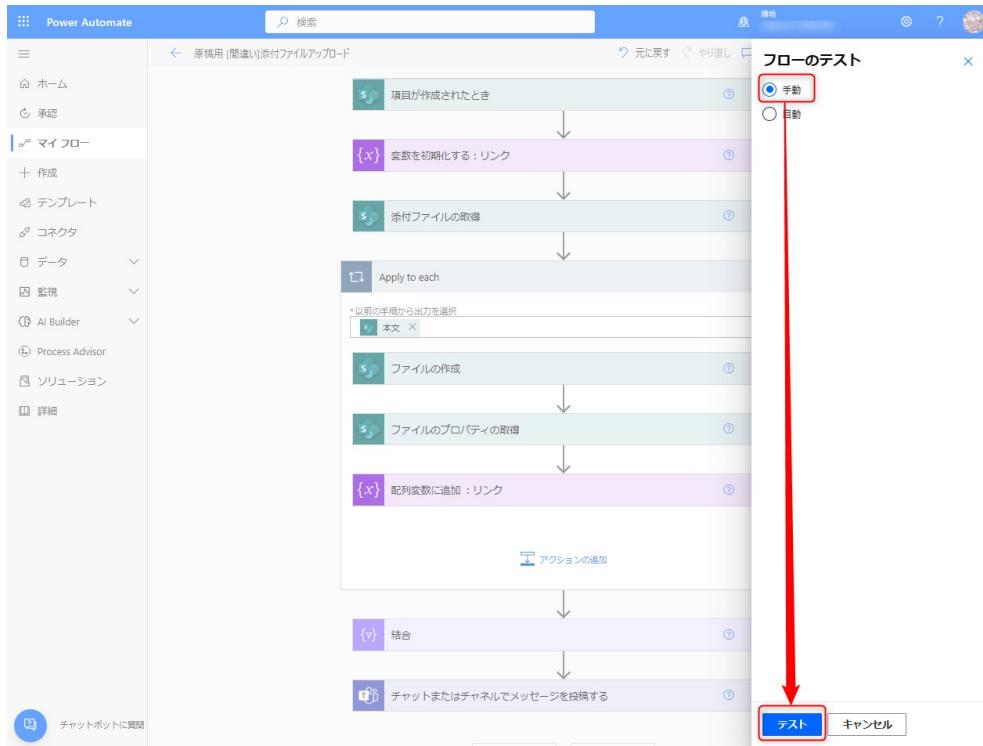


図 11.5: テスト機能 2

最初のテストは「手動」から行います。今回のトリガーは、SharePoint リスト のレコードの新規作成なので、テストボタンを押した後にレコードの新規作成を行ってください。

第 11 章 フローで思い通りの結果を得るために大切なこと

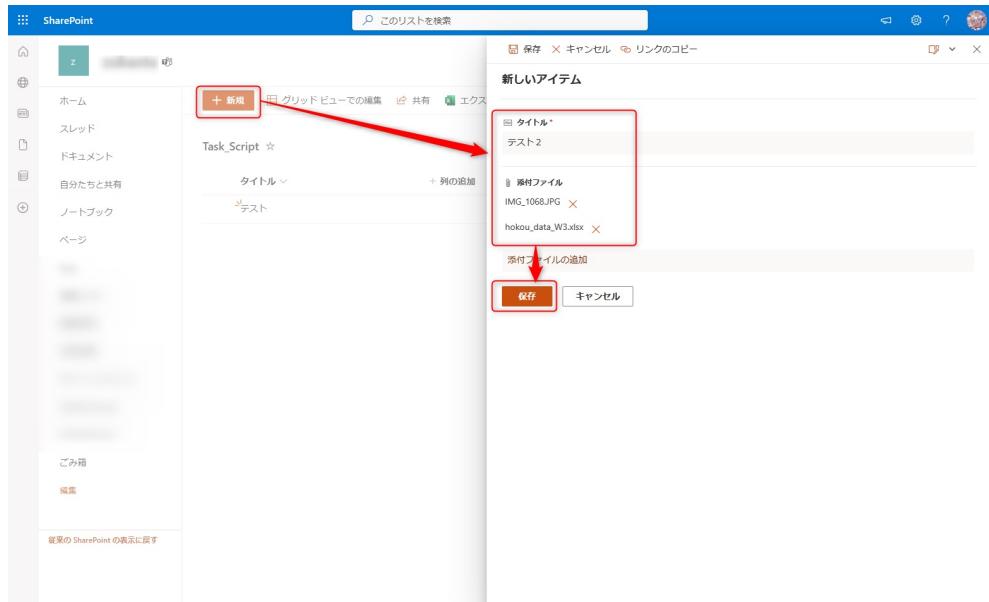


図 11.6: SharePoint のレコードの新規作成

そうすることでフローが実行されます。テストが成功すると、緑色で「ご利用のフローが正常に実行されました」という通知が出てきます。

11.3 Case アップロードしたはずのファイルが開かない



図 11.7: フローの実行

ちなみに 2 回目以降は、1 回目のデータを利用して、「自動」からテストを行うこともできます。

第 11 章 フローで思い通りの結果を得るために大切なこと

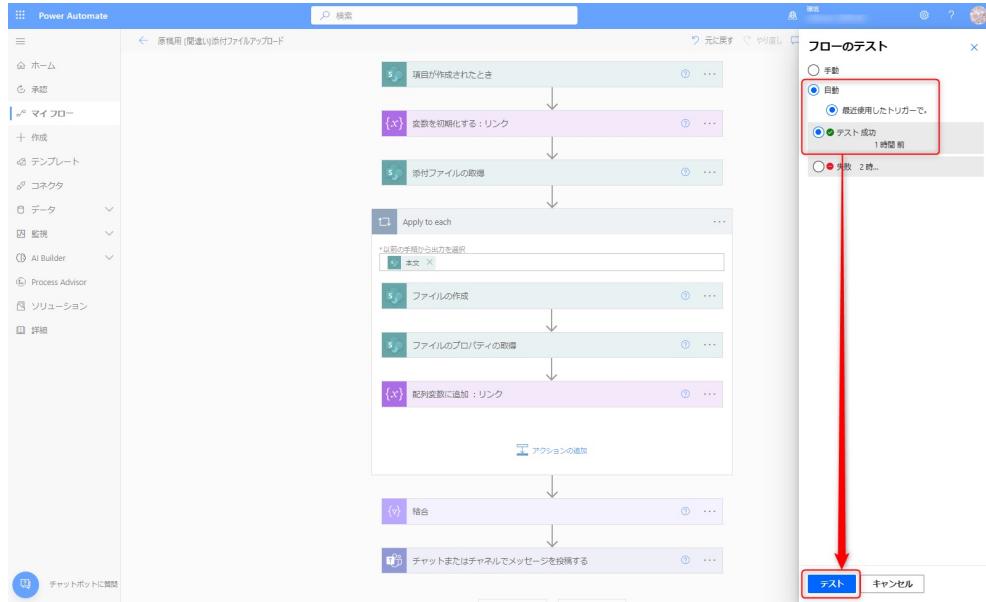


図 11.8: 自動からテストを実行

11.3.4 出力結果が正常な動作をするか確認する

フローが成功し、無事 Teams にメッセージが投稿されたので、リンクをクリックして、ファイルを確認してみます。

11.3 Case アップロードしたはずのファイルが開かない

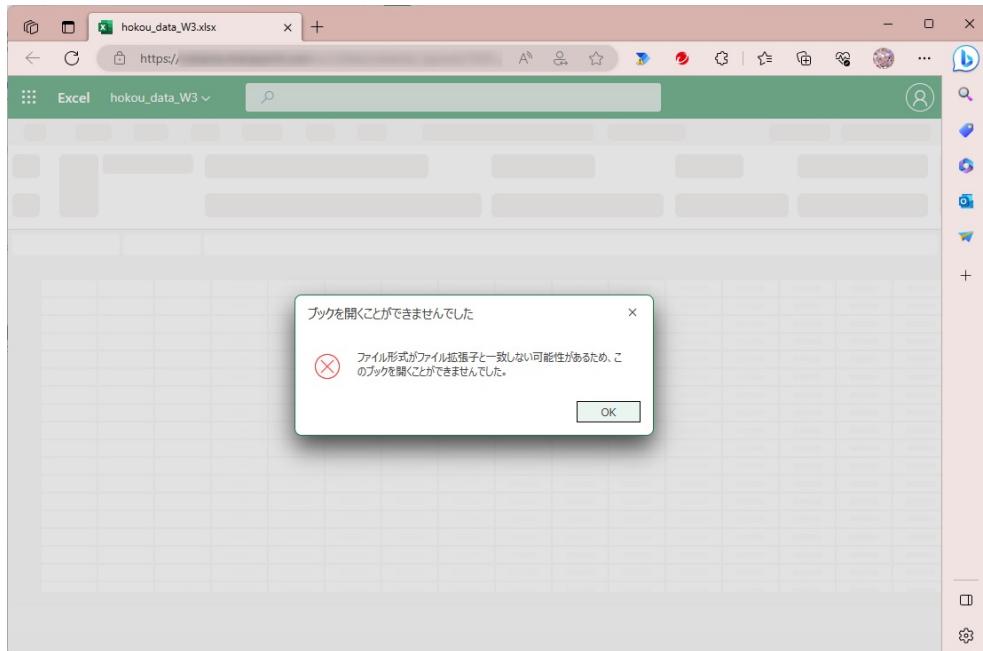


図 11.9: 出力結果

すると、なんということでしょう！！

「ブックを開くことができませんでした」というメッセージが表示されています。これはExcel固有の現象なのか、それとも画像など他のファイルでも同じようになるのか調べるために、画像のリンクもクリックしてみます。

第 11 章 フローで思い通りの結果を得るために大切なこと

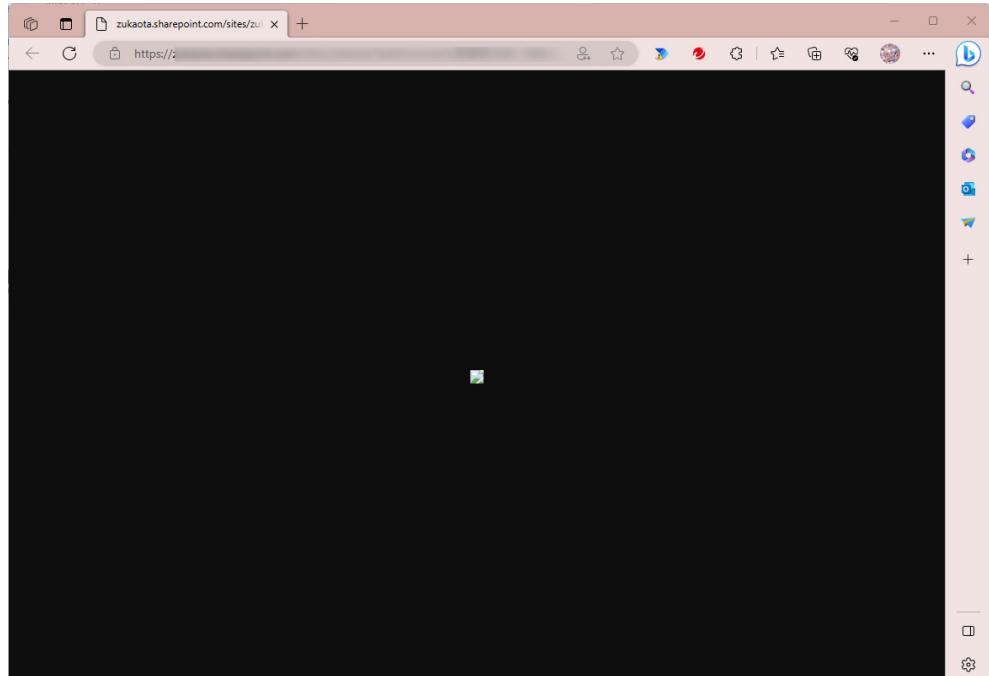


図 11.10: 画像の表示

すると、画像も開くことができませんでした。ものの見事に真っ暗です。つまり、「なんらかの事情でメッセージにあるリンクが開けない」というバグが見つかりました。

11.3.5 「心当たり」をすべて確認し、問題を切り分ける

バグの修正は、無くしたものを探すことと似ている気がします。例えば財布を落とした時、どうするでしょう？ 最後にどこで使ったかを考え、最後に使った場所から現在地までどのように行動したかを思い出して、財布を落としたかも？ と心当たりのある場所を 1 か所ずつ訪れて、探していきませんか？ バグの修正も同じです。「心当たり」を 1 か所ずつ確認していきます。私は、2 つの「心当たり」があります。

- 実はフローが失敗している（テスト結果は見間違いだった）
- 添付ファイルがドキュメントライブラリにアップロードされる過程で壊れた

11.3 Case アップロードしたはずのファイルが開かない

11.3.6 実行履歴を確認してみる

フローのテストを実行し、成功通知を確認しているとはいえ、バグが出ている以上は本当に成功しているのかの確認は必要です。実行履歴を確認してみましょう。フローの編集画面左上の「←」ボタンをクリックして、1つ前の画面に戻ります。



図 11.11: 1つ前の画面に戻る

「28 日間の実行履歴」から先ほどのテストの日時をクリックしましょう。

第 11 章 フローで思い通りの結果を得るために大切なこと

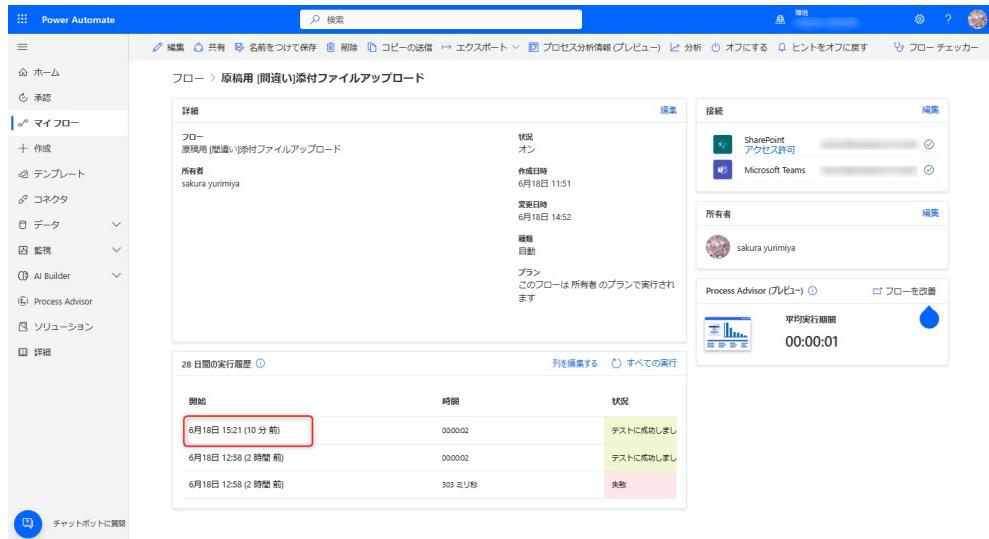


図 11.12: 28 日間の実行履歴

すべてのアクションに緑のチェックマークがついていて、画面上部に成功通知も出ています。間違いなく成功していることが確認できました。

11.3 Case アップロードしたはずのファイルが開かない



図 11.13: 正常に終了

11.3.7 ファイルサイズを確認してみる

フローが正しく動作していることがわかりました。それならば、なんらかの原因でファイルが壊れている説が有力になってきますね。添付ファイルと同じものがドキュメントライブラリに正しく保存されていれば、ファイルの大きさも全く同じなはず……。なので、元データとドキュメントライブラリ保存のファイルサイズを比較してみることにします。

第 11 章 フローで思い通りの結果を得るために大切なこと

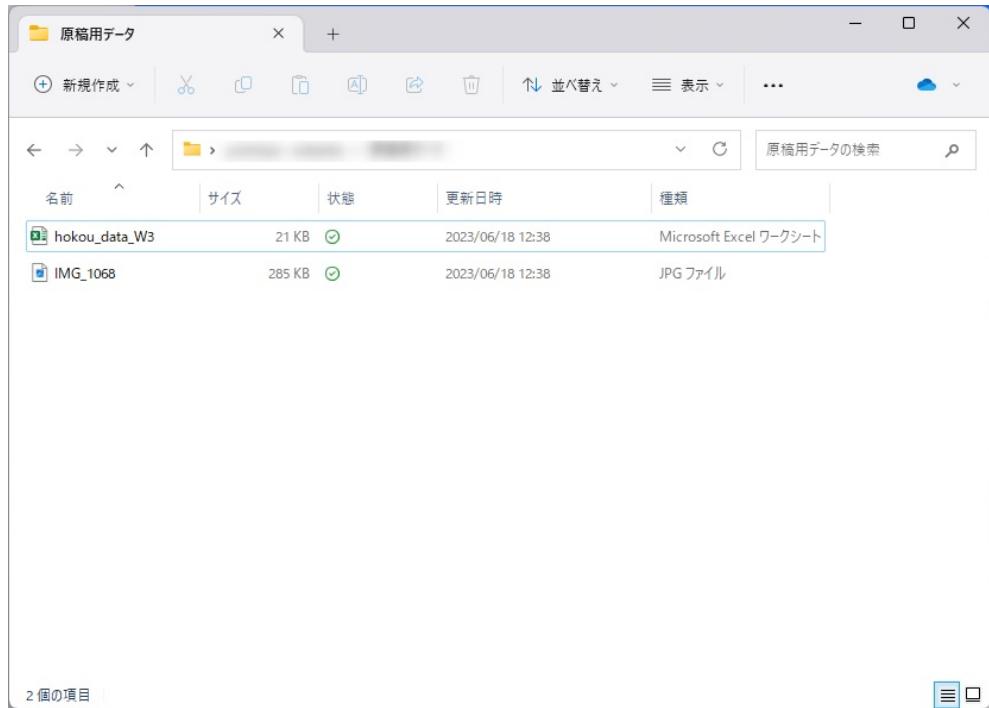


図 11.14: 元データ

TaskDocument > 営業部

名前	ファイル サイズ
hokou_data_W3.xlsx	71 バイト
IMG_1068.JPG	65 バイト

図 11.15: ドキュメントライブラリのデータ

比較してみると、明らかにサイズが異なります。元データは単位が「KB(キロバイト)」

11.3 Case アップロードしたはずのファイルが開かない

なのに対し、ドキュメントライブラリ側は「B(バイト)」です。元データよりもファイルが小さすぎることがわかりました。ここで思い出してほしいのは、ドキュメントライブラリのファイルは Power Automate のフローで作成しているということです。つまり、フローの動作テストは成功しているが、フローに何か不具合があるためにファイルが正しく作成されておらず、ファイルサイズが元データと異なる、という状態であることがわかります。

11.3.8 ファイルの作成アクションを確認してみる

フローの中でドキュメントライブラリに保存するファイルを作成しているのは「ファイルの作成」アクションです。このアクションの設定が正しいかを確認していきます。編集画面から見ると、ファイルの作成アクションの設定は以下のようになっています。

