RPA2.0
実現に必要な４つの要素

Robotic Process Automation 2.0

業務プロセスの自働化

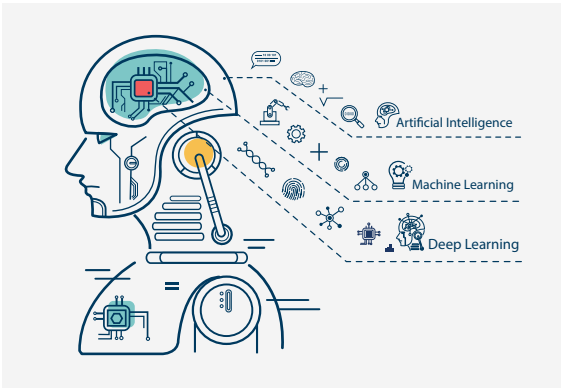
RPA で期待されるソフトウェアロボットが創る未来像は、面倒な仕事はロボットに任せ、人は楽しみながらクリエイティブな仕事に専念できる明るい職場です。人はロボットを監視するか、指示をするか、成果物を受け取るだけで、理想は業務オペレーションのすべてをソフトウェアロボットに任せ、常に会社にいなくても仕事が完了するワークスタイルです。

とはいえ、そこに行きつくまでの道のりはまだまだ長く感じられます。それでも、AI や IoT が発達し、人にしかできないと思われていた判断が必要となる処理も、いろんな分野でコンピュータがやってくれるようになってきました。
中には、膨大すぎるデータからディープラーニングのように自律してデータを習得し、人でも難しかった判断の正確性を増すことができる分野も出てきました。

一方、残念ながら一般的な RPA ツールで実現できる単純作業は、業務の一部しか自動化しません。業務全体を自動化するためには、自動化された各プロセスをさらに連携して、ソフトウェアロボットによる無人運転を実現する必要があります。

無人運転のためには、ソフトウェアロボットによる業務プロセス全体の自働化 (ニンベンの働) が必要です。つまり、動かすだけでなくロボットに働かせるのです。
自働化は、業務経験がない担当者でも、ソフトウェアロボットを管理できるようにする完全自動化の仕組みです。エラーが想定されるあらゆるケースをロジックに組み込み、可能な限り異常終了を無くし、万が一処理が止まってもすぐにリカバリーできるように必要なログを収集し、迅速に通知する仕組みでの開発が必要です。
作業の自動化によって業務の一部を省力化をするだけに留まらず、業務全体を自働化して省人化してこそ、業務担当者はさらに付加価値の高いポジションへ異動できるようになるため、人を活かす RPA による活人化が可能になります。

RPA2.0 で実現可能となるソフトウェアロボットによる業務の自働化は、業務プロセスの連携が大切なポイントになります。



RPA ツールの誤解

ここで一般的に広まった RPA ツールのプログラミングについての誤解を整理しておきます。

RPA が世の中に受け入れられ流行した理由は、IT 部門ではなく、業務部門で業務を自らが自動化できるという期待からでした。エクセルやワードのオフィスソフトのように、プログラミングを知らなくても業務処理が自動化できる、という勘違いからです。
ここであえて勘違いという言葉を使っているのは、あくまでプログラミングなしで簡単にできる部分は、PC 操作などの単純作業のみで業務の一部でしかないからです。人間だからいとも簡単に行なっている操作であっても、ロボットに教えることはかなり面倒な作業です。回線のエラーが出た時や、いつもと違う画面が表示された時など、臨機応変な対応をさせるためには、ある程度のプログラミングの技術が必要です。
プログラミング不要といっても、GUI を使って記述しているだけで、結局、ほとんどの業務は変数やデータ形式など、プログラミング知識がないと作成できません。そして、テンプレートにない処理の場合は、スクリプトという表現で、プログラムコードの記述が必要になります。スクリプトを記述するにも拘わらず「プログラミングなし」と説明されることが問題ないのであれば、Ruby や PHP などのスクリプト言語で作成したプログラムは、プログラミングなしで作成できることになってしまいます。
よって、残念ながらプログラム知識無しで作成できる部分はかなり限られていて、多くの場合、プログラミング経験がない業務担当者には、期待していた自動化はかなりハードルが高く難しいのが現実です。自分達で業務の自動化ができると期待して、RPA ツールを導入した業務部門の多くの人が、現在困惑してしまっているのがこの理由です。

