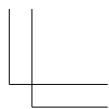
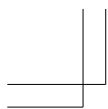


現場で使える!自動化入門

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、百合宮 桜、小崎 肇 著

2023-07-31 版 発行



はじめに

お読みいただきありがとうございます。

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、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。

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たものです。

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。

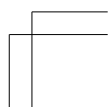
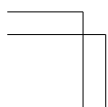
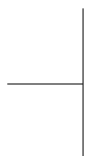
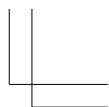
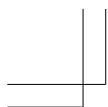
2023 年 7 月

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 (と呼べるかどうか分からないけど 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	色々書いてきましたが…	12
筆者紹介		15
	北崎 恵凡 (きたざき あやちか)	15
	山田 雄一 (やまだ ゆういち)	15
	鎌田 誠 (かまだ まこと)	15
	瀧川 大樹 (たきがわ だいき)	16
	青木 敬樹 (あおき ひろき)	16
	大野 泰歩 (おおの やすほ)	16
	松岡 光隆 (まつおか みつたか)	16
	新山 実奈子 (にいやま みなこ)	16
	百合宮 桜 (ゆりみや さくら)	16
	小崎 肇 (こさき はじめ)	17
	ラムダΛ	17

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 名を超えているのですが、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されております。

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

第 1 章 RPACommunity とは



図 1.1: compass 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。

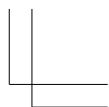
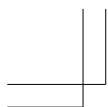
^{*2} <https://www.youtube.com/@RPACommunity>



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、こ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と独断と偏見によるものです

第2章 RPA をいれたり自動化を検討する前に考えたい事

2.2 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×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らない事もありますが、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、そういった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まれます。

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方がいいが、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がちなんですが、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った所の統制を取っ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。やむを得ないのです……

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のはある意味最強の一手です。ハネムーン係数なども改善するかもしれません。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なと考えられます。

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: 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りが嬉しいのか？

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必要の事が多く、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、そういった意味でも良いです。

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、そういう人をお願いするとどうしても単価そのものが高く、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ずとも、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 とかでしっかり目の構成)

さて、コストをがつつりかけて 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ったりする

2.6 自前システム（と呼べるかどうか分からないけど 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もの……

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（と呼べるかどうか分からないけど 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っ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 を使った業務プロセスが回っ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しまいます。

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な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

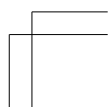
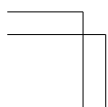
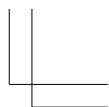
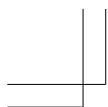
- システムリプレースや改修するほどではないね
- RPA で改善をしようね

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



筆者紹介

北崎 恵凡 (きたざき 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。

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。三次元画像の特徴量抽出に関して研究。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）

XXX

新山 実奈子（にいやま みなこ）

XXX

百合宮 桜（ゆりみや さくら）

XXX

小崎 肇 (こさき はじめ)

XXX

ラムダΛ

XXX

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、百合宮 桜、小崎 肇