第 11 章 フローで思い通りの結果を得るために大切なこと

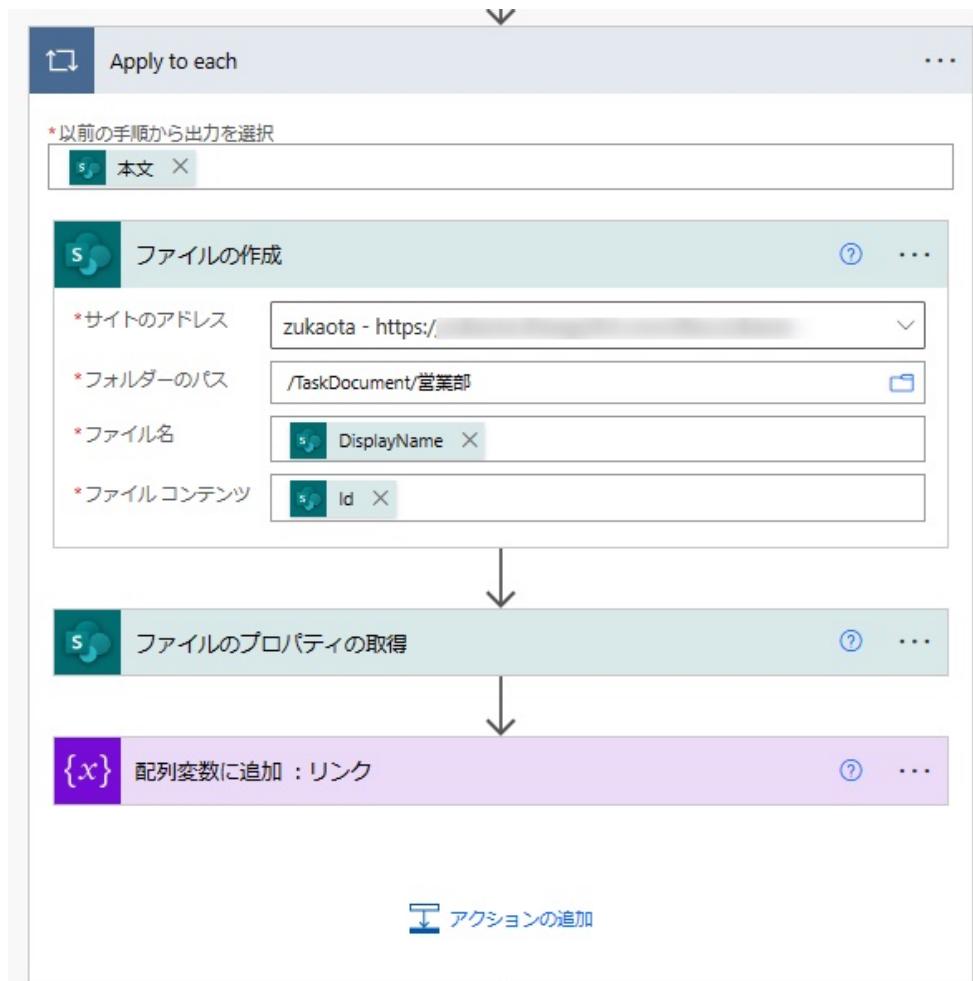


図 11.16: ファイルの作成 1

実行履歴からテスト時に入力された値を確認していきます。

11.3 Case アップロードしたはずのファイルが開かない

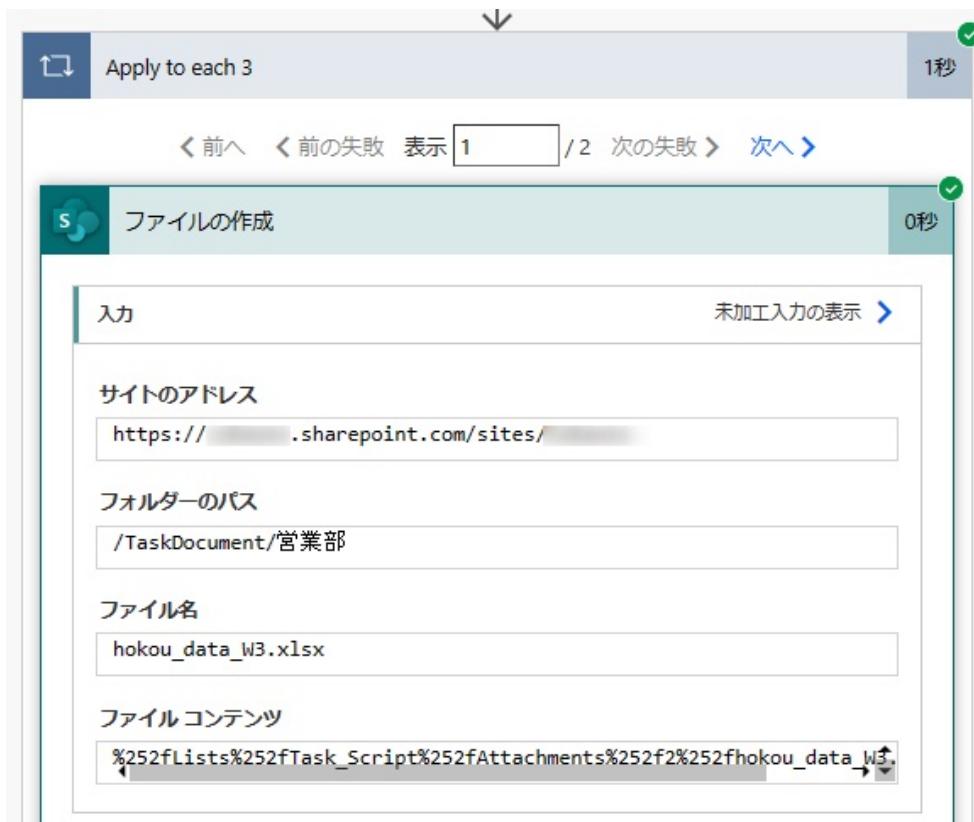


図 11.17: ファイルの作成 2

サイトのアドレス、フォルダーのパス、ファイル名は想定通りの値が入力されていることが確認できます。この 3 つは問題ないでしょう。でもファイルコンテンツは、不思議な並びの文字列で、この値が正しいものなのかどうか、現時点ではわかりません。改めてファイルの作成アクションとは何かを調べてみる必要がありそうです。

言葉の意味を調べる時は公式リファレンスをあたるのが 1 番です。今回の場合は、Power Automate のフローなので、Microsoft Learn^{*1} を読み解いていきます。

*1 <https://learn.microsoft.com/ja-jp/connectors/sharepointonline/>

ファイルの作成

操作 ID: CreateFile

SharePoint サイトにファイルをアップロードします。必ず既存のライブラリを選択してください。

パラメーター

Name	キー	必須	型	説明
サイトのアドレス	dataset	True	string	例: https://contoso.sharepoint.com/sites/sitename 。
フォルダのパス	folderPath	True	string	既存のライブラリから開始する必要があります。必要に応じてフォルダを追加します。
ファイル名	name	True	string	ファイルの名前。
ファイルコンテンツ	body	True	binary	ファイルのコンテンツ。

図 11.18: ファイルの作成 (出典:Microsoft Learn)

Microsoft Learn には、ファイルのコンテンツ欄には「ファイルのコンテンツ」を入れると説明されています。では、先ほど「ファイルのコンテンツ」欄に入っていた「Id」はファイルのコンテンツなのでしょうか？

11.3 Case アップロードしたはずのファイルが開かない



図 11.19: ファイルのコンテンツ欄

動的な値から改めて「Id」を確認したところ、「ファイルの識別子」であり、「ファイルのコンテンツ」ではないことがわかりました。つまり、ファイルの作成アクションの「ファイル コンテンツ」の設定を間違っていたため、ドキュメントライブラリに正しくファイルが作成されていないということがわかりました。

11.3.9 ファイル コンテンツ は存在するのか？

ファイルの作成アクションよりも前のアクションで「ファイルのコンテンツ」が取得できていれば、ファイルの作成アクションの「ファイル コンテンツ」に「ファイルのコンテンツ」を入れることができます。では、今のフローで「ファイルのコンテンツ」を取得できているのでしょうか？ 動的な値を検索して、確認してみましょう。

第 11 章 フローで思い通りの結果を得るために大切なこと



図 11.20: 動的な値 1

11.3 Case アップロードしたはずのファイルが開かない

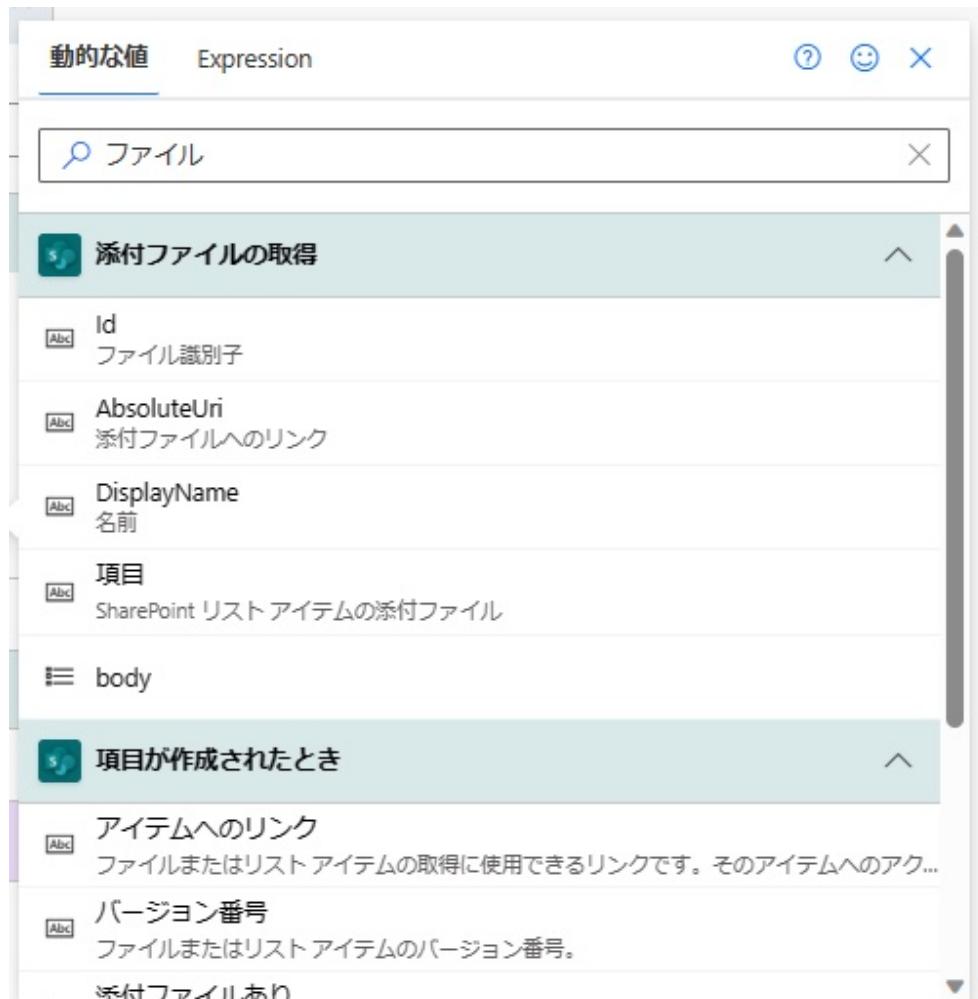


図 11.21: 動的な値 2

第 11 章 フローで思い通りの結果を得るために大切なこと

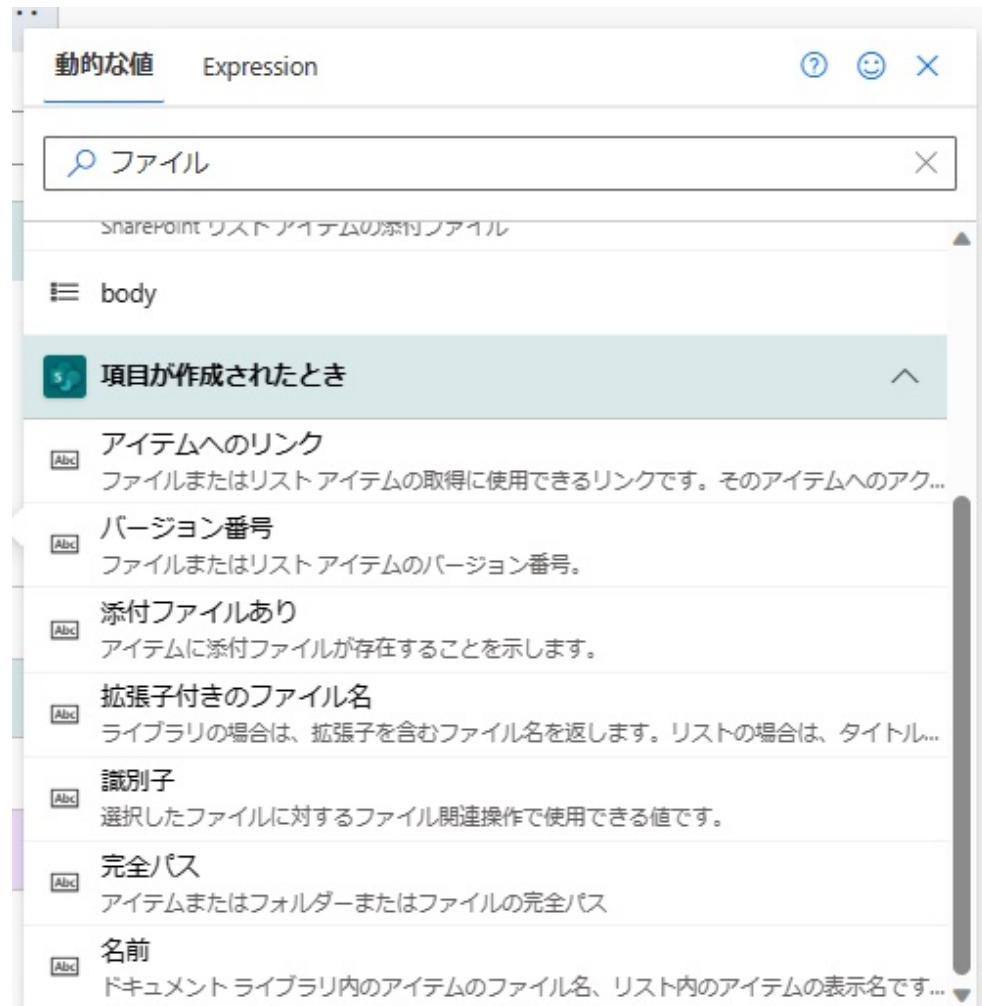


図 11.22: 動的な値 3

11.3 Case アップロードしたはずのファイルが開かない

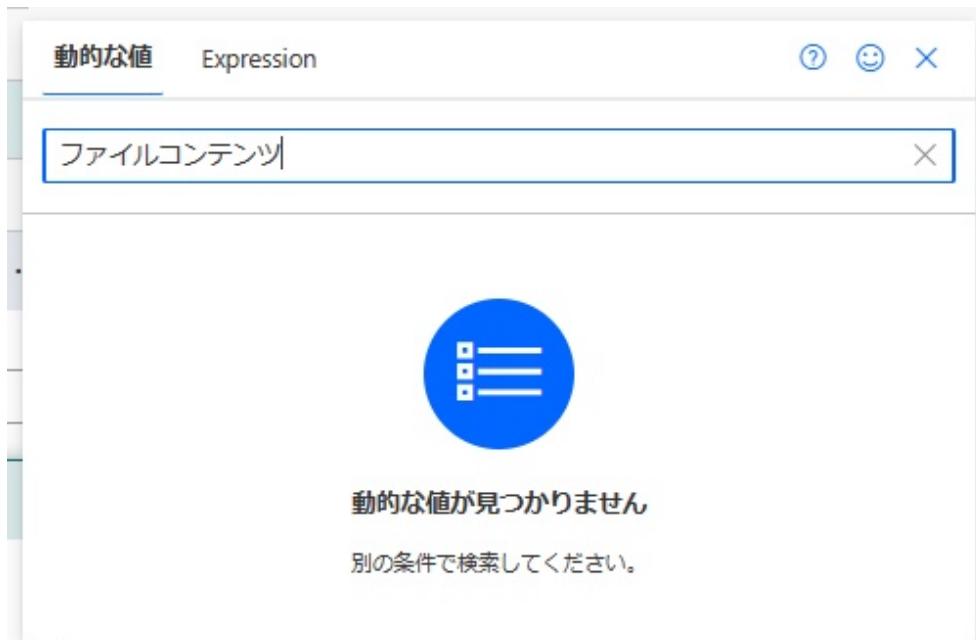


図 11.23: 動的な値 4

「ファイルの作成」アクションの「ファイル コンテンツ」欄にカーソルを当て、「コンテンツ」「ファイル」「ファイルコンテンツ」の 3 つで検索しましたが、「ファイルのコンテンツ」らしき動的な値は見つかりませんでした。つまり、新しくアクションを追加して、「ファイルのコンテンツ」を取得しなければならないということがわかりました。

11.3.10 どのアクションで「ファイルのコンテンツ」を取得するか

取得したいファイルのコンテンツは、レコードの添付ファイルのコンテンツです。ゆえにレコードの添付ファイルのコンテンツを取得するには、どうすればいいかを考える必要があります。わからないことがある時は、公式リファレンスに立ち返るのが 1 番です。今回も Microsoft Learn を読み解いていきたいと思います。

SharePoint コネクタのアクション一覧を見ていくと、「添付ファイルのコンテンツを取得」というアクションがあることがわかりました。

第 11 章 フローで思い通りの結果を得るために大切なこと

新しいドキュメントセットを作成する	新しいドキュメントセットリスト品目を作成します。
新しいフォルダーの作成	新しいフォルダまたはフォルダパスを作成します。
添付ファイルのコンテンツを取得	ファイル識別子を使用してファイルの内容を返します。 内容は別の場所にコピーすることも、添付ファイルとして使用することもできます。
添付ファイルの削除	指定された添付ファイルを削除します。
添付ファイルの追加	指定したリストアイテムに新しい添付ファイルを追加します。
添付ファイルを取得する	指定されたリストアイテムの添付ファイルのリストを返します。「添付ファイルの内容の取得」の手順を追加し、このアクションによって返される「ファイル識別子」プロパティを使用して、ファイルの内容を取得できます。

図 11.24: 添付ファイルのコンテンツを取得 (出典:Microsoft Learn)

これを使えば、添付ファイルのコンテンツが取得できそうです。 Microsoft Learn のアクション名をクリックして、詳細ページに飛んでみましょう。

11.3.11 「添付ファイルのコンテンツを取得」アクションを理解する

初めて見るアクションは、Microsoft Learn で確認。ここまで読み進めた皆さん、きっと覚えられたと思います。

11.3 Case アップロードしたはずのファイルが開かない

添付ファイルのコンテンツを取得

操作 ID: GetAttachmentContent

ファイル識別子を使用してファイルの内容を返します。 内容は別の場所にコピーすることも、添付ファイルとして使用することもできます。

パラメーター

Name	キー	必須	型	説明
サイトのアドレス	dataset	True	string	例: https://contoso.sharepoint.com/sites/sitename 。
リスト名	table	True	string	SharePoint リスト名。
ID	itemId	True	integer	ファイルが添付されたリスト ID。
ファイル識別子	attachmentId	True	string	添付ファイルのファイル識別子。

戻り値

添付ファイルのコンテンツ。

添付ファイルのコンテンツ binary

図 11.25: 添付ファイルのコンテンツを取得 (出典:Microsoft Learn)

Microsoft Learn を読むと、ファイルが添付されたレコードの ID と 添付ファイルのファイル識別子がわかれば、添付ファイルのコンテンツを取得できるということが理解できます。

11.3.12 フローをコピーし、「添付ファイルのコンテンツを取得」アクションを追加する

いよいよ「添付ファイルのコンテンツを取得」アクションを追加して、「ファイルコンテンツ」を取得したいところですが、まずはフローをコピーしておきましょう。フローをコピーすることで現在のフローをそのまま残すことが可能になり、差分比較が行いやすくなります。フローは「名前を付けて保存」でコピーできます。