AI の発達のおかげで、ロボットは人間の考えを先行予測して何でもやってくれるだろうという期待に反して、現在の RPA のソフトウェアロボットは、人が無意識に判断して行なっている操作などは、いちいち事前に細かく指定しておかないといけないため、個別に追加のプログラミングが必要となります。一般的な RPA ツールは、自動生成されるワークフローなどを利用して、プログラミングを楽にしているだけです。この部分は、発達した AI を実装したソフトウェアロボットが自らの考えで行動できるようになるまでは、ツール毎の違いは単にプログラミングをどうように簡単に行なうかの手法の違いでしかありません。
事実、RPA ツールによっては、内部的に XML や Java のソースを生成しており、開発を楽にするためのプログラムソースのジェネレーターの一面を持っています。

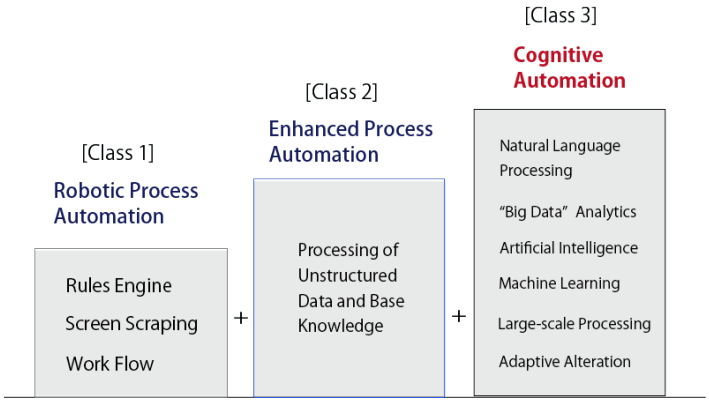
RPA の意味

RPA は、Robotic Process Automation が語源ですが、そもそも、この Process（プロセス）はいったい何を指しているのでしょうか？

プロセスという言葉は、過程や手順を意味しますが、コンピュータの世界では、プログラムによって生成されたオブジェクトの「プログラム実行中のインスタンス」を指します。
ところが、RPA のプロセスで誰もが思い浮かべるのは、業務のビジネスプロセスです。なぜなら、RPA は、ソフトウェアロボットが業務を代行してくれるという意味で、世の中に広まっているからです。ビジネスプロセスとは、つまり業務プロセスのことであり、業務の自動化が RPA に期待されているわけです。
但し、業務も千差万別で幅広いため、簡単に自動化はできません。デジタルレイバーの３つのクラス分けの中に、RPA は第一段階として登場します。その中では、ルールエンジンや、スクリーンスクレイピング、ワークフローなどを利用して自動化する手法として、単純作業に特化したソフトウェアロボットを意味しています。そうした意味においては、多くの RPA ツールはその役目を果たしており、定型業務の自動化で効果を上げている事例も多いです。

ところが、RPA はソフトウェアロボットだという印象が強いため、人々は一般的な業務すべてを RPA で自動化したいと期待しています。それゆえパターン化できる作業のみならず、メールなどの非構造化データや人の判断に頼ってきた業務についても自動化ができる第二段階のクラスを、従来の RPA とは区別するために、RPA2.0 と呼んでいます。
この RPA2.0 では、RPA ツールの機能的な問題で、業務の一部の作業しか自動化できなかった業務についても、各プロセスを連携して、業務プロセス全体を自動化することができます。

ちなみに、クラス３では、人には不可能と思われていた大量データや、高度な計算が必要な処理についても AI の技術を活用して、ソフトウェアロボットが自律的に今まで人にはできなかった業務まで実現します。



[参照元：KPMG Bots in the back office　-The coming wave of digital labor]

自動化が必要な人の操作

人の操作をコンピュータによって自動化することを考えると、業務で使用するプログラムされたアプリケーションは、単に実行さえすれば通常自動的に動くように作られています。その中で、作成されたプログラムの内容によって、そのプログラムの実行指示や、使用するデータを渡すための入力処理、そして実行後に成果物を受け取った後の出力処理を人間が行ないます。