第11章 フローで思い通りの結果を得るために大切なこと

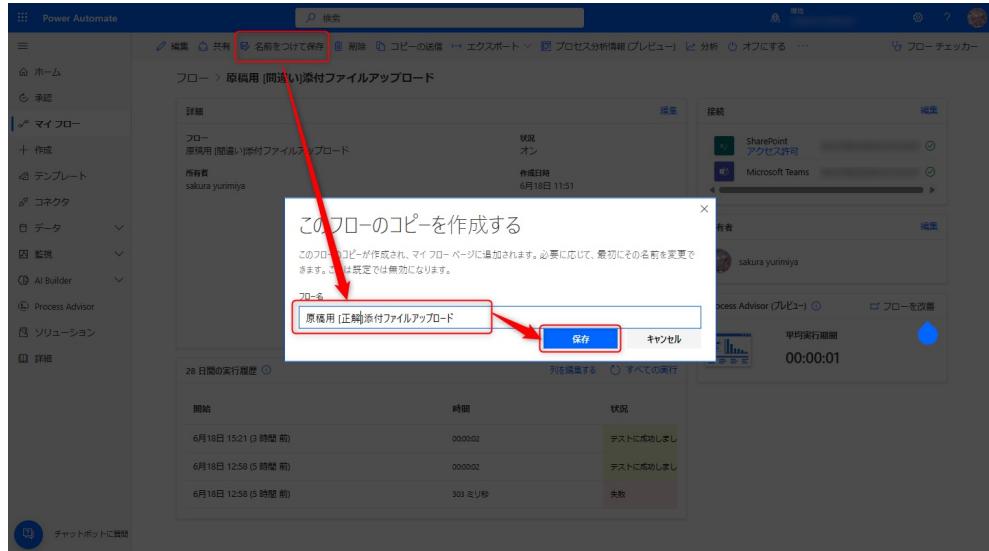


図 11.26: フローのコピー 1

コピーしたフローはデフォルトでオフになっていますので、オンにしてから編集しましょう。



図 11.27: フローのコピー 2

「添付ファイルのコンテンツの取得」アクションは、添付ファイルを1つずつ取得しな

11.3 Case アップロードしたはずのファイルが開かない

ければならないので、Apply to each アクションの先頭に入れます。

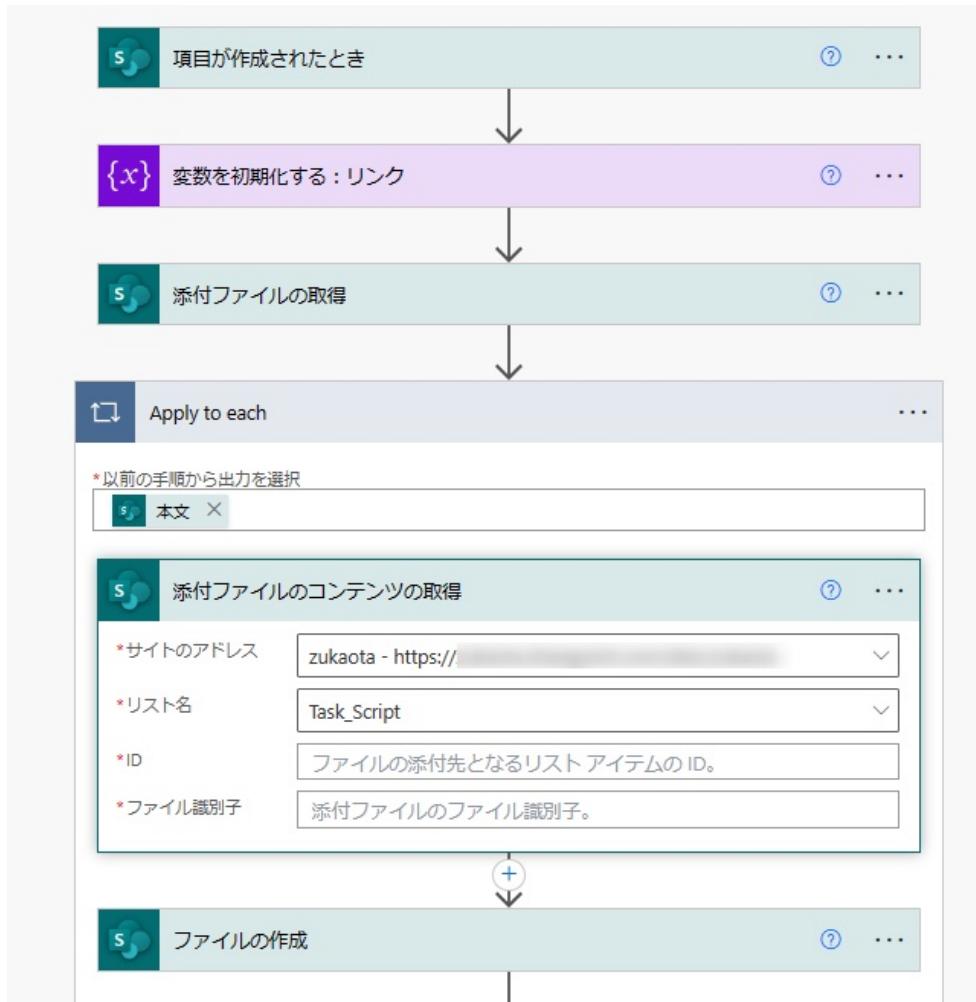


図 11.28: 添付ファイルのコンテンツの取得

11.3.13 ID と ファイル識別子が取得できるか確認する

では、先ほどのファイルのコンテンツと同様に、ID とファイル識別子が現在のフローで取得できているのかを確認しましょう。動的な値で「id」を検索した結果は画像の通りです。

第 11 章 フローで思い通りの結果を得るために大切なこと



図 11.29: 動的な値

項目名の下の説明で、「項目が作成されたとき」の「ID」が「ID」に、「添付ファイルの取得」の「Id」が「ファイル識別子」になりそうだということがわかりました。実際にアクションに動的な値を入れ込んでみると、こんな感じです。

11.3 Case アップロードしたはずのファイルが開かない

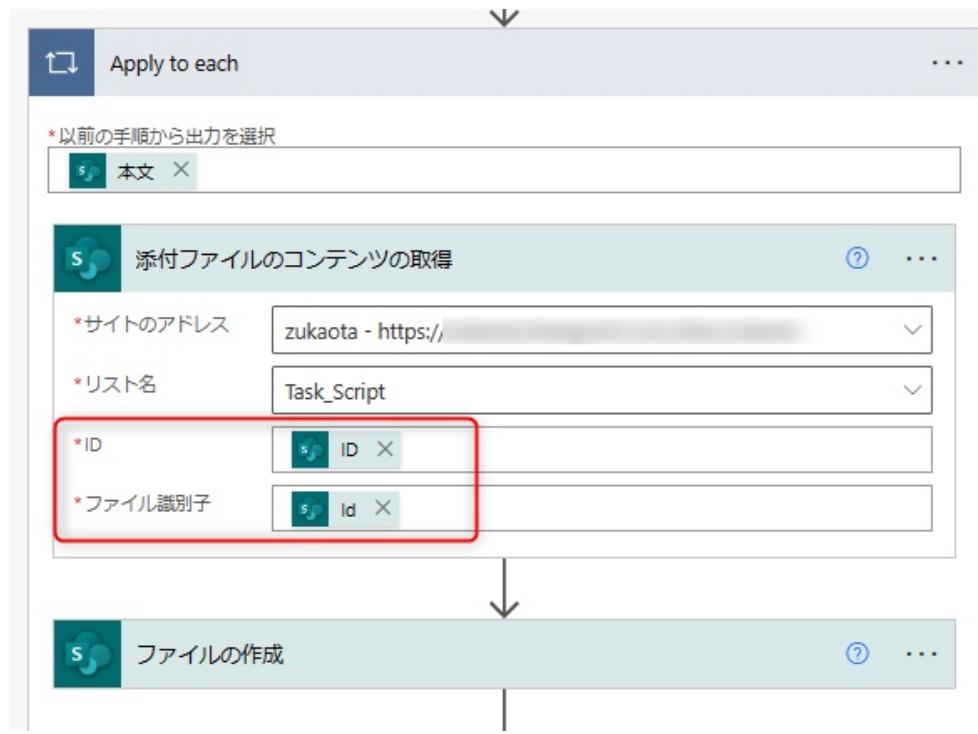


図 11.30: 添付ファイルのコンテンツの取得

11.3.14 「ファイルの作成」アクションの設定し直す

「添付ファイルのコンテンツの取得」アクションの設定が完了しました。これで「ファイルのコンテンツ」が取得できるようになるはずです。では改めて、「ファイルの作成」アクションの「ファイル コンテンツ」にカーソルをあてて、動的な値を検索してみましょう。

第 11 章 フローで思い通りの結果を得るために大切なこと

動的な値 Expression

コンテンツ

添付ファイルのコンテンツの取得

添付ファイルのコンテンツ

項目が作成されたとき

コンテンツの承認状態

このリストアイテムのコンテンツの承認に関連付けられているコメント

図 11.31: 動的な値

無事に「添付ファイルのコンテンツの取得」の「添付ファイルのコンテンツ」が出てきました。こちらを「ファイル コンテンツ」に入れていきましょう。

ファイルの作成

* サイトのアドレス: zukaota - https://

* フォルダーのパス: /TaskDocument/営業部

* ファイル名: DisplayName

* ファイルコンテンツ: 添付ファイ...

図 11.32: ファイルの作成

これで「ファイルが破損している可能性」を潰したことになります。改めてテストを行っていきましょう。

11.3 Case アップロードしたはずのファイルが開かない

11.3.15 フローをテストし直す

テスト機能を使用して、再びフローの動作テストを行います。最初のテストと同一の環境で行うために、1回目のテストでドキュメントライブラリにアップロードしたファイルは、すべて削除しておきましょう。



図 11.33: ドキュメントライブラリ

第 11 章 フローで思い通りの結果を得るために大切なこと

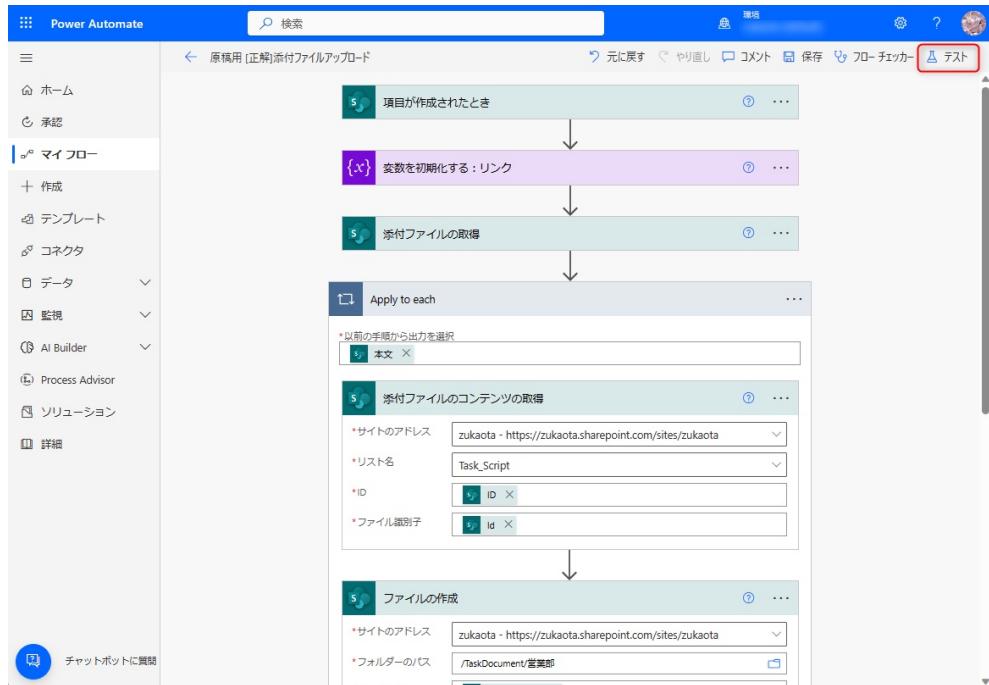


図 11.34: テストの実行

フローの成功が確認できました。

11.3 Case アップロードしたはずのファイルが開かない



図 11.35: フローの実行

前回同様、想定する「正常な動作」の確認をしていきます。まずは、Teams にメッセージが投稿されているかを確認します。

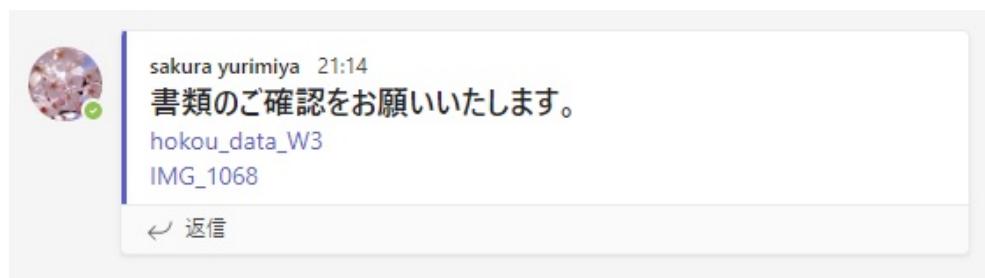
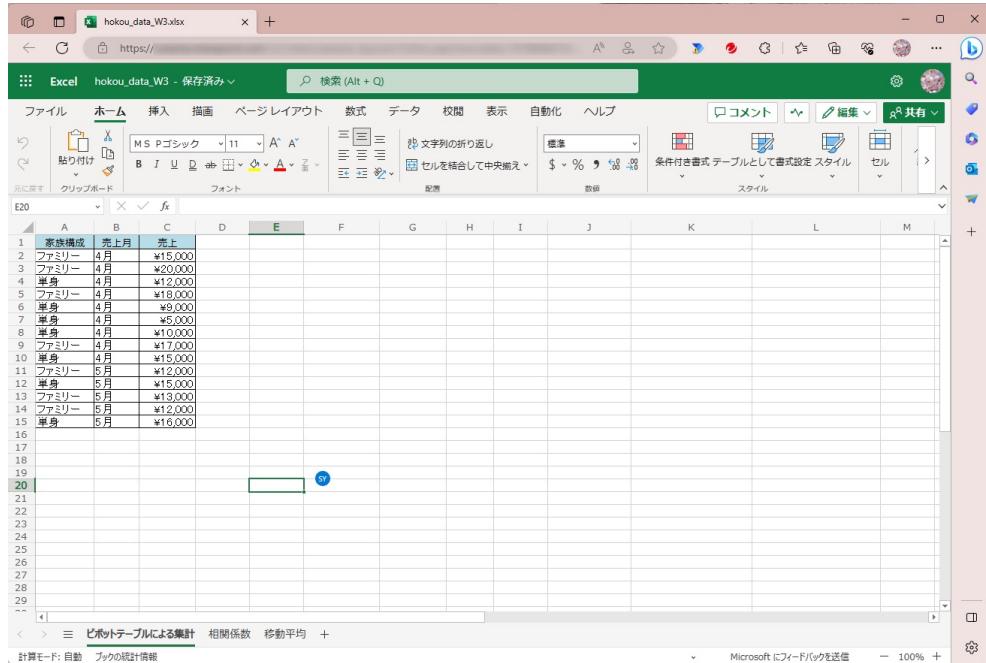


図 11.36: メッセージの投稿

きちんと投稿されていました。次にメッセージのリンクが開くかどうかを確認します。1つ目のリンク「hokou_data_W3」をクリックします。

第 11 章 フローで思い通りの結果を得るために大切なこと



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	家族構成	売上月	売上										
2	ファミリー	4月	¥15,000										
3	ファミリー	4月	¥20,000										
4	単身	4月	¥12,000										
5	ファミリー	4月	¥18,000										
6	単身	4月	¥9,000										
7	単身	4月	¥5,000										
8	単身	4月	¥10,000										
9	ファミリー	4月	¥17,000										
10	単身	4月	¥15,000										
11	ファミリー	5月	¥10,000										
12	単身	5月	¥15,000										
13	ファミリー	5月	¥13,000										
14	ファミリー	5月	¥12,000										
15	単身	5月	¥16,000										
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													

図 11.37: Excel ファイル「hokou_data_W3」の表示

Excel ファイルが問題なく開きました。2つ目のリンク「MG_1068」もクリックしてみます。

11.3 Case アップロードしたはずのファイルが開かない

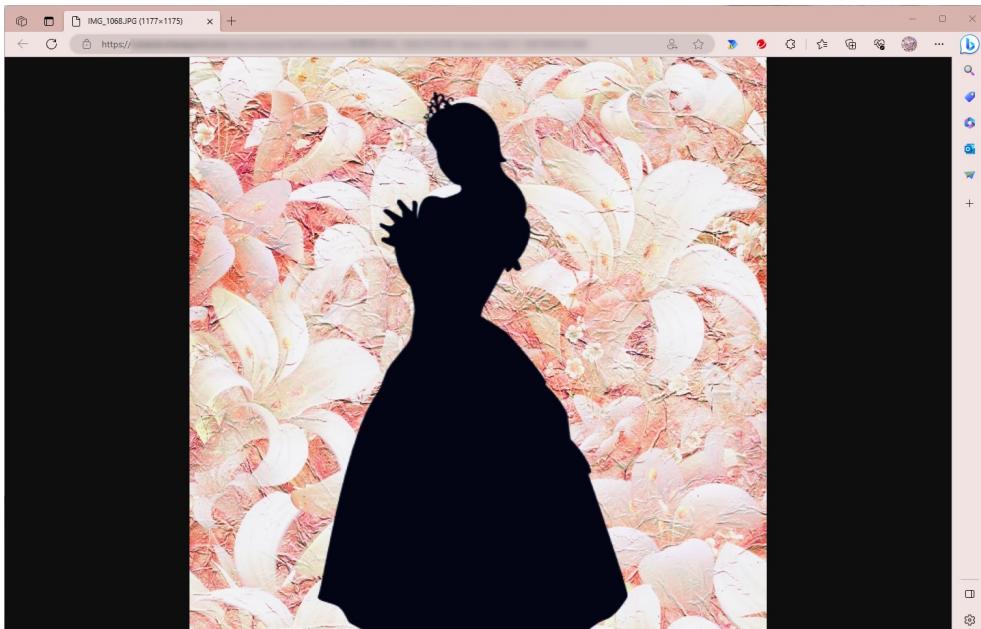


図 11.38: 画像ファイルの表示

こちらも問題なく開きました。ドキュメントライブラリ上のファイルサイズも確認しておきましょう。

第 11 章 フローで思い通りの結果を得るために大切なこと

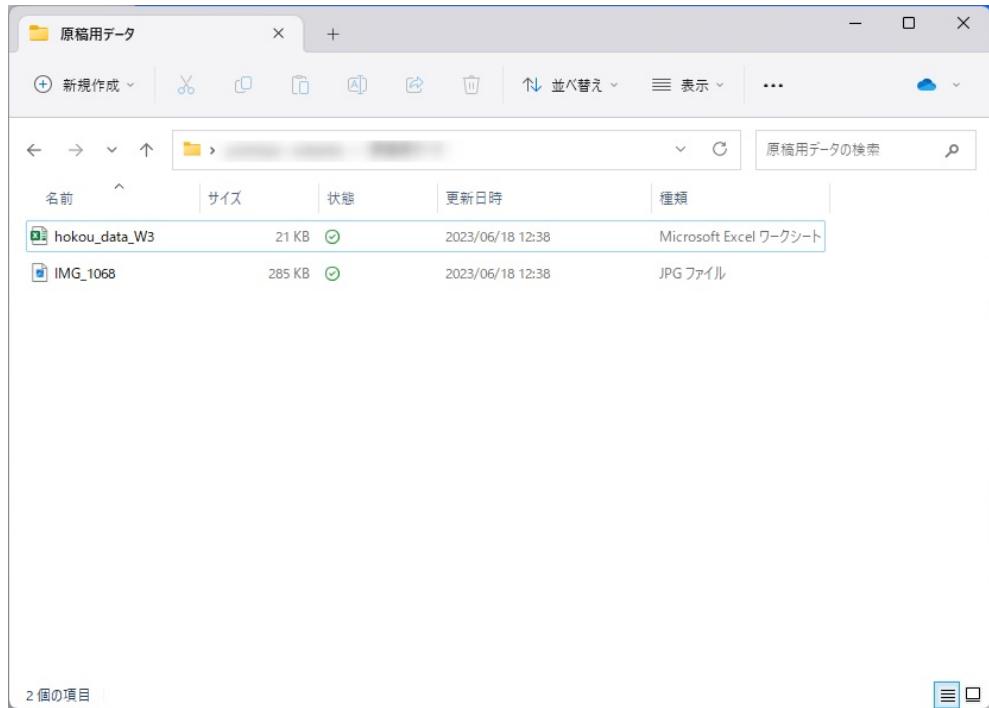


図 11.39: 元データ

TaskDocument > 営業部

名前	ファイルサイズ
hokou_data_W3.xlsx	20.8 KB
IMG_1068.JPG	285 KB

図 11.40: ドキュメントライブラリ上のデータ

元データとほぼ同じ大きさで、ファイルが作成されていることがわかりました。これでデバッグ完了です。作成者の想定通りに正しく動くフローができました。

11.4 バグのない正しいフローを作るために

11.4 バグのない正しいフローを作るために

バグのない正しいフローを作成するためには、テストが非常に重要です。アクション数が少ない小さなフローなら、頭の中でテスト項目を思い描いてテストすることも可能ですが、アクション数が多いフローや Power Apps など他のツールと連携するフローは頭の中でテスト項目を思い描いても、どうしても抜け漏れが出ることも多いです。そのため、バグを確実に潰した正しいフローを完成させるにはテスト項目を洗い出し、1つずつ確認していくことが必要です。最後の効率よく抜け漏れのないテストを行うためのコツを書いていきます。

11.4.1 テスト項目ってどうやって洗い出すの？

Power Automate の場合は、想定通りの「正常な動作」が何かを考え、それを書き出して確認していくのが良いです。今回のフローの場合は、以下の通りです。

- SharePoint リストにレコードが作成された時にフローが実行される。
- Teams にリンク付きメッセージが投稿される。
- メッセージの中にあるリンクを開くことができる。

11.4.2 テスト方法と想定結果、実際の結果を一覧表にして確認する

テストを漏れなくきっちり行うためには一覧表にするとわかりやすいです。次の表は、最初のテスト結果をまとめてみました。

テスト方法	想定結果	実際の結果
フローの実行 SharePoint リストにレコードを新規作成する	フローが実行される	想定通り
	Teams にメッセージが投稿される	想定通り
メッセージのリンク確認 メッセージ内のリンクをクリックし、問題なくリンクが開くか確認する	リンクが開き、ファイルが表示される	リンクが開かない

図 11.41: テストの一覧表

第 11 章 フローで思い通りの結果を得るために大切なこと

テスト方法の太字部分はテスト方法のタイトル、下の文章はテストの方法を説明しています。想定結果がいくつかある場合は、テスト方法のセルを結合させて、ひとまとまりに見えるようにしています。この方法の良いところは、4つあります。

- Excelなどの表が作成できるソフトを使うと、不足分の付け足しが簡単。
- 方法と想定結果/実際の結果が1行にまとまって見やすい。
- 確認事項が記録されるので、抜け漏れがなくなる。
- チームでの確認が行いやすい。

あくまでも一例ですが、テストで迷った時はぜひやってみてください。きっと以前よりも安心感のあるフローが作成できると思います。

11.5 終わりに

今回、Power Automate でのバグの潰し方とテスト方法を紹介いたしました。この2つを本書で紹介しようと思ったのは、実は Microsoft Build がきっかけです。Microsoft Build では、Windows や Power Platform をはじめとする数多くの製品への AI 搭載が発表されました。

Power Automate でも米国プレビュー環境のみですが、Copilot が実装されており、フローの作成は自然言語のみで可能となりつつあります。しかし Copilot はあくまでも「副」操縦士です。作成者のあなたの思考ややりたいことを完璧にトレースできるわけではありませんし、時には間違った解釈をしてしまうこともあります。簡単に動くものが作成できる時代だからこそ、フローが「想定通り」に「正常な動作」をするかどうかを確認するテスト作業は今後ますます重要になってきます。

皆さんにはぜひ入念なチェックに時間を使っていただき、楽しく AI と共に存する世界の担い手となって頂きたいと思います。ここまでお読みいただき、ありがとうございました。

第 12 章

普通の OL なら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

新山 実奈子



図 12.1: For Your DX

これをお読みくださっているということは、あなた様は、作業の自動化、ロボット作成にご興味があるか、または既に作成を始めた方とお見受けいたします。

ならば、輝く明日、円満退職、もしくは快適な老後のために、それらの作成や運営、保守作業の術をサッと身に着けていただき、役立ててまいりましょう。毎日の仕事を通して、これからずっと使える知恵を身の内に蓄積できますので、全力でお仕事をがんばって

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

まいりましょう。ええ、お仕事とは良いものです。対価を頂戴しつつ、実践の厳しさの中で自らの知恵、スキルの類を向上させてもらひただける、素晴らしい舞台でございます。

お仕事とは、あなたを育み、磨く舞台、いずれ自らまばゆく輝くための稽古場でもあると、そうお心得下さいませ。ちなみに、こちらの記事は特に文系の初心者/初学者の皆様に向けて書かせていただきました。草の根系ボトムアップ式の内容となります。お読みくださる際には。その旨をご承知の上、御覧くださいませね。

今回は、何らかのロボット作成 / 自動化ツールを導入する際にお役に立てるお話をさせていただきます。何事も使いまわしのきく御道具は重宝いたします。さあさ、御用とお急ぎのない方は、どうぞごゆるりと、自動化のコツをご照覧あれ。



図12.2: いつも心にノートを持とう

12.1 自動化の手順も手腕も、三者三様にござりまする。

代表的なローコードツールのRPA/自動化ツールを3種ご紹介します。いずれも新山がお仕事で使わせていただいている、非プログラマに優しい、扱いやすいツールです。

12.1 自動化の手順も手腕も、三者三様にござります。

12.1.1 WinActor



図 12.3: For WinActor

Windows 画面上であなたの役を演じる Actor ロボットを作成

公式サイト :

製造: NTT-AT (<https://winactor.biz/>)

販売: NTTdata (<https://winactor.com/>)

学習におすすめな無料動画: ヒューマンリソシア DX 公式 YouTube チャンネル (<https://www.youtube.com/@dxresocia>)

コピー ロボットならば、一晩中、踊りあかせる！

一言で申し上げるならば「プログラミングを知らなくてもロボットが組める」アプリです。一番の利点は、「ロボットを作成するために覚えることが少ない」こと。要は、組み立てが単純で、役立つロボットを作れる。

次の優れた点は「専用サーバが必要ないこと」。組んだロボットは、変数表を外に出して参照する形できっちり組んでおけば、ほぼそのまま管理ツールに乗せられますし、一晩中、働かせることもできます。

さらに管理ツールはオンプレミス版とクラウド版管理ツールがあり、このうちクラウド版は AWS 上で動きます。そのため、専用サーバを立ち上げて運用する為の費用と労力が不要です。このあたりの導入の容易さも魅力的な、純国産 RPA ツールです。ただし、お値段が張りますので、個人で勉強するには敷居が高いのが玉に傷。ですが、条件つきでお試し系の数か月だけ無料で使える、あるいは少額で試せるキャンペーンもありますので、ぜひともご検討ください。

さて、こちらの看板に「変数に入れてしまえばこちらのもの」と書きましたが、変数に

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

ついで簡単に説明します。PC上のUIで人間が手動で行う作業は、大別すると次の3つのうちのいずれかです。

それは、

1. アプリ画面のどこかを押すこと
2. 記入欄に値を設定（入力）すること
3. 出力された値を取得すること

ロボットに代行してもらう上記作業のうち、②と③で変数を用います。②では、値をExcel等アプリから読み込む際に、一度、変数に入れます。そして、③では出力された値を取得する際に、変数に入れてから、Excel等のアプリに書き込みます。

変数は「器」のような存在です。例えば、お財布や小銭入れに似ています。買い物の都度、必要な分の金額を用意し、必要に応じて支払う際に、それらを使用しますよね。品物によって、あるいは量によって金額が変わります。変わるものなので変数、と覚えておくと楽かもしれません。

変数とは逆に決まった数値を入れておく、定数というものもあります。こちらは消費税とかに例えるとイメージしやすいでしょうか。どの品物でも一律10%（ただし軽減税率の品物は除く）みたいな感じで、固定値として使われます。もし消費税率が上がれば、定数の数値を変更させることで、必要な修正箇所に対して一気に対応できますね。こちらの率ばかりは、上昇ではなく下降していただきたいものです。

12.1.2 UiPath



図 12.4: For UiPath

初心者と非エンジニアは StudioX から使い始めましょう

12.1 自動化の手順も手腕も、三者三様にござります。

公式サイト: <https://www.uipath.com/ja>

公式 E-learning: <https://www.uipath.com/ja/rpa/academy>

学習におすすめな無料動画: 自動化が誰でもデキる時代に！ ゼロから始める RPA 入門
UiPath StudioX 導入編

<https://www.udemy.com/course/ogushi-rpa-uipath-studiox-01/>

安心してください、日本語化されましたよ。

UiPath は、近年、初心者向けと上級者向けに組み立て用ツールが異なるようになりました。こちらのロボットは、人がボタンを押してスタートさせる扱いやすいものと、完全自動で動くものに分かれます。どちらも Excel と親和性が高く、扱いやすくなっています。初心者向けの優しいバージョンは、StudioX です。優しいとはいって、バックオフィスのお仕事自動化は、大抵これひとつで片付いてしまうほど高機能です。何よりうれしいのは、個人で使用する際には、無償バージョンを提供いただけます。制限付きですが、小規模事業者でも無償でお使いいただけます。(詳細は公式ページでご確認くださいね)

さらに嬉しいお知らせがございますよ。公式 E-learning が提供されているとはいって、初学者はひとりで学習を進めるうちに様々な不安に駆られたり、わからないこともどう調べればよいか困ってしまうことがよくありますよね。しかし、UiPath はイベントやコミュニティ活動が、とても充実しています。先輩ユーザーの皆さんも穏やかで優しい方ばかりなので、安心してコミュニティに参加できます。学習をすすめる際に、力になっていただけます。でもコミュニティに甘えすぎは禁物。参加者全員が、楽しく学習を進めていけるよう、長く良いお付き合いができるオトナとして、ご自重の上、ご利用くださると嬉しいです。さあ、コミュニティで僕と握手だ！

12.1.3 Power Automate (デスクトップ版)



図 12.5: For Power Automate

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

VBEでマクロを組む勢の皆様、ぜひ、お試しくださいね

公式サイト: <https://powerautomate.microsoft.com/ja-jp/>

公式E-learning: <https://learn.microsoft.com/ja-jp/training/>

学習におすすめな無料コースや資格情報等:

<https://www.microsoft.com/ja-jp/events/top/training-days>

Microsoft 365、Office系のアプリに準じたUIが、使いやすいです。あるいは、なにかしらMicrosoft系のプログラミング学習を行った、DosやらShellやらVisual Basic等の経験がある場合は、その経験を生かして素早い習熟が見込めます。Power Platformは単体で使うというよりも、各々のアプリの足りない部分をMicrosoft 365系の他のアプリで補完して複数のアプリを組み合わせて、自動化を広げていきます。勿論、Excelとも親和性が高く、Excelで培っていたスキルも生かせます。特にExcelの関数使いの皆様には、Appsの命令文の作成は容易です。習得を決意された暁には、ぜひ開発者向け環境を申請ください。OKとなれば、こちらもMicrosoft 365 E5プランの環境が無料で使えます。学習が捲りますよ。

そして、Power AutomateのDesktop版は（無償版であっても）、なんと、部品ごとに「例外処理」が指定できます。例外処理用の部品は、どのRPAツールにも専用部品がそろっていますが、通常はグループ等まとまった部品の集合に対して、行えるものになります。ひとつひとつの部品に標準で例外処理が組めるというのは、初心者には特に嬉しい機能です。

Appsで入出力、Automateが制御、SharepointをDBに！ + PowerQueryとの親和性が高く自動化の強い味方！

もちろんVBAやJSONとの親和性は、とてもとても高くて本当に安心して使えます！お仕事でExcelマクロをVBEで組んでいますよ、という皆様はこのツールを試さないと本当にもったいないです。

そしてそして、Excel2016で登場した「取得と変換」機能が、Power Platform版でパワーアップして名前をPower Queryとなりました！普段のお仕事で「取得と変換」をお使いの皆様は、今後の御社の自動化展開に向けて、ぜひPower Automateをお試しください。

■ Power Queryの解説は、初代Excel MVP田中亨先生のYouTube版がおすすめです！【機能】Power Queryとは何か？何ができるのか？を詳しく解説

<https://youtu.be/Z-08XG5wwRc>

■コラム: ロボット作成に役立つ”正しい”Excelの解説は、実績と信頼のOffice TANAKAからどうぞ。

お仕事でExcelを使われる際には、誰しも一度はお世話になったハズ。「みんな大

12.2 ロボット作成時の注意事項を申し上げますね。

好き！ VLOOKUP」の名言で有名な、田中先生のサイトはこれらです。

公式サイト: <http://www.officetanaka.net/index.htm>

公式 You Tube: <https://www.youtube.com/@OfficeTANAKA>

12.2 ロボット作成時の注意事項を申し上げますね。

繰り返しの多い単純作業等の労力を軽減するために、何らかの自動化ツールを導入する、あるいはロボットを作成しよう、または保守や運用を、とお考えですか？ ならば、失敗と落胆を避けるために、ここで少々お耳を拝借いたします。OL31年生、ロボット作成6年生の新山から、おすすめの下準備についてお話させていただきますね。

大丈夫、あなた様は、ご自身が担当するお仕事のプロです。どうすれば業務をこなせるかを身に着けてらっしゃいます。ということは、必要不可欠な要素の半分は、すでにお持ちです。ならば、身に着けたお仕事の手順や内容を、これまで後輩に引き継いできたように、ロボットに教えてあげればよいのです。ただ一点、心配な点があるとすれば、ロボットに教えるコツをまだご存じない、ということでしょうか。ご心配なく。私が教わったり編み出したりしたコツについてこれからお話ししてまいります。このお話の幾許かでも、自動化作成のお役に立てれば幸いです。

12.3 突然のツール変更も視野に入れた準備が必要です。

さて、「会社がRPA/自動化ツールを導入することになりました」となりますと、ある種の覚悟が必要です。いずれのツールも値が張るもので、まずは、3か月なり半年なりの。無料のトライアル期間から入られるかと存じます。トライアル期間を経て、あるいは契約が切れるタイミングで、自動化作成ツールを突然変更されることも、実は、ままあります。そう珍しいことではありません。変更理由は、様々です。費用対効果であったり、客先や本社に合わせようであったり、他の施策や顧客との兼ね合いでということもあります。

そして、この変更は、作成者側に大きな打撃を与えます。例えば、これが事務用品であれば、変更されたとしても、使い勝手が悪くなる程度で済みますし、慣れてしまえばそれほど困りません。あるいは、消耗品費の枠内でこっそり買いなおしてしまう、という手も使えますが、変更対象が「自動化作成ツール」となれば、なかなかそうは参りません。

なぜ作成ツールの変更が大打撃を与えるかと申しますと、まず、各社のRPAや自動化ツールの間には、互換性はほとんどありません。すなわち、ロボットはすべて作り直しと

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

なります。作成画面も運用画面も用語さえも変わってしまいます。これまでRPA導入のために勉強してきたことが、ほぼ全部通用しなくなります。つまり、何か月も残業をしてまで得たスキルが、ある日突然、使い物にならなくなります。時間的にも費用から見てもメンタル的にも、大赤字確定です。

この悲劇を避けるためにはどうすればよいのでしょうか。それには、「ツールが変更されることもある」ことを念頭に置き、最初の段階から準備をしておけばよいのです。途中でRPAツールの変更があっても対応できる形で準備をすすめましょう。

特に、着目すべき点があります。すべてのRPAツールに有効な、最強のアプリの存在です。この最強のアプリを、あなたの味方につけるのです。どのRPAツールにも必ず専用部品を用意させるアプリ界の超実力者、その名は、皆様おなじみ、どこの職場でも大活躍、“みんな大好きVLOOKUP”的Excelです。「応用の利く基礎が詰まった”正しい”Excelブックを必ず作り出し、あるいは発掘し、探し出し、装備する。」これが、RPA/自動化のハードルを下げ、早期導入化をかなえてくれる立役者になり得ます。

Excelごときで、そう変わるわけではないと思われますか？皆様は職場で、業務引継ぎの折に、"これ仕事で使ってるから"と、先輩からExcel文書を渡されませんでしたか？他の表計算ソフトを熟知しているわけでもないため、代替の提案もできず、いわばなし崩し的に、ずるずると、そのExcelとお付き合いを続けておられませんか？

私もそうでした。しかしながら、OLを続けるうちに気が付きました。実は正しいExcelと、そうではないExcelが存在します。どうやって正しいExcelを見分け、味方につけるのかは、若干、長いお話になりますので、ここでは詳しく説明できませんが、自動化やRPAを扱ううえでExcelがあなたを助けてくれることを、この機会に覚えておいてくださいませね。

12.4 “最初から 100% 自動化を目指さない” が成功の鍵！



図 12.6: Excel を自動化に組み込む利点

12.4 “最初から 100% 自動化を目指さない” が成功の鍵！

まずはスモールスタートから始めましょう。自動化を行う予定の作業については、開始時点では、ご自身とロボット（あるいは自動化ツール）との協業からはじめることを検討ください。次に、徐々にロボットに任せる比率を大きくしていくことをお考えください。この考え方には、ロボットを作成することにも、スキルを効率よくあげていくにも、運用保守することにも有効です。

これから、具体的にどのように段階的にロボット/自動化をすすめていくかについて私の経験から、おすすめの手順をお話していきます。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

最初から 100%の 自動化は 目指さない

1. 自動化の優先順位を決める
費用対効果を計算して、優先順位をつける。
2. 作業を塊に分ける
自動化する予定の作業を、箇条書きで並べましょう。
3. 作業を見直す
箇条書きに並べた各作業が最適であるかを検討します。
4. 分解して再構成する
箇条書きの作業を区切りの良いところで分けていきます。
5. 分担を決める
ロボットと協業する内容とスケジュールを決めます。



図12.7: 最初から100%の自動化は目指さない

<作業手順の例>

- 「自動化の優先順位を決める」 費用対効果を計算して、優先順位をつける。
- 「作業を塊に分ける」 自動化する予定の作業を、箇条書きで並べましょう。
- 「作業を見直す」 箇条書きに並べた各作業が最適であるかを検討します。
- 「分解して再構成する」 箇条書きの作業を区切りの良いところで分けていきます。
- 「分担を決める」 ロボットと協業する内容とスケジュールを決めます。