まず、自動化が必要な操作処理は入力処理で、アプリやツールへの実行指示の多くもここに含まれます。入力処理をコンピュータで実行させるためには、業務処理したい目的物を、データ化することが必要です。目で見た映像や、音声などであっても、データ化しないと処理ができません。たとえば、人は、手書きのデータを読み取ってキーボードを使って入力したり、メールで依頼された内容に従って、売上データを入力したりします。
処理の結果行なう出力については、コンピュータ処理されたデータを報告するためにプリントしたり、データを保存したりする行為があります。

そして、人が業務上行なっている操作として、判断するという行為があります。通常、ここが業務経験がある人しかできないような属人化されている場合が多く、それゆえ人の異動ができないため、自動化処理が強く望まれている部分でもあります。
その他、エクセルなどを使って、マニュアル操作でデータを選別したり、加工したり、集計したりする場合があります。また、すでにあるアプリやツールに渡して、データの選別や加工をする場合もあり、その実行をするための一連のオペレーションが必要になる場合もあります。

こうして、人が行なっている操作を大別すると、入力処理、判断、データ連携、出力の処理に分けられます。

人が介入する操作

① データの入力処理
② 判断をする
③ データの選別、整理
④ データの出力処理

RPA2.0 のプログラミング

RPA2.0 の開発と、普通にプログラミング開発する業務用アプリケーションとの違いは、RPA2.0 では業務担当者が行なっている PC 操作などの処理が、通常のロボット化する部分として増えることです。
アプリケーションに API があるものはプログラミングによって簡単に自動連携できますが、通常はエクセルなどのオフィス系ソフトウェアや Web アプリケーションなどとのやり取りがあって、マウスやキーボードの動きに頼る部分が、業務担当者の仕事の多くを占めており、その部分の開発が追加で必要になります。つまり、RPA2.0 でイメージしているのは、業務担当者が行なっているすべての業務に関するオペレーションすべてをソフトウェアロボットに代行させるのです。よって、その業務を行うアプリケーションを開発したプログラムソースがあるのであれば、そのプログラムに PC 操作で行なう UI の部分のプログラムを加えるだけでも効率的に開発できます。
業務の多くは、1 つのアプリケーションのみで成り立っているのではなく、様々なアプリやツールの連動があります。たとえば、メールで受注があったことを確認し、Web からデータをダウンロードして、エクセルで加工して、販売管理システムに登録して、商品手配指示を連絡するといったように、業務担当者の仕事には、いろいろなプロセスが存在します。

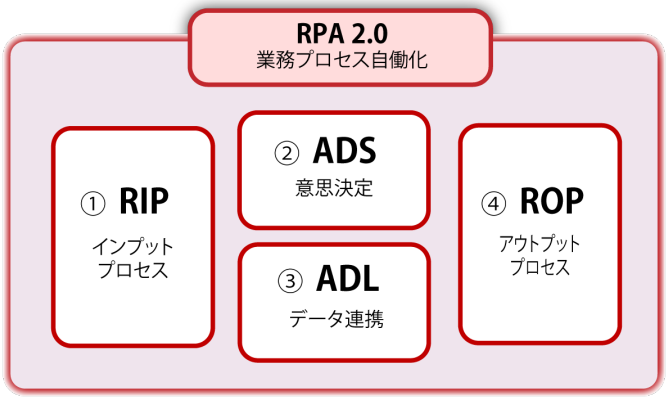
ロボットに任せたい業務は、その担当者毎の PC で 1 台づつのソフトウェアロボットを割り当て、担当者が行なう業務の一連の流れをすべてロボットに引き継ぐことを目標にします。その業務を複数人で行なっている場合、ロボットもその処理を連携して行えるようにプログラミングします。つまり、A さんのロボットの処理が終わったら、B さんのロボットを起動し、データを渡してあげるといった具合です。担当者が、複数業務を行なっている場合は、それだけ、ロボットに指示するプログラムが増えていきますが、まずは人間と同じように少しづつ引き継ぐのがよいでしょう。