それでは、上記それぞれについて、以下で詳細に、ご案内いたします。ですが、その前にひとつだけ。

12.5 安易に下請けを使うと、危険です。保守ができずに、すぐ錆びてあっけなく壊れます。

これまで、社員自らが作成したと喧伝されるロボットやDominoのような自動化がございます。実は、その多くは黒子が作成しています。黒子は自社社員の場合もありますが、契約社員や派遣であったり、下請け孫請けに作成させたものです。これは、リーマンショック以降、正社員数の急激な現象の影響かと思われます。社内は本業を（残業込みで）ギリギリ回せる人数しか残っておらず、再教育するお金も手段もないことから、教育は外部委託、必要あれば稟議書を回して外注というのが、私が経験してきた2010年以降のスタイルです。

12.5 安易に下請けを使うと、危険です。保守ができずに、すぐ錆びてあっけなく壊れます。

それまではセキュリティの観点から、外部講師を招いて自社内で社員が作成するというスタイルが主流でした。しかし、終身雇用制度が崩れて社員数が激減した以降は、社員自らが作成する機会自体が、本当に少なくなりました。

ここ数年、リスクリングという名で、向上心の高い社員が自らスキルを身に着けようとする動きがあります。それも全社員が行えるかというと、これは現実的には余裕のない企業が多く厳しい状況です。200社の人事部担当者を対象に、学校法人産業能率大学総合研究所が出した「日本の企業・組織におけるリスクリング実態調査 報告書 2021」において、リスクリングという言葉辞退の認知度が36%、すでに取り組んでいるのは、11.5%でした。

<https://www.hj.sanno.ac.jp/cp/research-report/2021/09/30-01.html>

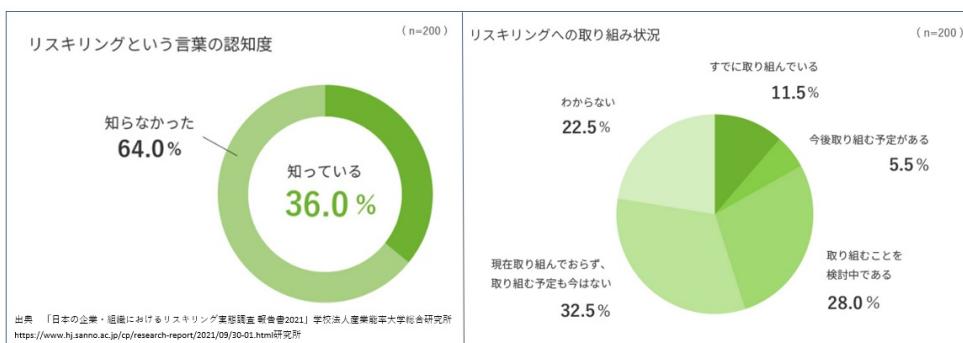


図 12.8: リスクリング

上記から、リスクリングが浸透しているとは言えない状況です。しかしこれを踏まてもなお、ロボットに関しては、社員の方のリスクリングによる技術習得をお考え下さい。外部から講師を呼んで、しっかりと習得して作成していただいたロボットは、外注した場合と本質が異なります。

自動化作成の現実を見てみましょう。下請けが作成したロボットを使用していて、変更が必要になった、あるいは壊れた場合のことをお考えください。RPA/自動化の歴史は浅く、いまだ標準化や均質化に至っていません。ロボット/自動化作成というのは、属人化的極みです。そうなりますと、作成したロボット/自動化は、ほぼ本人にしか保守できませんし、修復や改修は不可能に近くなる。これが現状です。

もし壊れた場合、派遣や下請けから作成者本人を呼び戻すことは、できるでしょうか？残念ながら、まず不可能です。彼らは、次の仕事に派遣されているか、移籍して他社で仕事を行っています。するとどうなるでしょうか？ 別の人が派遣されます。考えてみてください。もし属人化されたプログラムを、何ら引継ぎなく、資料もほとんどない状態で、修正しろと渡されたとしたら、彼はどうするでしょうか？

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

その場合の多くは、契約期間内に、「最初から作り直す」または「元通りには使えないが、いくつかの部分だけはなんとか動く、似て非なるモノを作成する」あるいは「動かせないし、直せない」ことになります。

なお、下請けに丸投げしても、期間内に希望した品質のロボット/自動化が出来上がるることは稀です。バブル期以降の「下請けに無茶振りすれば、ALL OK!」文化は、通用しなくなっています。下請けも、それをわかっていますから、契約時に「必ず完成品を引き渡す」との条項は入れずに、「努力目標」に留めます。

ここは、覚悟をお決めください。社員の皆様自らで作成し保守運用を行っていただくことは、結果としてコスト減にもつながります、ぜひとも社員の皆様に、スマートスタートから自動化を広げていただきたい。そして、自らと共に働く仲間たちの手で、輝く未来を、切り開いて参りましょう。

お待たせしました。それでは、作業手順の説明を始めてまいりましょう。

12.6 最初は、半分だけ手伝ってもらうつもりで進めましょう

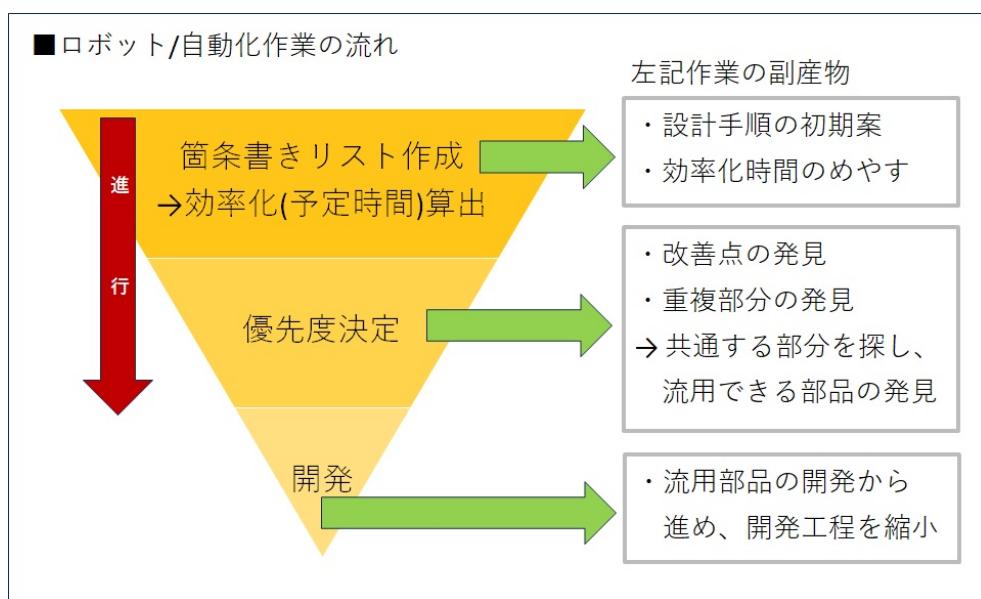


図 12.9: ロボット/自動化作業の流れ

業務をいくつも抱えておられるかと思います。それらをひとつずつ、確実に、正確に自動化しようと、考えられておられませんでしょうか？これまで他のアプリで作成したも

12.6 最初は、半分だけ手伝ってもらうつもりで進めましょう

のも、この機会にロボット/自動化に変更しようとされておられますか？

それは、危険です。難易度が跳ね上がります。初めてロボットを作るときに、難易度の高いものを目指してしまうと、挫折への近道になりますよ。100% 確実になると、設定が増えます。ロボットに任せる部分が大きいということは、設定することが多いということ。つまり、作成者側が覚えなければならないことも多くなりますし、確認やテストをしなければならない箇所も増えてまいります。比例してエラーの発生率も上がりますし、比例して修正箇所も増えると予想できますよね。

それを避けるためには、例えば、「今まで他のアプリを使って作業できていた部分は変更せずに残す。」と決めます。これだけでも、かなりの労力が低減化できます。そして、作業の流れを、すべて一度に置き換えるとせずに、段階的に置き換えること、小さくとも成功例を増やすことを優先しましょう。

最初は単機能、次の段階では別の単機能をと、単純なものをいくつか作成し、小さな成功を積み重ねて、徐々にスキルを上げていきます。最後に、作成した小さなロボットを組み合わせて、完全自動化まで持っていく、この順番がおすすめです。この方法で 100% の完全自動化ロボットをテストし、それが確実に動くとなったら、その後には、人間の手を離れ、完全自動で、会議中や終業後から翌朝まで、残業代も休日手当もなく働いてもらうことも可能になるかもしれません。

まずは、業務の中から単純な繰り返し作業の箇所だけを、ロボットに担当させて、人間あるいは Excel で確認や突合の作業を挟んでください。ご存じの通り、技術の習得をする際は、最初のひとつを完成されることに最も時間がかかります。その次からは、同じ内容であっても、所要時間は短くなります。いくつか作成するうちに工夫が生まれ、よいものが作れるようになっていきます。ですから、簡単なロボットから始めましょう。いくつか簡単なロボットを作成することでスキルが上がり、それに伴って視野と発想が広がり、解決策にも幅が出てきます。

まずは、半分だけ、あるいは少しだけ手伝ってもらうために簡単な、ごく小さなロボットをいくつか作成し、実際に動かしてみながら、ロボットと人間が協業することに慣れる、その成功例を横展開で社内に広げることが、自動化を無理なく成功させるためには必要です。その視点を手に入れることを目指してくださいね。まずは、あなた様から、仲間たちに、そして、社内の皆さんとの順番で、よろしくお願ひいたします。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

12.6.1 自動化の優先順位を決めましょう。

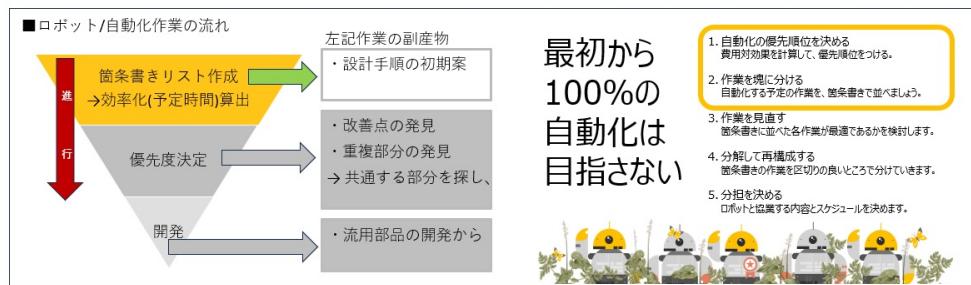


図 12.10: 自動化の優先順位を決める

さて、上司を説得、あるいは納得させるためには、効率化できる裏付けになる数字が必要になります。どの業務から手を付けるかは大きな問題です。おすすめは費用対効果を計算して、優先順位をつける方法です。まず、担当業務リストを作ります。次に、そのリストから、ロボット/自動化部品を選定します。テーマや処理の流れ、手順が異なる上位3つの単純なつくりの、ただし行う作業にかぶりのないロボットを選んで作成しましょう。これらを、ひな形にして、横展開する際や、先々で作成するであろう複雑なロボットの基礎になる"素体"を作成しておくのがおすすめです。

担当業務リストには、具体的な作業を箇条書きで並べていきます。そのあとで、必要に応じて適宜組み換え等を行い、あなたの作業に沿った形で最適化していきます。上司用の、説明用資料のメモもあわせて書き添えておきましょう。手順は次の通りです。

1. 任意の担当業務を洗い出し、それぞれの内容を単純な箇条書きとしリストを作成。
2. 自動化を予定している担当業務の数だけ、すべて①を行う。
3. およそその時間を算出し、ひと月あたり何人月かかっているかを割り出す。

割り出した時間数から、業務毎に効率化できる時間が決まります。他に配慮する項目がなければ、この時間が大きなものから作成します。なお、上記の1.から3.で割り出した時間は、のちのち上司に説明する資料になりますので、メモしておきましょう。

12.6.2 これから作成するメモのサンプル

さあ、次の例を参考に、あなたのメモを準備していきましょう。

■ 会議資料の作成

12.7 現状の状態と、どの程度効率化できるかを可視化する

- 毎週火曜日、午後一開催の会議で使う資料を作成する。
- 金曜日の午後 4 時に課長にメールで送信。CC に主任を入れる。
- 変更点あれば月曜日午前中に修正、なければそのまま会議フォルダへ格納する。
- データ集めは吉田さんと田中さんと、あと経理の大泉先輩に手伝ってもらう。
- Net 上にある収集用 URL と ID と PW は変更ないかチェックしてから 3 人に配布。

上記のようなメモを、業務の数だけ作成してくださいね。最初は、骨組みだけで OK。

12.7 現状の状態と、どの程度効率化できるかを可視化する

それでは、時間を算出し、ひと月あたり何人月かかっているかを割り出しますよ。最終的に、どれだけの時間が削減できる見込みがあるかを算出していきます。

- 計算の手順
 - (1) 任意の作業に対して現状でどれだけ時間かけているかを、算出します。
 - (2) その作業のうち、PC で行う作業の割合(率)を仮定します。
* この部分が、後でロボット/自動化で効率化できる時間になります。
 - (3) (1) に (2) をかけて、効率化できる(予定の)時間を産出します。
- 単純な計算の例: 会議資料を作成する際には、資料作成の時間の他に、作成のために情報収集の時間が必要ですし、承認者が存在する場合は承認者の時間も必要です。これらを併せて考えます。
 - (1) 毎週末に行うのであれば 4 回、作業時間が 2h あれば $4 \times 2 = 8H$ です。
 - (2) その作業の 90% を PC で行う場合は、 $8H \times 0.9 = 7.2H$ になります。
 - (3) 忘れずに作業人数を (2) にかけて、作業ごとの合計時間をしてください。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

業務種類	業務内容	単位 作業時間	人数	回数	PC作業率	効率化 予想時間	効率化 難易度	優先順位
会議資料 作成	資料収集	60分	3	4	90%	13H	容易	高
	作成時間	120分	1	4	85%	14H	容易	
	承認時間	15分	1	4	100%	1H	容易	
経費申請	資料収集	15分	8	1	30%	1.3H	容易	低
	作成時間	20分	8	1	50%	1.3H	容易	
	承認時間	30分	1	1	100%	0.5H	容易	

図12.11: 単純な計算の例

ひとつの担当業務を行う際には、複数の作業があり、またその作業ごとに細かい作業が発生しますので、その作業毎に、上記の時間を算出します。

- リストに箇条書きになっている箇所のおおよその時間を青字で書き込みます。
- 1か月ごとに算出します。複数回実行する作業はその回数分をかけてください。
- 他の部門に手伝ってもらっている分、下請けに出している分も考慮します。
- おおよその時間を合計して、判断材料にします。

冒頭で申し上げた担当業務の各箇条書きリストに対して、費用対効果の表を作り、どの部分を自動化するのが効率的であるかを勘案して、上位のものからロボット/自動化するかの優先順位を決めていきましょう。

12.7.1 作業を塊に分けましょう。

仮の優先順位が決まりましたら、次に自動化する範囲を決めていきます。目印になるのが作業の塊です。作業を塊・まとまりごとに分けて、効率的に自動化できる範囲を見つけていきましょう。

- 箇条書きの担当業務リストから引継ぎ準備メモを作る。
 - 各作業の引継ぎ用メモも、箇条書きにして並べます。
 - 余白は、十二分に取ってね。
 - 番号を付けたり、印をつけたりして、わかりやすくしましょう。
 - ”繰り返しの単純業務”があれば赤枠で囲むか、小さく赤字等で書き添える。
 - 感想もヒントになり得ます。最初のうちは、書き加えていきましょう。

12.7 現状の状態と、どの程度効率化できるかを可視化する

12.7.2 これから作成するリストのサンプル

さあ、次の例を参考に、あなたの準備メモを修正していきましょう。

■ 会議資料の作成 その 2 (準備用)

- 目的: 毎週火曜日、午後一開催の会議で使う資料を作成する。
 - 目的は、当面変更ないのでこのままで OK。
 - 資料集めのときに、フォーマット渡してあるけど、結構、バラバラな様式になる。これを自動で揃えたい。
 - Excel のマクロで、シートに貼り付けると、整理して表にコピペできるのを見た。よその部署だけど、譲ってくれないかな。頼めないけど、教えてもらえるかな。本当は、こっちを自動化したい。
- 日程: 金曜日の午後 4 時に課長宛にメール。資料を添付して送信。CC に主任を入れる。
 - 人事異動の時に宛先変更あり（現状ではこの部分は自動化が難しそう）
 - 异動報がイントラサイトに掲載されるようになったら、自動化できる？
 - メーリングリストを作れば一斉同報できるから、これは OutLooook でも OK。
送信前に目視確認しよう。確認誤送信したら始末書だから、それは避けたい。

手順：

- 変更点あれば月曜日午前中に修正、なければそのまま会議フォルダへ Up.
 - フォルダの名前とパス (どこにあるか) を調べておく。
 - 変更あるかないかで、作業が分かれる。分岐発生。
 - フォルダも整理したほうがロボットにわかりやすいかな。
- データ集めは吉田さんと田中さんと、あと経理の大泉先輩に手伝ってもらう。
 - ここが全自動化できたら、先輩たちの手が空くかな？
- Net 上にある収集用 URL と ID と PW は変更ないかチェックしてから、お手伝い 3 人衆に配布。
 - メンテ等で変更もありえる。メールは私宛にくるから目視でチェックが必要。

メモにどんどん追加して具体化していきましょう。これを、業務の数だけ作成してくださいね。先々で役立ちますので、面倒ですががんばりましょう！

担当業務数だけのリストを作成いただきました。さらに、そのリストの中には業務に必要な作業を箇条書きにした内容を作成いただいたかと思います。何事も、アウトプットは大事です。時間はかかりますが、自動化に必要な数だけ、リストの作成を引き続きお願い

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

します。次の作業では、これら作成したリストを、先々で自動化の設計図に流用できるよう直していきますよ。

12.7.3 リストの修正タイム

さあ、設計図に向けて整えていきましょう。下表はサンプルの修正例ですよ。

■ 会議資料の作成 その2(清書)

- 目的: 会議で使う資料を作成する。
 - 自動化希望案 1. 「メール添付資料から、統合時の様式を統一する」
 - 手順
 - * 1. Outlook 受信フォルダ内の該当メールを特定する。
 - * 2. 当該メールの添付資料を、PW 入力して解凍する。
 - * 3. 解凍した資料を、メモ帳経由で任意の Excel に貼り付ける。
 - * 4. 任意の Excel には、あらかじめ書式を設定しておく。
 - 代替案: 「データ収集そのものを自動化」
 - 現行では、他部署含め 3 名に協力してデータ収集を行ってもらっているが、これをロボットに代行させることで、大きく省力化が見込める。
 - 手順
 - * 1. インターネット上のファイルサーバに接続。
 - * 2. ID/PW の入力。(セキュリティのためここは手作業)
 - * 3. 任意のファイルを指定して、任意のフォルダへダウンロードを行う。
 - * 4. 作業後にログアウトする。
 - * 5. 任意のフォルダから、当該ファイルを PW を使って開く。
 - * 6. 資料収集用ファイルに開いたファイルから必要箇所をkopペする。
 - * 1. から 6. の作業を必要なデータ分、繰り返す。
 - 手動確認 1. URL、ID および PW が変更されていないかを事前に確認する。
 - 事前準備 1. 作業が容易に自動化できるよう、フォルダ整理を行う。
 - 自動化希望 2. 「資料作成時の労力を低減化する。」
 - 検討: Excel の関数またはマクロ導入による自動化。
 - * 他部署にて成功事例あり。
 - * 当該部署に取材と協力を希望する。
 - 自動化希望 3. 「資料修正時の労力を低減化する。」
 - 検討: 上司から修正の指示があった場合の、対応見直しを行いたい。

12.7 現状の状態と、どの程度効率化できるかを可視化する

- * フォルダ移動作業の自動化を行いたい。
- * 事前にフォルダ名、ファイル名のルール等の整備を行う必要あり。

- 日程: 開催日時 毎週火曜日 13時~
 - 準備期間 金曜日の午後4時に課長宛に資料添付 (CC: 全主任)。
 - 手動確認 1. 「人事異動時に宛先変更の確認を行う。」
 - * 異動報がイントラサイトに掲載された場合は、こちらも自動化検討を希望。
 - 手動確認 2. 「事前承認用のメーリングリスト送信前には、必ず目視確認を行う」

効率化、実現へのヒント等を、思いついた順に、どんどん追加していきます。
では、上記を踏まえてリストの箇条書きを見直す作業に入りましょう。具体例をあげていきますので、参考になさってください。

- 着目点
 - “人間の判断が必要である”ものに取り消し線をつける。(消さないでね) 取り消し線という目印で区切る理由があります。その前後で分けることで、部分的に自動化できる場合があるからです。
 - 繰り返しを見つけましょう。リスト内の業務を分解して考えて、お決まりの単純作業を探します。発見できたら、その個所をくるっと囲むか、星印等の目印をつけて、後で見分けられるようにしておきます。
 - 分岐を見つけましょう。斜線を入れて、赤字で「ここで分岐」と書き込みます。分岐先への矢印をつけるか、何か番号をつけてどこへ進むかを明らかにします。
 - 書き込むことが増えた場合は、切り貼りを行うのも手です。別の用紙を準備して、貼り付けていく、あるいは付箋で付け足す、いっそ全部付箋に置き換えもOK。
 - Word や Excel、メモ帳を使うのも、もちろんアリです。やりやすいツールで Go!
 - この作業は、ゆくゆくは効率化につながります。面倒ですが、がんばりましょう。

自動化するお仕事の数だけ繰り返し、塊リストを作成しましょう。リストをコピーして見直す。今度は、自分が手伝ってもらえると助かる順にする。この段階で、作業時間効率化メモと同様に、上司説明用の文書用のメモを作っておく。適宜、付箋貼るでもOK。本格的に優先順位が決まったら、次の作業に進みます。

ポイントは、分岐箇所を発見することと、単純作業が繰り返される部分の特定です。
なお、PC画面でロボットが行う基本的な行動は、次の3点です。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

1. アプリ画面のどこかを押すこと
2. 記入欄に値を設定(入力)すること
3. 出力された値を取得すること

この3点を、現実の作業にどう落とし込むかを考えてリストの内容を見直しましょう。とはいっても、どういった部品があるのかが不明のままだと、イメージがわかないですね。部品名は異なりますが、おおよそ、次のような部品がどのツールでも提供されています。細分化された約500程度の部品が提供されていますが、実際に使う数はおおよそ数十個でカバーできます。次に、RPAツール側で提供される部品を簡単にまとめましたので、おおよその参考になさってください。

12.8 およそ大体のRPAアプリで準備されている部品の簡単なまとめ

- 変数に値を入れたり出したりする部品
- フォルダやファイル、URLを開いたり閉じたりする部品
- ExcelやWord等のOffice製品のための部品
- テキストを書き込んだり読み込んだりする部品
- 数字や日時を計算する部品
- 変数内の文字列や数式をあれやこれや変更したり追加や削除する部品
- 繰り返しの回数や範囲を指定する部品
- エラー対策用の部品（エラーを見つけたり、エラーが起きた場合の処理を指定）
- 画面の全体指定、範囲指定、キャプチャ等を行う部品
- 各種ブラウザへの対応、操作を行う部品
- サブルーチン（よく使う部品の塊 / 必要な都度、呼び出して使用する）制御
- グルーピング用の部品（部品をまとめてグループ化し扱いやすくする）
- グループ毎、あるいは全体を制御用部品
- 動作時に画面にRPAアプリを残す、残さないスイッチ、一時停止等を扱う部品

12.8.1 リストの修正タイム

さあ、さらに具体化に向けて、メモを修正していきましょう。

■ 会議資料の作成 その3

- 目的：会議で使う資料を作成する。

12.8 およそ大体の RPA アプリで準備されている部品の簡単なまとめ

- 自動化希望案 1. 「メール添付資料から、統合時の様式を統一する」
 - 小型ロボ 1. 受信メールから添付資料を探し、解凍して指定のフォルダに格納する。
 - * ロボットは 1 の作業を、3 回繰り返す。
 - * メール有無により分岐も作れるが、今回は複雑にせずまとめて処理。
 - * 手動：処理が空振りしないように、受信を確認してから動かす。
 - 小型ロボ 2. 指定のフォルダ内の各ファイルを開いて、必要部分をコピーする。
コピーした内容を、資料ファイルへペーストする。
 - * ロボットは 1 の作業を、3 回繰り返す。
 - * ファイル有無により分岐も作れるが、今回は複雑にせずまとめて処理。
 - * 手動：処理が空振りしないように、ファイル数を確認してから動かす。
 - 小型ロボ 3. 資料ファイルにあらかじめ設定した、関数 / マクロを動かす。
 - * ロボットは 1 の作業を、3 回繰り返す。
 - * 手動：処理が空振りしないように、ファイル数を確認してから動かす。
 - * 手動：要望通りに資料転記が行われたか、間違いないかをチェックする。
- 手順:
 - ① Outlook 受信フォルダ内の該当メールを特定する。
 - * 1) Outlook は開いておく。
 - * 2) 送信元の名前と、メールタイトルから対象を探す。
 - ② 当該メールの添付資料を、PW 入力して解凍する。
 - * 1) 解凍用のフォルダを作成しておく。
 - * 2) PW を確認しておく。外部から見られない形で保存しておく。
 - * 3) 解凍後に、もとのファイルは保存しない。
 - ③ 解凍した資料を、メモ帳経由で任意の Excel に貼り付ける。
 - ④ 貼り付ける側の Excel には、あらかじめ書式を設定しておく。
 - * 1) メモ帳を開く。
 - * 2) 転記元ファイルを開いて、必要個所をコピーする。
 - * 3) 書式を削除するため、いったんメモ帳にペーストする。
 - * 3) メモ帳から、必要個所をコピーする。
 - * 4) 転記先ファイルを開いて、必要個所をペーストする。
- 代替案：「データ収集そのものを自動化」
 - 現行では、他部署含め 3 名に協力してデータ収集を行ってもらっているが、これをロボットに代行させることで、大きく省力化が見込める。
- 手順:
 - ① インターネット上のファイルサーバに接続。
 - * 1) 事前に URL に変更ないか確認する

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

- ②ID/PWの入力。(セキュリティに関わるため、ここは手作業で対応します)
- ③任意のファイルを指定して、任意のフォルダへダウンロードを行う。
 - * 1) 事前にファイル名、フォルダのパスを確認する。
- ④作業後にログアウトする。
- ⑤任意のフォルダから、当該ファイルをPWを使って開く。
 - * 1) 事前にPWを確認する。
- ⑥資料収集用ファイルに開いたファイルから必要箇所をkopペする。
 - * 小型ロボ②を流用、一部修正して作業を行う。
 - * 上記の①から⑥の作業を必要なデータ分、繰り返す。
- 手動確認 1. URL、IDおよびPWが変更されていないかを事前に確認する。
- 事前準備 1. 作業が容易に自動化できるよう、フォルダ整理を行う。
- 自動化希望 2. 「資料作成時の労力を低減化する。」
 - 検討: Excelの関数またはマクロ導入による自動化。
 - * 他部署にて成功事例あり。
 - * 当該部署に取材と協力を希望する。
 - * 小型ロボ③を流用し、省力化を行いたい。
 - 手動確認 1. 担当者確認 → 打合せを依頼、時間を確保する。
 - 事前準備 1. 打合せ前に、依頼内容をPPT作成し、メール添付して送信する。
- 自動化希望 3. 「資料修正時の労力を低減化する。」
 - 検討: 上司から修正の指示があった場合の、対応見直しを行いたい。
 - * 小型ロボ②を流用し、フォルダ移動作業の自動化を行いたい。
 - * 事前にフォルダ名、ファイル名のルール等の整備を行う必要あり。
- 日程: 開催日時 毎週火曜日 13時~
 - 準備期間 金曜日の午後4時に課長宛に資料添付(CC: 全主任)。
 - 手動確認 1. 「人事異動時に宛先変更の確認を行う。」
 - * 異動報がイントラサイトに掲載された場合は、こちらも自動化検討を希望。
 - 手動確認 2. 「事前承認用のメーリングリスト送信前には、必ず目視確認を行う」

最初のメモに比べると、かなり具体的になってきましたね。

この段階で、実行させるために、どういった動きをする部品が必要で、何を設定すればよいかを考えます。どこが重複していて、どのロボでも同じ動きをする箇所はどこかを探しましょう。最小公倍数を見つけるように、注意を払って最適解を見いだすコツをつかんでください。これはどの場面でも必ず行う作業ですので、手順を掴んでしまえば、非常に楽になります。まずは、ここをしっかりと押さえていくことが上達の早道です。

12.9 黄金パターンをご紹介します (Excel 間のデータやり取り、ログイン等に使用)

12.9 黄金パターンをご紹介します (Excel 間のデータやり取り、ログイン等に使用)

1. 入力用のリストと、出力した文言や図や表を格納するフォーマットを準備します。
2. 入力用リストを開き、変数に入力する文字や値を代入します。(変数についての説明は、冒頭の WinActor の紹介欄を参照ください。何らかの器、容器、ケースが変数。例えば本章の最初の方では、お財布みたいなものが変数とお話ししました。)
3. ロボットに入力欄を押させる。
4. 変数に格納された文字や値の貼り付けを行う。(※入力するだけならここで終了です)
5. 出力された文字や値や図を、変数に格納する。
6. あらかじめ準備した出力フォーマットを開く。
7. 変数の中身を所定の場所に押して、貼り付ける。
8. フォーマットを閉じるか、別のファイルに貼り付ける等の作業を行う。(ファイルの開閉等よく使う部品は、あらかじめ RPA ツール側で準備されています。)

さらに、確実に入力できたかどうかを確認するために、追加作業によりミスを避ける安心設計を目指しましょう。値の貼り付けを行った後に、変数に入ったままの文字や値と、入力欄に貼り付けた文字や値を比較する仕組みを作り、この 2 つが一致していれば○、していないければ×等を別の欄を設けて記入します。ロボット作業後に人間がこの列を確認し、失敗あれば手でリカバリします。クラウド +PC は、専用線 + 専用システムより劣化した環境です。ミスはどうしても起こりやすくなります。ロボットと協働して良いお仕事をするためには、常に人の目で確認し、エラーやミスが出た際には手動で修正することを心がけてください。

転記	業務種類	業務内容	単位 作業時間	人数	回数	PC作業率	効率化 予想時間
○	会議資料 作成	資料収集	60分	3	4	90%	13H
×		作成時間	120分	1	4	85%	14H
○		承認時間	15分	1	4	100%	1H

図 12.12: 確実に入力できたかどうかを確認

ここからは、文字ではなく画面を対象にロボットの動きを追っていきます。箇条書きリ

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

ストを片手に、実際のPC画面をキャプチャする作業に入ります。省略せず、キャプチャ画像を全部集めていきます。Excelにどんどん貼り付けていきましょう。

可能であれば、全画面をコピーして紙に打ち出し、その束の各ページに、ロボットが押す等の操作を行う場所を赤丸で囲んでください。入力する場所には、何を入力するかを青ペンで記入します。あるいはExcelにそれぞれの画面を貼り付けていき、赤丸を適宜配置、青文字で注意書きや入力内容を書き添える、でも結構です。実は、この画像集は、これだけでも簡単な設計図にもなりえます。紙は便利です。私たちは千年以上、紙と一緒に仕事をしてきました。これからも必要な場面で適切に使いましょう。

さて、集めた画像を並べてみてください。どこで分岐しているか、繰り返しているかが判別しやすくなるかと思います。最初に作るロボットは簡単なもの、分岐も繰り返しもない部品です。その部品を使って、後々で分岐と繰り返しを加えて実務に耐えるものに仕上げていきましょう。ここでは、どの箇所が核になっているかを見つけ、部品の候補をいくつか仮決めしておき、担当業務の箇条書きリストの添付資料としましょう。

ものすごく忙しくて、設計図をお願いしても出してもらえない部署のロボットを、この方法で作成したことがあります。忙しい部署ほど自動化は必要なので、導入後に、とても感謝されました。手順と画面がわかれば、なんとかなります。本当に忙しい部署には、フロー図でなくても紙束でOK！として差し上げてくださいね。

なお、原則として「ロボットには、判断ができない」という問題があります。ですので、判断が必要なものは自動化作業に向きません。思い出してください、皆様が業務を引き継がれるときも「これは、どうしますか？」と、ひっきりなしに聞いてくる後輩がおられたかと存じます。しかし、それは、長くは続きませんでしたよね？

その理由は、同じ作業を何回も繰り返すことによって、判断力が身についたからではないでしょうか。ロボットや自動化には、通常、繰り返し作業の実行で判断力が上がる機能はありません。ですから、判断が必要な作業については、人間が行う必要があることを意識なさってください。判断が必要な個所までをロボット、判断は人間、判断後の繰り返しはロボット、のよう協働から始めましょう。完全に自動化するためには、ロボットが判定できるような工夫が必要になります。具体的には、判断の目安やものさし、確認表などを作成してテストを行う必要があります。まずは単純に、ゆくゆくは複雑にとお考え下さいね。

12.10 作業を見直します。

12.10 作業を見直します。



図 12.13: 作業を見直す

このとき大切なことは、最初は分岐しないコースを想定し、まっすぐなレールを敷くことです（下図 Step1 の状態）。最初は分岐も繰り返しもない、最短コースの直線的な、必ず成功する例を、まず、レールの上に作ることが大切です。まずはシンプルにしっかりと形を整えます。バリエーションは後から付け足します。

かといって、さきほどリスト上で作った分岐や繰り返しの箇条書きが無駄になるわけではありません。まっすぐなレールで最短コースがうまく動くようになってから、分岐を加え、繰り返し作業を追加していきます。まずは、設計やテストでもこの短いまっすぐなレールのコースがしっかりと動くように何回も練り直します。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

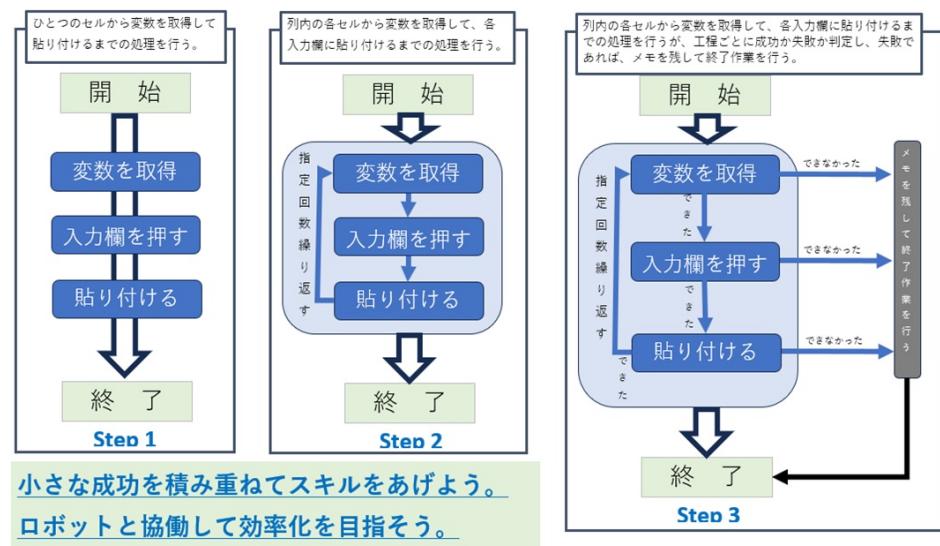


図 12.14: 作業の流れ (Step.1~3)

それでは、ロボット/自動化ツールを3つ作る気持ちで作成しますよ。作成順も優先順位を気にしてつけてまいりましょう。ここで作成するウチの子ロボは、後で横展開をする場合の素体となりますので、種類の違う自動化を選んでくださいね。さきほど、費用対効果の順番と、手伝ってもらえると助かる順番をつけてくださったかと思います。今度は、それら上位の子たちにダブりがないかをチェックします。ダブってしまった子は、最初に作る子の素体を使えますので、次回作にまわしてあげてください。

ここでは、複数のExcelをまたいだ処理、クラウドから資料をダウンロードするもの、備品やファイルの管理表など、どの作業でも必ずといっていいほどお世話になる機能の塊をリストと突き合わせて見ていきます。機能のまとまり、塊で分けて、小さなロボットを作ります。ひとつの塊として分けることができる、役目ごとのちいさなロボットを作成していくことにいたしましょう。ちいさなロボット、略してちいロボさんですね。ちいロボさんは、素体用の、お役立ち&愛されロボを目指しますよ。

確認です。よろしいですか、ロボットは単純なことしかできません。複雑なことには向いていません。そうですね、あなたの仕事を引き継ぐ後輩が、アホな子だと仮定してください。ただしめっちゃスピード早くで正確。根がアホなんだけど、正確かつやることがとても早い、濃ゆいキャラの子です。そして笑顔がとても可愛い。そういう子には、どう指導してあげれば良いですか？

ダメですよ、藁人形と白装束を準備しては。金槌とろうそくも買ってきました？ それは災害用の備えにまわしましょう。いや、すごい行動力ですね。さすが、お仕事が早い。素晴らしい

12.10 作業を見直します。

らしい。いや本当に。ま、それはそれとして、よろしいですか？ 仮定のお話ですからね、イメージですからね、ホントにそういう後輩がいるわけじゃないんですよ。落ち着きましょう。はい、ここで深呼吸です。

それでは、もう一度、お尋ねしますよ。一度にたくさんのお仕事ができない、単純な作業なら早く正確にできるタイプの子には、どのようにお仕事を指示していけばよい関係を築けますか？ そうです、各作業ができるだけ細分化、単純化して、型にキッチリはめてあげればよいのですよね。では、それをちいロボさんにもあてはめて、作業を進めて参りましょう。

前作業として、それぞれのちいロボさんになる予定の塊に手を入れます。自動化する予定の箇条書きの作業を、今度はさらに細かく、部品化していきます。本当に単純なパートに分解していきます。これをタブか字下げを使って、箇条書きで並べましょう。行間は多めにとってください。一番細かいと思ったパートでも、さらに細かくなることもあります。余白大切。

この時に大切なことは「現在の業務で使っているアプリ/ファイルは最大限残す」ことです、たとえ RPA/自動化の作成ツールに専用の代替部品が存在していたとしても、その機能に特化した専用アプリと代替絹の作成ツール部品では、処理速度や性能が異なります。これまで専用アプリで行っていた作業に関しては、そのすべてを自動化ツールの中で置き換えるのではなく、そのアプリが得意なことはアプリに任せることをおすすめします。

この考え方を軸にしていただき、ロボット/自動化ツールには制御を中心にして組み立て担当として働いてもらうことにして、作業の肝となる必要な各アプリの間を”繋ぐ”構成としましょう。こうしておけば、ロボット/自動化ツールが突然、変更になったとしても、最小限の置き換えですみますね。このあたりの備えも大切です。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。



図 12.15: 自動化は単に手段であって目的ではない

12.11 (そのまま乗せ換えることが正解とは限りませんよ？)

というよりも、現行のシステムをそのまま乗せることは難しいです。安からう悪からうではないですが、安価であるということは、それなりの理由があるわけです。全部が今より新しくて素晴らしいそれでいて安いとすれば、それは夢の製品ですが、実現化には今よりも数多くの技術革新が必要となるでしょう。今は手が届かない、未来に発売される製品ですね。

我々は、上司が決めたツールでちいロボさんを作つてお仕事を助けてもらいましょう。ゆくゆくは、社内中で様々なかいロボさんたちの活躍がみられるはずです。そのためには、さきほど取り組んでいただいたように、「単純なことしかできないけれど、早くて正確」であることを生かして、現在ある作業を、どう変形・合体させたら効率化できるかを考えてみてはいかがでしょうか。

12.12 分解して再構成する。

我がロボに、一点の ヒューマンエラーなし!