RPA2.0 では、想定されるありとあらゆるケースにおいて、人がオペレーションする場合に実施するであろうと思われる動きについて、その処理をロジックとして組み入れます。具体的には、通常ありえない形式でデータが入ってきてしまった場合、人であれば画面上で修正したり、連携したいアプリケーションの画面レイアウトが変更になった場合、別の場所にある OK ボタンを探してボタンを押しているなど、様々な場合を洗い出し、ソフトウェアロボットにあらかじめ指示しておく必要があります。

プロセスの自動化方法

それでは、業務担当者が行なっている作業をどのように自動化したらよいのでしょうか？

それには、もちろん業務フローを作成してプログラムを作るわけですが、先ほどの人が介入する操作の 4 つの処理をプロセスのグループとして、分けて対応方法を考えれば整理しやすくなります。



① RIP ： Robotized Input Process　　インプットプロセスの自動化
② ADS ： Automated Decision Support　　意思決定支援の自動化
③ ADL ： Automated Data Linkage　　データ連携の自動化
④ ROP ： Robotized Output Process　　アウトプットプロセスの自動化

①と④のインプットとアウトプットのプロセスが、Robotized で自動化と表現されているのは、その作業を RPA ツールなどを使って、新たにロボット化すべきプロセスだからです。
それに対し、②と③が Automated と表現されているのは、AI や ETL ツール、既存のアプリなど、業務プロセスの一部として取り込みたい部分がすでに自動化されているからです。
つまり、RPA2.0 では、最初からすべてを新たに作成して開発する必要はありません。すでに取り入れているアプリやツールはもちろん、その業務に最適な AI やパッケージソフトなどがあれば、それを上手く組み込んで最適化します。

そして、一番重要なのは、必ず 4 つを順番に取り込まなければいけないわけではなく、それぞれのプロセスを業務に合わせて、適宜何度も組み入れ、最適化して無人運転ができるように連携することです。

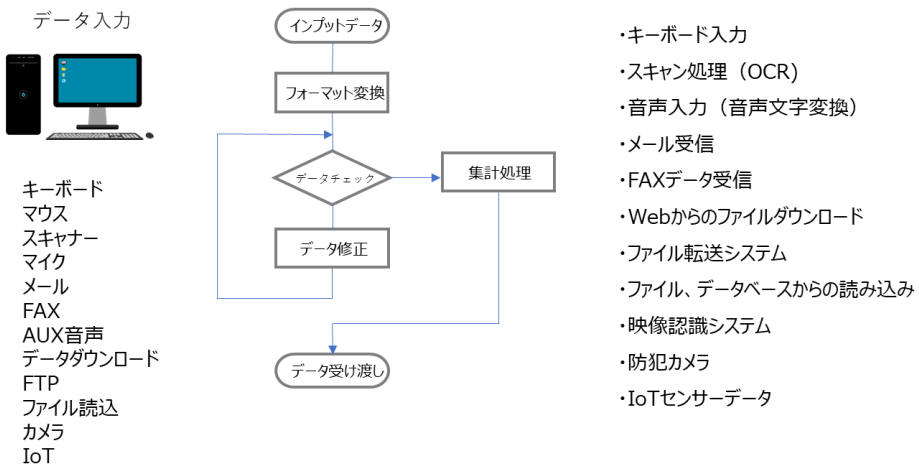
① RIP インプットプロセス

RIP(Robotized Input Process) は、インプットの自動化されたプロセスです。

入力部分の自動化は主に RPA ツールを使用します。というのも、主なオフィス業務として、多くの人がマウスやキーボードを使って、この入力作業に携わって PC 操作をするからです。
メールからの情報を元に、Web のアプリを開いてデータを入力したりする典型的なデータ入力する作業などは、多くの RPA ツールが得意とするところです。
毎日 Web から申し込みのあったデータをダウンロードして、集計処理をする場合なども、ソフトウェアロボットにやり方を教えておけば、自動化できてしまいます。
この時に重要なのは、取り込んだデータを次の処理につなげやすいようにフォーマット変換したり、その内容を可能な限りチェックして、例外データが入ってきたとしても、そのロボットが止まらないようにあらかじめエラー対応のロジックを組み入れたりすることです。
そして、エラーデータの情報もログとして蓄積し、その対処方法を知識ベースに反映していきます。