図 12.16: 我がロボに一点のヒューマンエラーなし!

これはナイショのお話ですが、私のお師匠は「"ロボットには、こうしないと業務が載せられないんです" という言い訳を生かせ。これまで上司先輩から邪魔されてきた、ずっと進めたかった効率化があるだろうから、それを今行え」と教えてくださいました。現場の人間が一番、業務に関して理解している。だから、これじゃなきゃ自動化できないんだというのを言い訳にして、この機会によくわからない癖に横やり入れたがる上司先輩を排除して、やりたくてもできなかった効率化に、こっそり挑んでしまえ、と、それは思い切り悪い笑顔で進めてくださったのですわ。

もちろん、上司先輩からの指摘が正しい場合はあてはまりません。ロボットにかこつけた修正の提案は、あくまで理不尽に妨害を受けて来た場合に限ります。それに、効率化の目途がきちんとたてていることも重要です。そうでなければ、自分の首を絞めるだけになりますので、その点も理にかなったものであることは必須です。

それでも私は、助言をくださったお師匠を尊敬しておりますよ。はい、心から。ちなみにこれは、くれぐれも誠に本当に絶対になにがなんでもナイショですけどね。

12.12 分解して再構成する。

各作業が最適であるかを検討してまいりますよ。プラレールを思い出してくださいね。開始から終了まで、まずはまっすぐのレールを引きます。次に分岐のふたまた等のパーツを準備して、カーブを組み合わせたり段差をつけたりして別のルートを作ります。電車はうまく走れそうですか？ イメージの中では、ちいロボさんが運転手だったりすると楽しめますね。それがうまく走ったら、今度は繰り返しの塊、ぐるぐる立体交差をつけたして、全体のコースイメージを整えます。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

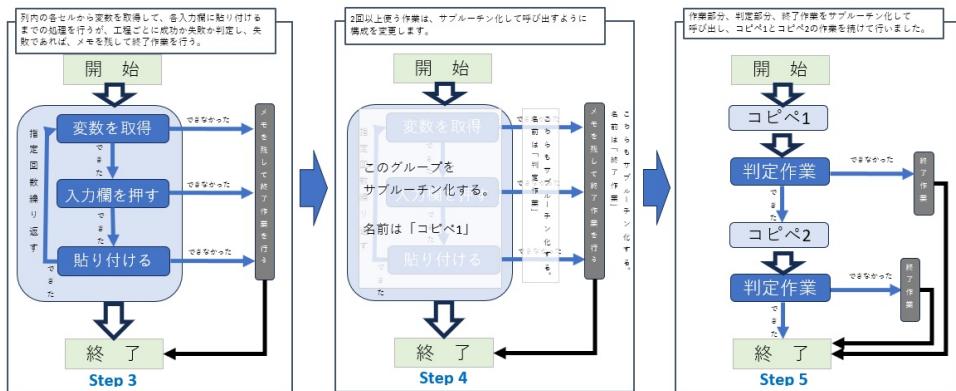


図 12.17: 作業の流れ (Step.3~5)

お気づきになりましたか？ ロボット/自動化は、つまりプログラム化するということの芯というか軸になる箇所は、プラレールのように、ブロック遊びのように、パーツをつないで遊ぶこととそな変わりません。その延長線にあるものです。パーツを選んで、組み合わせて、自分が望む形・動きを目指していくような、皆さんのが子供の頃に遊んだあの経験が生きてきますから、安心して自動化に進んでくださいね。

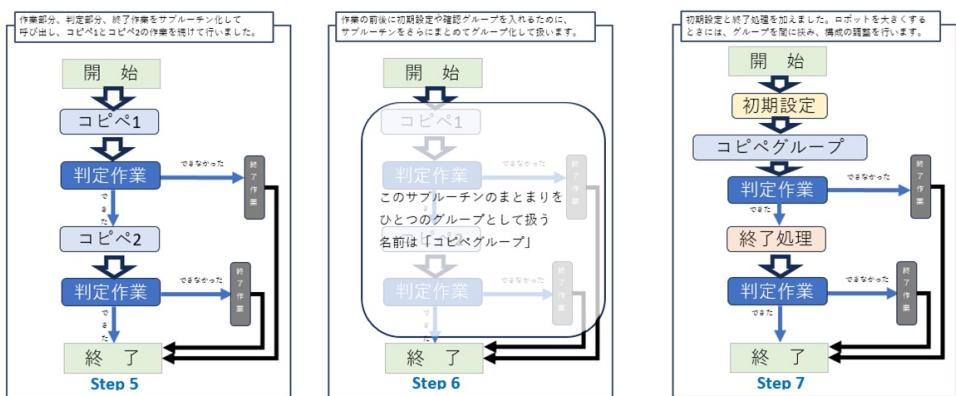


図 12.18: 作業の流れ (Step.5~7)

12.12.1 メモの追記・追加タイム

さあ、追加メモを作成してみましょう。以下の表は、サンプルですよ。

12.13 分担を決める。

■ 会議資料の作成 その 4 → 小型ロボ展開メモ

- 目的: 会議で使う資料を作成する。→ 作成した小型ロボを他の業務の効率化に使用したい。
- 作成した小型ロボについて：
 - 小型ロボ 1. 受信メールから添付資料を探し、解凍して指定のフォルダに格納する。
 - 小型ロボ 2. 指定のフォルダ内の各ファイルを開いて、必要部分をコピーする。
 - 小型ロボ 3. 資料ファイルにあらかじめ設定した、関数 / マクロを動かす。
- 小型ロボの追加作成を考える。
 - 作成済み業務案件リストから、流用できる場面を見つける。
 - 追加作成する小型ロボについて考える。
 - 追加ロボも、作成した小型ロボ 3 種を流用して作る。

ちょっと先の自分が、ちょっと前の自分をもっと好きになれるように、ちゃんとメモを残してあげましょう。自分大切。自分大好き。褒められるの好き。役に立つの好き。このうちふたつは、ロボットも人間もきっと同じなのではと思うことがあります。

12.13 分担を決める。

ロボットと協業する内容とスケジュールを決めます。お時間のこともそうですが、内容も分けていきます。なにせ、ちいロボさんなので、100% 全部、ひとりでこなせるわけではありません。作業の分担を決めていきましょう。特に、最初のうちは、人間の目で必ず確認作業を行ってあげてください。

ここでひとつ、覚えておいていただきたいことがあります。ロボットは必ず命じられたことを行います。命じられていないことは行えません。だから、ロボットを動作させて、その結果が希望通りではなかった、あるいはまちがっていたとしたら、あなたがどこかで間違った命令を出しているということです。

悪いのは、ロボットではありません。なぜならロボットは自分で判断できません。命じられたことを行っているだけです。だから、うまくいかなかったときには、意地をはらずに、ロボットにごめんなさいをして、部品と設定や手順を見直しましょう。ご存じのように、努力は決してあなたを裏切りません。が、間違った努力は失敗につながります。もちろん失敗が次の成功への土台となることもあります。ロボットも、努力と同じように扱ってあげてください。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

12.13.1 メモの追記・追加タイム

さあ、追加メモを作成してみましょう。以下の表は、サンプルですよ。

■ 会議資料の作成 その5 → 小型ロボとのタイムスケジュールを考える
！ エラーは必ず発生すると考える。発生時は手動に切り替えて作業を完遂する。！
PCで一般回線を使用するロボット/自動化は、専用機で専用回線を使う従来のシステムに比較し、安定性に勝るとは言い難いことを意識する。
動作スケジュールを考えてみる。

- 管理ツール導入前と導入後で使用できる時間帯が異なる
 - 導入前: エラー発生に備え、基本的に、定時内に行う。
 - * エラー発生時に、即時手動に切り替え、フォローを行うため。
 - * ロボット故障時には、手動操作に切り替えて遅滞なく作業を行うことが大切。
 - 導入後: 深夜零時（サーバ停止の1時間前）迄使用可能とする。早朝6時からOK。
 - * エラー発時のフォローもロボットが行えるよう調整する。
 - * エラー発時には、ログを残して停止する。
 - * 停止した際は、動作をエラー発生前までに戻し、動かす前の状態に戻す。
- 通勤時: 離席の際（昼食時、会議時）に、ローカル上で動作させるのはOK。（課長確認済）
 - ディスプレイを切っておく必要あり。
 - セキュリティとの兼ね合いがあるが、スクリーンは外しておく。
 - マウスを少しづつ動かしてスクリーンが出ないよう専用小型ロボが必要。

処理しなければならない情報の量は、日々、増加していきます。来月にも、そのメモは、ちょっと先にいる自分への大事な宝物に変わります。小さなことでも書き加えていきましょう。ちょっとしたことを書くことや、書いた内容を恥ずかしいと思うのは、ずっと先、ずっと大人になったら、その頃にお願いします。

12.14 ツールを使い自動化/ロボット組み立てを行います。

自転車に乗れるようになるまでは、皆さんも何回も転ばれたかと思います。ロボットを動かせるようになるまでには、どのRPA/自動化ツールを選ばれたとしても、やはり、何回も転ばれるかと思います。私も、専門用語で「こけ」つ、ツールの組み立て方を覚える

12.14 ツールを使い自動化/ロボット組み立てを行います。

のに「まろび」つ、悪戦苦闘しながら身に着けました。数十年ぶりに単語帳を使って用語を覚え、「いまさら受験生かよ」とのお褒めの言葉を周囲から頂戴し「人間、死ぬまで勉強です」と笑顔でお応えしたものですね。

そういううちに、さらに設計でも派手に転び、組み立てですりむき、エラー対策で青あざをいくつもこさえました。ですが、それも最初のうちだけです。ちいさくとも、ひとつつのロボットが動くようになる頃には、転びにくくなります。転んでも痛くなりません。そういううちに、転ばなくなっています。だから、きっと、だいじょうぶです。安心して、前進していきましょう。匍匐前進でもOKです。前へ前へ前へ。



簡単にできる、と 誰でもできるは違います



簡単に作成できるようになりました、ということは「以前に比較して」簡単になっただけで、「誰にでも作れる」程、簡単になったわけではありません。RPA/DXの「以前」は、プロフェッショナルなプログラミングです。プロフェッショナルなプログラミングが、アマチュアのプログラミングレベルに簡単になっただけです。ですから、まったく知識がない向きには、簡単に理解できるとは限りません。作成にあたっては、適性のある方でも、それなりの努力と時間を必要とします。例えはですね、ゲーム作成用のアプリがありますね。サンプル作品を作る時は選択肢を選び続けるだけで、ひとつのゲームが作成できます。ですが、通常は思い描いた物との乖離、落差があります。それは絵で、あったり音楽であったり構成であったり、想定してはいなかったスキルが必要だったということにそこで気が付きます。自動化作成用のアプリも同様に簡単な選択肢から選び続ければ思い通りのアプリが作成できるとはお考えにならないでください。この点を十分に留意いただき、計画や作成をお進めくださいますようお願いします。

図 12.19: 簡単にできる、と誰でもできるは違います

さて、組み立てのコツは、というよりも、お仕事のコツになりますかね？ それは得意パターンを持ち込んで、落とし込むことです。必勝パターンを編み出して、それで大枠を作っておいて、調整を繰り返して動きを良くしていく。これが私のロボット/自動化作成パターンです。

パターンを作っておくことで、RPAツール変更時の対応も楽になります。現実に組み立てる際には、それぞれのRPAツールにある程度習熟する必要がありますが、基本パターンに沿って、必要な順番で学ぶことで、効率の良い覚え方ができます。

ただし、固有の部品や必要な構成というものは、各RPAツールで異なりますので注意が必要です。RPAツール変更時には、どの部品の組み合わせで自分の得意パターンが再現できるか、部品の性質や機能を見極めて、ツールにあわせた黄金パターン再組立てをお試しくださいね。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

それでは次に、成功例、私が使ってきた具体的な構成の例を申し上げますね。

12.15 普及型ローコード RPA ツールに有効な共通の構成

WinActor、UiPath、PowerAutomate/Desktop)、それらを使って自動化を組み立てる際に、共通して有効な方法、考え方、構成があります。部品が多く使われる中規模以上向けになります。先々でお役立てくださいませ。もちろん初号機から使っていただくとさらに良きかなでございます。というわけで、おすすめは次の通りです。

12.15.1 ロボット内に、雛形とするグループ構成および、その中にに入る定型部品を格納

1. 初期設定のグループ 定数および変数読み込み、時刻設定等
2. ファイル処理のグループ 存在確認、読み出し、書き込み等
3. 処理本体（大きさによって変わる/入れ子構造で複数入る）
4. サブルーチン群
5. 終了処理のグループ

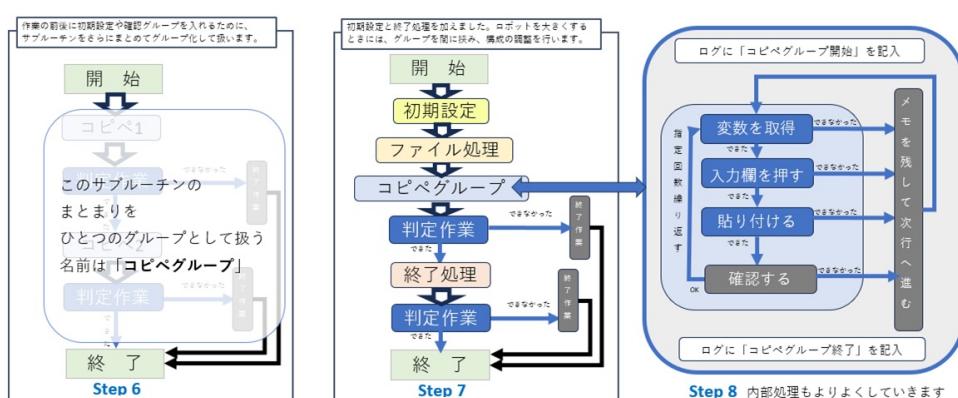


図 12.20: 作業の流れ (Step.6~8)

- 据付のログだけに頼らずに、専用ログも準備する。

作業が進行中、グループやサブルーチンの出入時に一行メモ（グループ名と、入/出、あるいは開始/終了等）を書き込むためのログを別途、作成します。組み立てを行う際に、どこまで進んだがエラーが発生したかを確認できます。（ロボットは内部速度が早いので、実際のエラー地点よりちょっと前に対応すべき問題が存在しますよ。）

12.16 Excel はお好きでしょう？ もう少し話しましょう。

- 2回以上繰り返す作業は、サブルーチンとして切り出す。

切り出したサブルーチンは本体から、都度、呼び出して使うようにします。とはいえたが、最初はどこからどこまでをサブルーチンにするかを判断するのは難しいです。そこで、全部作ってしまって、ここがダブっているな、という箇所を抜き出します。これをグループ化して、サブルーチンとして作成します。

サブルーチンとして切り出すことで、前後にある処理の重複が避けられます。全体のサイズ（部品数）も縮められますし、重複する箇所の分だけエラー発生も防げます。（エラーは必ず出るものなので、あらかじめエラー対策の準備をしておくことはとても重要なのです。できるだけあちこちで手を打っていきましょう。）

12.15.2 ロボットとその資料を入れておく定型のファイル構成を決めておく

- ① 資料置き場 設計図、マニュアル格納用フォルダ
 - 上司説明や引継ぎ時に使用するものを格納しておく
 - そのロボット／自動化に関する情報はすべてここに収める
- ② 作業用フォルダ群
 - 仮置き用 稼働する際の作業時に一時的に格納しておく場所（作業内容により複雑化する場合は複数必要な場合あり）
 - 転記元資料配置場所 所定の場所から情報をコピーし一時保管
 - 転記先資料配置場所 必要情報をペーストし所定の場所へ移動する
 - なお、初期設定と狩猟処理で両方とも空にする、フォルダに残さない
- ③ テンプレート格納用 表のひな形置き場
 - 必要情報をまとめるためにテンプレート（枠組み）を作成しておく
 - 保存せず中間処理に使用する表のひな形を数推作成しておく
- ④ 構築作業時の作業用 ロボットの部品置き場 / 終了時に削除する
 - ロボット組み立て時に途中作成の部品を預かっておく場所
 - 素体組み立て時にも、流用できる部品が見つかったら仮に保存しておく
 - 数種類の部品で比較実験するときなどに候補を待機させておく等に使用

12.16 Excel はお好きでしょう？ もう少し話しましょう。

組み立てる際に、Excel の助けを借りると自動化のハードルが下がります。Excel のブックに変数表や、よく使う文言、行事の日付、本支店リストと連絡先をまとめておくと便利です。詳細はまた別の機会にお話ししますが、ほんのさわりだけお伝えしますね。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

Excelには、連続したセル範囲や表等の範囲を指定して「名前」をつけて扱うことができます。もちろん、ひとつのセルにも名前がつけられます。これを応用してロボット/自動化を楽に作成する、横展開に備える等を行うことができます。

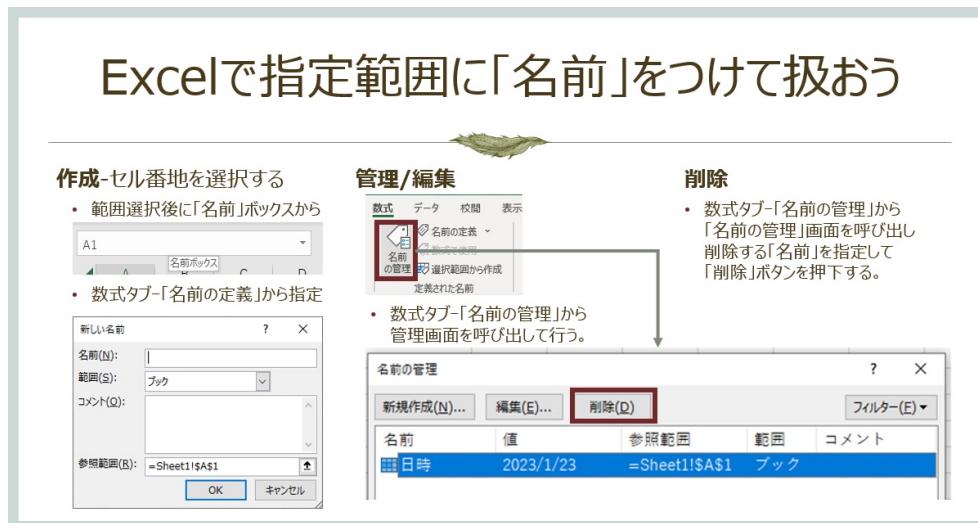


図 12.21: Excel の名前つけ

Excelのブックを一つ眺めていただき、そこに先ほどの「名前」の機能を用いて、変数の一覧、フォルダのURL一覧、ファイルの名前一覧、よく使う社内行事の情報や、上期下期の異動、定期的な会議や社内行事、緊急連絡先リスト、よく使う文例、フォーマット、テンプレート等をまとめておき、ロボットから引用しやすいように整えましょう。さらに関数やマクロを組み合わせると、強力なお助けツールとなります。

12.16 Excel はお好きでしょう？ もう少し話しましょう。



図 12.22: Excel でお助け帳を

例えば、あるシートに CSV リストを貼り付けると、別のシートの指定欄に自動的に張り付ける、といった仕組みは比較的簡単な関数で行うことができます。貼り付けた指定欄に、さらに関数を埋め込んでおくことで、更に様々な働きが見込めますね。

この Excel ブックは、部内で共有する、あるいはコピーして横展開することで部内の Excel が得意でない方にも使っていただけます、ロボット作成者を増やすことへむけてのハードルを低くすることができます。最終的には部内の複数名、欲を言えば誰でもが簡単なロボットを修正することができるようになることを当面の目標となさってください。簡単なロボットを作成し、必要に応じて変更し、壊れたら修理ができる、そのようなチームができれば、これまでできなかった仕事を目指せます。

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。



図 12.23: 作りはじめましょう

Excelのみならず、Word や PowerPoint でも同様に、横展開できるテンプレートやフォーム、お助けツールを作成することで業務の省力化を図れます。こちらもあわせてご検討ください。

次のシートは、お助け Excel ツールのサンプルです。ロボット内には、専用部品が設けられていますが、それを使って作成するよりも、Excel の関数を使って作成するほうがハードルが低いです。もし、RPA 作成ツールが変更になったとしても、この Excel は新しいツールからでも読み込めますので、一度作成しておけば、その点でも安心です。

12.16 Excel はお好きでしょう？ もう少し話しましょう。

A	B	C	D	E	F
名前	数式欄セル書式（日付）	セル書式 (日付/0埋め)	B列の数式内容	説明	新山某謹製 備考
1 今日の日付	2023/1/22	2023/01/22	=TODAY()	本日の日付。	基準なのでB2セルは変更しない
2 今日の日付_和暦	令和5年1月22日	-	=今日の日付_シリアル値	セル書式の日付-和暦から設定。	
3 今年の日付_シリアル値	44948	2023/01/22	=TODAY()	本日の日付のシリアル値。	
4 今年_西暦	2023	1905/07/15	=YEAR(今日の日付_シリアル値)	シリアル値から年を取り出す。	数式の設定上、B列のセル書式を標準に
5 今月	1	1900/01/01	=MONTH(今日の日付_シリアル値)	シリアル値から月を取り出す。	
6 今日	22	1900/01/22	=DAY(今日の日付_シリアル値)	シリアル値から日を取り出す。	
7 来月の今日	2023/2/22	2023/02/22	=EDATE(今日の日付_シリアル値,1)	西暦_来月の本日のシリアル値。	セル書式を日付に戻しています。
8 来月の今日_暦日	令和5年2月22日水曜日	2023/02/22	=EDATE(今日の日付_シリアル値,1)	和暦_来月の本日のシリアル値。	セル書式をユーザー定義で変更している。
9 今月初日の日付	2023/1/1	2023/01/01	=EOMONTH(今日の日付,-1)+1	今月初日の日付	EMONTH関数は元の日付から求める日
10 今月末日の日付	2023/1/31	2023/01/31	=EOMONTH(今日の日付,0)	今月末日の日付	に使用する。()の中は、まず基準となるセル書式または名前、次に当月0から
11 先月初日の日付	2022/12/1	2022/12/01	=EOMONTH(今日の日付,-2)+1	先月初日の日付	1月を求めるのか、1や+1などの整数で
12 先月末日の日付	2022/12/31	2022/12/31	=EOMONTH(今日の日付,-1)	先月末日の日付	
13 今年度初日の日付	2022/4/1	2022/04/01	-	B列手入力	
14 今年度末日の日付	2023/3/31	2023/03/31	-	B列手入力	よく使う日付を保存しておき、必要ある。
15 今年度末日の日付	2022/9/30	2022/09/30	-	B列手入力	
16 今年度前期最終日の日付	2023/3/31	2023/03/31	-	B列手入力	
17 今年度後期最終日の日付	2023/3/31	2023/03/31	-	B列手入力	
18 対象日の入力欄	1999/7/14	1999/07/14	-	B列手入力	取得する年月日を入力する。
19 対象日のシリアル値	36355	1999/07/14	=対象日の入力欄	対象日のシリアル値。	
20 表示形式の変更	平成11年7月14日水曜日	1999/07/14	=対象日の入力欄	対象日の和暦表示	
21 西暦_年	1999	1905/06/21	=YEAR(対象日のシリアル値)	シリアル値から年を取り出す。	上記年月日から求める形式で表示を行
22 和暦_年	平成11年	1999/07/14	=対象日のシリアル値	シリアル値から年を取り出す。	
23 和暦のみ	平成	1999/07/14	=対象日のシリアル値	シリアル値から年を取り出す。	
24 月を取り出す	7	1900/01/07	=MONTH(対象日のシリアル値)	シリアル値から月を取り出す。	
25 日を取り出す	14	1900/01/14	=DAY(対象日のシリアル値)	シリアル値から日を取り出す。	

図 12.24: Excel でお助け帳サンプル

さて、後日、このあたりを含めて Excel を DX 推進の心強い味方にする方法について、詳しくご説明いたしますので、どうぞお楽しみに。



図 12.25: やっぱり心にノートを持とう

第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。

12.17 終わりに これからはロボットと二人三脚で

種明かしをしましょう。どうして、「普通のOLなら業務に役立つロボットを必ず作れます」とタイトルに書いたのか。デジタル化もRPAも、そしてCX,DXさえも昭和の御代から受け継がれ、使い古された「改善」のお色直しです。それをきちんと立たせて動かして、一緒に働いていくためには何が一番必要ですか？

言うまでもなく現場の知恵ですね。今々に足りないものがあり、あるいは間違っているものがあり、または無駄なものがあることに気づくことが改善の始まりで、それは現場に居合わせなければ気が付かないもの。現場こそが、DX/RPAの鍵を握っています。実務、実作業を行っている皆様こそが、上司の誰よりも、本当のお仕事を、お仕事の現実を知っている。そのことは、自動化/ロボット作成作業の半身です。それがわかっていればこそ、改善ができ、自動化/ロボット作成作業が生きてきます。外注に出したすぐに錆付き壊れてしまうロボットではなく、改修しながらずっと生きて、仕事を助けてくれるもの、あなたの代わりに単純な繰り返し作業を行い、あなたに「ずっとやりたかった仕事に取り組む」時間を与えてくれるものです。

私が一番最初に作成したロボットは、2018年にコンサルさんに相談しながら作成したWinActorのロボットです。このロボットは、私がその現場を去った後に改修を重ね、さらにそのロボットをベースに横展開され、今でも働き続けています。面白いでしょう？自分が作ったロボットに兄弟や子供や孫ができる、その子たちが社会の一端を支えているのです。考えるだけでワクワクしませんか？

次は、あなた様の番でございますよ？ いけませんよ、他人事みたいな顔をされちゃあ。明日といわず、今日から、小さなロボットを、ちいロボ君を作り始めましょう。さあ、たったひとつのちいさなロボットから、大きな世界征服をはじめましょう（違）

See you Next Book!

■コラム: DXはスマールスタートからはじめましょうね

12.17 終わりに これからはロボットと二人三脚で

 **DXとは**

DXリテラシー標準 ver.1.0 経済産業省

DXの定義*

企業が**ビジネス環境の激しい変化に対応し、データとデジタル技術を活用して、顧客や社会のニーズを基に、製品やサービス、ビジネスモデルを変革するとともに、業務そのものや、組織、プロセス、企業文化・風土を変革し、競争上の優位性を確立すること**

ポイント

データとデジタル技術を活用して	… デジタルツールの導入 = DXではなく、データやデジタル技術はあくまで変革のための手段
製品やサービス、ビジネスモデルを変革するとともに、業務そのものや、組織、プロセス、企業文化・風土を変革し	… デジタルを使った製品やサービスを提供するだけでなく、データやデジタル技術を活用したプロセスの改善や、デジタルを活用しやすい組織づくりへの取り組みが必要
ビジネス環境の激しい変化に対応し／競争上の優位性を確立する	… 環境変化の中でも、企業が市場で淘汰されずに、成長し続けることが目的

*出所：経済産業省「デジタルトランスフォーメーションを推進するためのガイドライン（DX推進ガイドライン）Ver1.0」（平成30年12月）

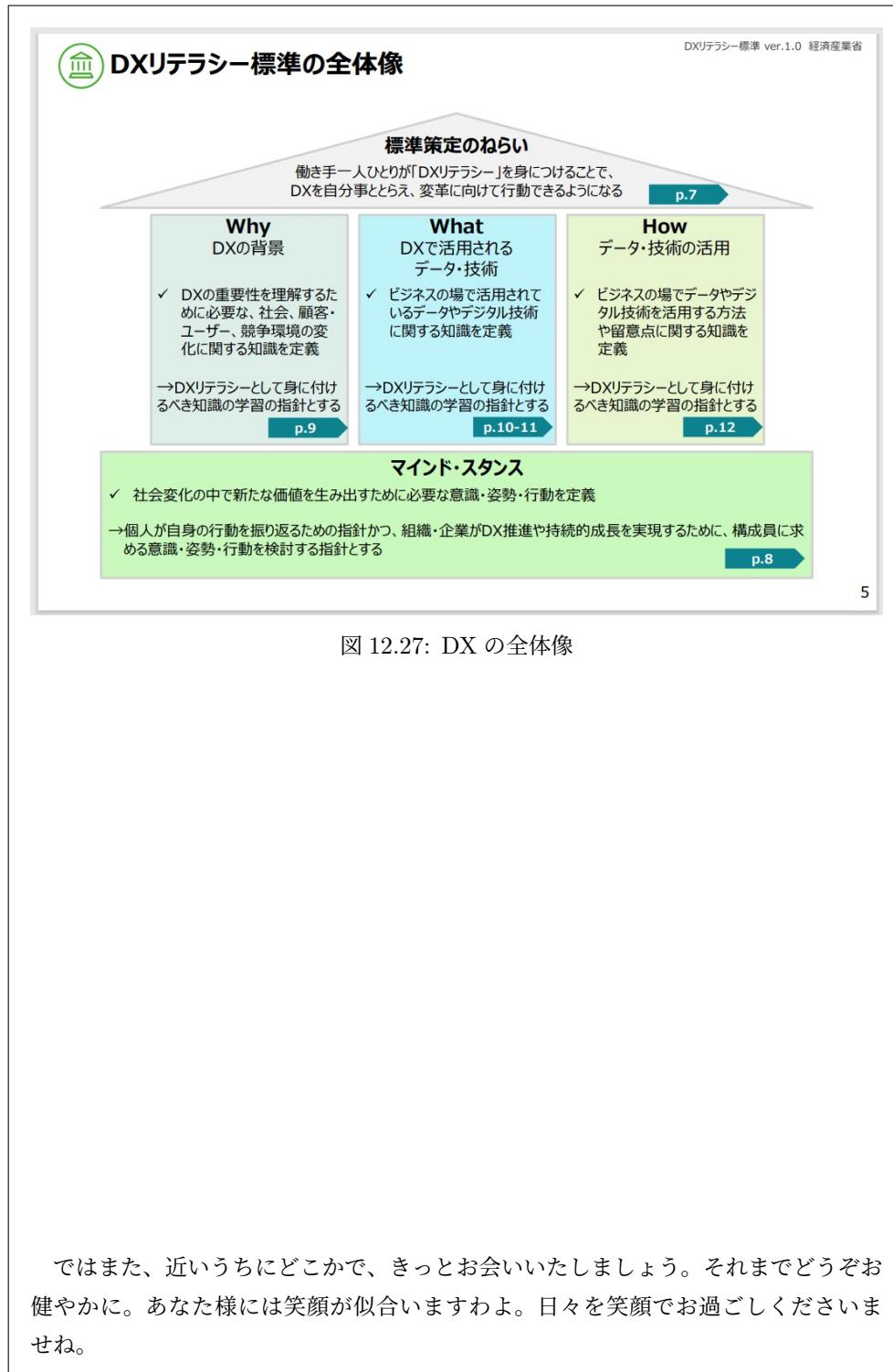
図 12.26: DX とは

出典: DX リテラシー標準 ver.1.0 経済産業省
https://www.meti.go.jp/policy/it_policy/jinzai/skill_standard/DX_Literacy_standard_ver1.pdf

経済産業省の DX リテラシー標準によりますと、DX の定義とは「企業がビジネス環境の激しい変化に対応し、データとデジタル技術を活用して、顧客や社会のニーズを基に、製品 やサービス、ビジネスモデルを変革するとともに、業務そのものや、組織、プロセス、企業文化・風土を変革し、競争上の優位性を確立すること」だそうです。勇ましいわ。なので、ロボットを作つて効率化するだけでは DX ではないみたいですね。

あらあら、先は長そうですわね。ですが、何事も「千里の道も一歩より」なのですよ。私たちはちいロボさんを作つて作業を効率化し、その先にある何かを目指して手に手を取り合つて協力して進んでまいりましょう。この一歩は小さいけれど、DX にとっては、きっと偉大な一歩ですわ。

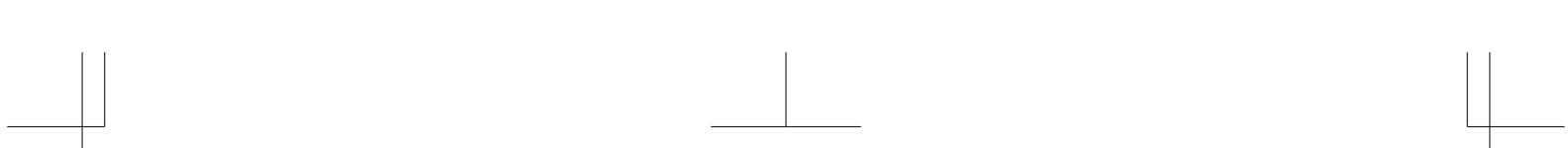
第12章 普通のOLなら業務に役立つロボットを必ず作れます。もちろんこれは体験談です。



12.17 終わりに これからはロボットと二人三脚で



図 12.28: さいごに



筆者紹介

北崎 恵凡 (きたさき あやちか)

趣味でモノづくり活動やコミュニティ「野良ハック」や技術書の執筆や月刊 I/O(工学社)、シェルスクリプトマガジン(USP 出版)などへの寄稿を行う。共著書に「Jetson Nano 超入門」(ソーテック社)、「現場で使える！ Google Apps Script レシピ集」(インプレス R&D)、「図解と実践で現場で使える Grafana」(インプレス R&D)、「はじめての Node-RED MCU Edition」(工学社)がある。コミュニティではオンライン・オフライン・ハイブリッドイベントの企画・運営・配信・アーカイブ編集/管理支援を担当。

山田 雄一 (やまだ ゆういち)

SIer での受諾開発 SE、社内 SE(内製型) を経て、現在はサポートエンジニア。技術同人誌を中心とした技術書の執筆や、たま～に月刊 I/O(工学社) のライターだったりもする。(主に近畿圏近郊のイベントレポートを担当) 共著書に「エンジニアのための見積もり実践入門」、「エンジニアのためのオンライン生活ガイドブック」、「エンジニアのための目標設定の技術」、「エンジニアのための「カジュアル面談のトリセツ」、「積み基板を作らないための電子工作入門」などがある。(全て ditflame 名義(株)インプレス刊)

鎌田 誠 (かまだ まこと)

地元工場の情シスを 20 年経験した後、現在はその経験を活かして、名古屋を拠点にネットワーク系の技術営業に従事。お客様の課題解決等を行う。最近はコミュニティ活動に積極的に取り組んでおり、RPA Community ではライトニングトーク支部の主催。 Babylon.js 勉強会では「Babylon.js レシピ集 vol.1」(インプレス R&D) 及び「Babylon.js レシピ集 vol.2」の執筆も一部担当。IT 知識は広く浅くをモットーに日々精進。

付録 筆者紹介

瀧川 大樹（たきがわ だいき）

学生時代は社会情報学を専攻、推薦アルゴリズムについて研究。強調フィルタリングとコンテンツベースフィルタリングを組み合わせた独自のアルゴリズムを考案、LAMPで実装。また、研究の傍らデータセンターのアルバイトでサーバーの保守運用業務を経験。2014年より通信会社にてサーバー/アプリケーション保守運用業務に従事。保守運用業務に注力する傍ら、情報処理安全確保支援士の資格を取得し、システムのセキュリティー管理業務にも従事。

青木 敬樹（あおき ひろき）

学生時代は機械工学を専攻。三次元画像の特微量抽出に関する研究。2021年に某通信会社に入社。入社後はサーバー・インフラの保守運用業務を担当。社内の「プロアクティブな運用」の推進チームの一員として活動し、運用可能な自動化を実現するために日々勉強中。

大野 泰歩（おおの やすほ）

学生時代は機械学習の中でも強化学習という分野について研究。RoboCup 2D Soccerに触れておりました。2020年に通信会社に入社し、サーバ・インフラの運用保守業務に従事する傍ら、運用保守業務を自動化するためのシステム開発にも関わっておりました。

松岡 光隆（まつおか みつたか）

株式会社コミュカル 代表取締役 CEO。元 IT エンジニア、複数の IT 系 Web メディアでライター・編集業も経験。エンジニア時代に、ユーザー側の立場として 200 回以上のコミュニティイベントへの参加、50 回以上の登壇を経験。その経験をベースとした独自のコミュニティ運営論を展開し 2021 年に株式会社コミュカルを起業。2023 年現在では 20 以上のコミュニティを同時に企画・運営する、コミュニティのスペシャリスト。同時に約 20 社のコミュニティを支援しながら、常に様々なコミュニティへの参加を継続しており、年間約 200 回コミュニティイベントに参加、年間約 100 回のコミュニティイベントを主催運営、年間 10 回以上の登壇を続けている。

新山 実奈子 (にいやま みなこ)

OL 31 年生、RPA 6 年生。職歴はロボットアニメ背景作画に始まり、事務ロボット作成に至る。主に WinActor、UiPath、Power Automate (Desktop) で業務自動化のお手伝いをしています。導入支援から開発、納入、教育までお役に立ちますよ。かくし芸として html を読んだり書いたりできる旧 CIW Website Designer です。モットーは、"Quo Sera, Sera"。

高井 美佑 (たかい みゆう)

株式会社ソントレーズ所属。RPACommunity ライトニングトーク支部 / Power Automate 支部 主催。元・事務職の Power Platform エンジニアです。Power Platform の教育・開発支援・コミュニティ活動支援に従事しています。モットーは、「楽しく学んで、よりラクで明るい未来を作る」。

小崎 肇 (こさき はじめ)

フリーランス。COBOL から始まった 30 年以上のエンジニア経験を経て、Excel-VBA のノウハウと共に UiPath を使った業務効率化プロジェクトに参画。RPACommunity での登壇をキッカケに、運営側としても活動。定年退職後も UiPath 開発者として日々精進。RPA 界隈を賑やかしている中では、最高齢ではないかと呟いています。座右の銘は「人間万事塞翁が馬」。

澁谷 匠 (しぶや たくみ)

株式会社 ASAHI Accounting Robot 研究所 テクニカルエバンジェリスト。一般社団法人中小企業個人情報セキュリティー推進協会認定 DX アドバイザースペシャリスト。プログラミング知識ゼロの状態から RPA/AI-OCR の技術・知識及び概念、全社展開の導入スキーム等を独学で勉強。モットーは "Make everyone's work easier. (全ての人の仕事を楽にする)"。

現場で使える!自動化入門

2023年7月31日 初版第1刷 発行

著 者 北崎 恵凡、山田 雄一、鎌田 誠、瀧川 大樹、青木 敬樹、大野 泰歩、松岡 光
隆、新山 実奈子、高井 美佑、小崎 肇、濫谷 匠