入力するデバイスによっては、RPA ツールだけではすべてを自動化できません。
たとえば、スキャナーより読み込まれるデータが、手書きの場合は、AI-OCR のようなツールと連動して、テキスト認識できるデータになった状態から突合処理やデータの振り分け処理を組み入れたりします。
インプットプロセスは、簡単に表現すれば、データを受け入れ次の処理に繋げるための前処理で、ほぼすべての業務に共通して必要になり、このプロセスが中心となって、他のプロセスと連携しているともいえます。

【インプットプロセス例】



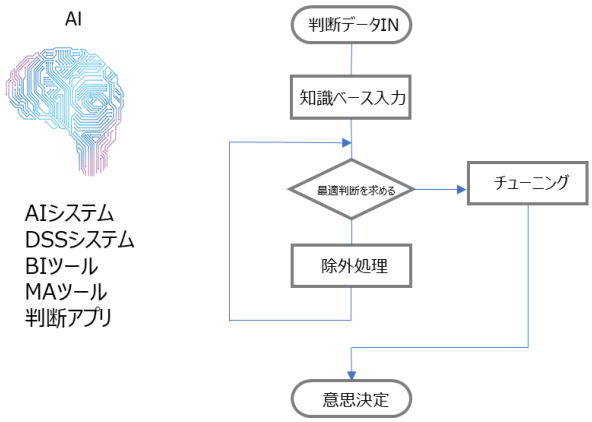
② ADS 意思決定支援

ADS(Automated Decision Support) は、自動化された意思決定支援のプロセスです。

このプロセスは、人が判断していた処理を自動化するために、AI（人工知能）や、DSS（意思決定支援システム）などを利用するプロセスです。
ディープラーニングなどの機械学習が発達したおかげで、AI は自分自身で学習する術を得ました。それにより、今まで人間では到底推測できなかったような複雑な要素を含む判断も大量のデータを元に導き出すことさえ可能になってきました。

大量なデータがなくても、判断材料となる知識ベースがあれば、弱い AI と呼ばれる推論エンジンにより人の経験による判断を、コンピュータに任せれる場合もあります。たとえば、判断する元となるデータさえ揃えば、たとえ承認プロセスのような場合であっても、比較的公平な判断基準で申請事項の可否を自動的に出すことさえできます。
とはいえ、全てを合理的に判断できない事情が日本の会社には多いのも事実です。属人化業務から、経験豊富なベテラン業務担当者を解放してあげるためにも、ソフトウェアロボットに徐々に判断させていく事を増やしていくというのが、働き方改革にとても有効です。今後の人材不足を考慮すれば、AI 投資をして、RPA に AI を取り入れていくことが、ますます重要になっていくでしょう。
RPA ツールを手足に例えると、RPA ツールでカバーしきれなかった頭脳の部分は、AI 等に依存して連携することにより、自動化できる業務範囲がかなり拡大します。

【意思決定支援プロセス例】



・AI-OCR
・推論エンジン
・機械学習
・ディープラーニング
・BI（ビジネス・インテリジェンス）
・BPMS（ビジネスプロセスマネジメント・システム）
・チャットボット
・AIスピーカー
・DSS（デシジョン・サポート・システム）
・MA（マーケティング・オートメーション）

③ ADL データ連携

ADL(Automated Data Linkage) は、自動化されたデータ連携のプロセスです。

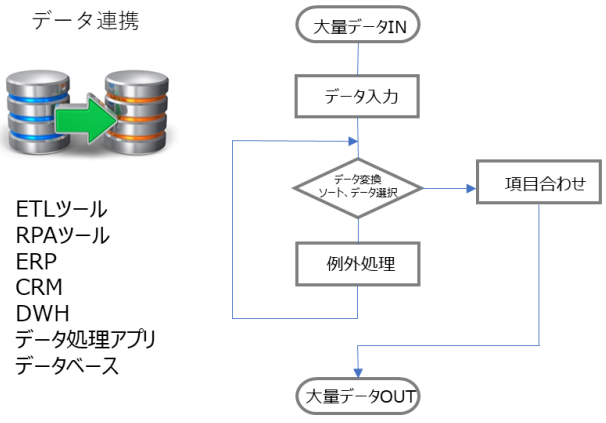
代表的なのは、大量のデータをたとえばエクセルの表からフォーマット変換したり、内容まで加工して連携するデータベースにロードする ETL ツールのようなものです。
大量のデータ処理を RPA のプログラムで新たにコーディングするのは大変です。その点、ETL ツールなどを利用すれば、高速処理が可能で、しかも正確性が増し、メンテナンスも容易になります。コストメリットが期待できる業務であれば、完全無人化を考えた時に外部のアプリケーションやツールを使用することは、大変有効です。

大量データを取り扱わなくても、データを連携したいアプリケーションに API があれば、既存の業務の処理を遂行するためのアプリケーションやシステムの多くは、データを受け渡して実行することで連携が可能になるため、このプロセスに入ります。
つまり、ERP(Enterprise Resource Planning) のようなツールもプログラムからデータ連携できる API があるため、これに該当します。

このデータ連携の自動化プロセスは、インプットやアウトプットのプロセスと密接に連携して、場合によっては、それらのプロセスを補完するばかりでなく、時には代替することさえあります。但し、その多くは、新たにプログラミングしなくても GUI 等で設定するものなので、その前後のデータチェックや、エラー時の処理は別途組み入れることが大切です。

このプロセスでも、業務に最適なツールやアプリケーションを選定することが、安全で高速な処理を実現できる大きなポイントとなります。

【データ連携プロセス例】



・CSV、エクセルデータ処理
・XML変換
・Office File処理
・データーベース
・Cloud データ
・CRM(カスタマー リレーション シップ マネジメント)
・ERP（基幹系情報システム）
・メールシステム
・ファイル転送システム
・DWH（データウェアハウス）
・セキュリティシステム

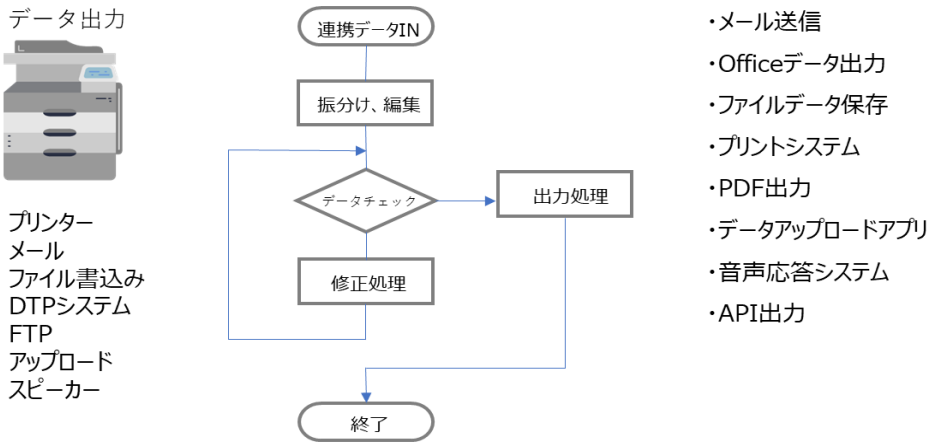
④ ROP　アウトプットプロセス

ROP (Robotized Output Process) は、自動化されたアウトプットプロセスです。

ここもやはり、今まで業務担当者が操作してきた部分が大半なので、RPA ツールなどを中心に開発されます。処理されて出てきた結果をファイル保存したり、プリンターに出力したり、メールに添付して送信したりする処理です。
たとえば、メール送信の場合、添付ファイルは暗号化して、そのパスワードは別メールで送るなど、それぞれの企業のセキュリティポリシーに従い、人が通常行なっているオペレーションはすべて盛り込むことになります。加えて、ここでも想定できるエラー処理をすべてロジックに組むことがポイントです。

アウトプットのプロセスは、多くの場合業務処理によって作成された成果物を出力するプロセスであるため、異なる角度からの複数のデータチェックをかけることが必要です。
通常、最終結果として出力されたアウトプットは、人が判断してからメールで送信等を行なう場合など、人の経験に頼るチェックが多かったのですが、ソフトウェアロボットにチェックさせる場合は、過去のデータを元に、間違ったフォーマットのデータや、あり得ない数値を検知するなど、成果物の正確性を増すためのチェックをロジックに加えます。その時、チェックするパターンは 1 つだけではなく、他のデータとの整合性をチェックしたり、データの範囲を限定したりなど、2 種類以上のいろいろな角度からダブルチェックをすることがお勧めです。

【アウトプットプロセス例】



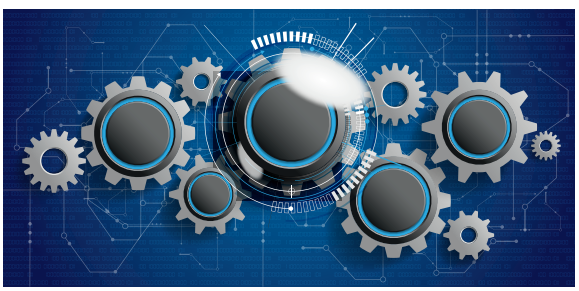
プロセス連携

それぞれ自動化されたプロセスを連携してこそ、一連の業務プロセスが完成します。

事前にプロセスを分けて考える目的は、それぞれの作業目的を明確にして、開発すべきか、既存アプリやツールを使うか、それとも人の介入を必要とするかプロセスのブロックごとに整理するためです。
人の介入が必要と思われるプロセスも、いずれは判断基準となるデータが揃えば、AI や他のツールなどの力を借り、自動化できる可能性もあります。その際、プロセスごとに分かれていれば、一から作り直す必要もなく修正が楽になります。
また、プロセスのブロックが分かれていることにより、並行した開発や、アジャイル方式の開発を適用し易くなり、開発効率が上がります。

各々のプロセス群を一連の業務プロセスとして連携するためには、それぞれにチェックポイントを設け、正常処理されていたかどうかをチェックして、次のプロセスに引き継ぐためのロジックを組み込む必要があります。その際、不具合があったり、予定していた結果を得られない場合などは、必要に応じで、それまでに行なったプロセスの適切なポイントに戻って再実行するケースもあります。
また、業務によっては、後続のプロセスへの影響を考え、強制的に異常終了させた方がよい場合もあります。その際は、終了させた理由がわかるよう、操作ログや、その時点で使用していたデータ内容や、処理に関係したアプリケーションのログなど、可能な限り情報が残るようにしておくことが大切です。

画面レイアウトが変更になっただけでも、ソフトウェアロボットは止まってしまう場合があるので、画面操作を録画するタイプのログ収集ツールと連動しておくことも有効です。
業務フローに従ってその手順をソフトウェアロボットに教えておけば、問題が発生しなければ十分期待通りの動きをします。しかしながら、環境の変化や、処理するデータの品質の違いで、予想外の動きをすることは、ソフトウェアロボットが所詮プログラムによって動くものであるため、必ず考慮しなければいけません。



無人運転を目指す場合、どれだけエラー対処の処理を詳細に組み込めれるかで、運用に耐えられるロボットを作成できるかが決まります。

自働化のための仕組み

RPA2.0 の開発には、自働化（ニンベンの働）の仕組みを取り入れることが重要です。

トヨタの TPS（トヨタ生産方式）で有名な自働化は、産業用ロボットが活躍する自動車などの生産ラインではすでに実現されており、もともとは異常があった時に生産ラインを自動的に止める仕組みです。つまり自働化の産業用ロボットは、手順通りに動くだけではなく、異常時に対応できるように働くレベルまで要求されます。これにより、知らない間に不良品を大量生産することはなくなり、ベテランの業務経験者が四六時中生産ラインのそばにつきっきりでいる必要がなくなりました。
RPA でも同じです。異常時にソフトウェアロボットが自らの判断で対応できるようにしてこそ、働かせることができるのです。そうでなければ、業務担当者は、常にソフトウェアロボットを監視し続け、少しのエラーでも人が介入しなければなりません。
ソフトウェアロボットに適切な指示と教育をしておけば、監視は遠隔地で誰でもできるようになり、異常時にはそのレベルによって、ソフトウェアロボットが自動的に回避、またはリカバリーし、想定外の異常事態には、適切な人へ自動的に通知が行く仕組みを構築することができます。職場の活性化のためにも、人材不足を補うためにも、ベテランの業務経験者でなくてもロボットの監視をできる仕組みをロボットに事前にプログラムすることによって、業務担当者をその業務から解放できれば、人はもっと付加価値の高い仕事に従事できます。

これは、特にベテランに属人化された業務に有効ですが、なかなかプログラミングでカバーするのは難しいのも事実です。現時点でも、エラー時の対応マニュアルがあれば、それを元に開発すればいいのですが、最初からリスクを洗い出すことが難しい場合、時間をかけてエラー対処の知識ベースを増やし、適宜ソフトウェアロボットに教育していくという方法が現実的です。

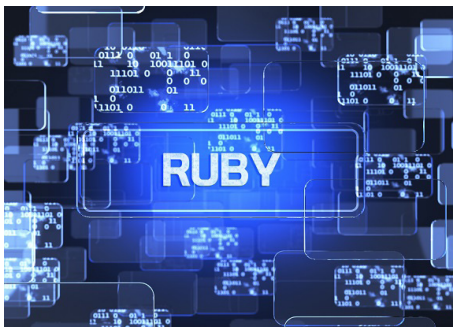
自働化は、ソフトウェアロボットによって、人が面倒な業務から解放され、人が望ましい働き方ができるようになるために、RPA2.0 にはなくてはならない仕組みです。

RPA2.0 の開発に向くツール

RPA2.0 の無人運転を実現させるためには、時にはかなり複雑なあるいはきめ細かな開発が必要になります。

GUI で設定できる RPA ツールによっては、条件式はなるべく減らして、いつ止まってもいいように、常に業務担当者がロボットのそばに付いていることを推奨している場合があります。これでは、担当者は業務から解放されません。
これは、フローチャートのように GUI でビジュアルで分かりやすく設定できるようにしているため、複雑なロジックを組みこもうとすると、画面を何度もスクロールすることになり、その図形の数も膨大になってしまうため、とても分かりづらくなるからです。
つまり、簡単に設定できるようにしたツールは、その分複雑な処理には極めて弱いです。
エラー時には止めてしまうという発想は悪くはないですが、エラーを起こした時の情報を収集できる様になっていないと、人が介入しても、どう対処してよいかわかりません。
それゆえ、設定方法が簡単な GUI 形式で指定できる RPA ツールでは、RPA2.0 の開発に向きません。

たとえば、誰かに外国の人に仕事を頼むとき、わざわざその人の国の言語を覚えて指示をするとしたら、どう感じるでしょう。その言語が簡単に覚えられるからといわれても、できれば日本語で指示したいと思いませんか？
ソフトウェアロボットに指示を与える時も同じです。いくら GUI で簡単とはいっても、そのロボット特有の指示の仕方を覚えるのは、異なる言語を覚えるのと同様に新たに学んでその使用方法を覚える必要があります。それが、複雑な指示であればなおさら面倒です。
一方、たとえば Ruby で開発をしていた人が、そのままのプログラミング言語で指示ができたらどんなに楽でしょうか。ROBOWARE のような開発系の RPA ツールであれば、API を通して Ruby のプログラムから直接ロボットに指示ができます。



しかも、ROBOWARE 用に作成する Ruby のプログラムから、Ruby のライブラリのメソッドを使用したり、AI やそのほかのアプリケーションを呼び出したりもできます。これにより、各プロセスにエラー用のロジックを加えながら、臨機応変に連携するには API の開発型 RPA ツールは最適です。

RPA2.0 で迎える未来

RPA は、単純な作業をソフトウェアロボットにやらせるだけの小さな働き方の変革では終わりません。
RPA2.0 によって、業務全体をソフトウェアロボットに任せて、ロボットを仲間として働いてもらい共に未来に向かっていきます。

かつて、産業用ロボットによってもたらされた大量生産は世の中を明るくし、世の中を豊かにしました。品質も向上して、安全で便利になりました。
RPA2.0 が実現する社会は、オフィスで働く人々をみんなハッピーにします。ソフトウェアロボットの自働化のおかげで、人々は場所や時間に縛られることなく、自分の得意とする好きな仕事に専念できます。
人手不足や、高齢化問題、経験不足による品質低下や、クレームの問題もロボットが解消してくれます。
真の働き方改革は、これからますます発達する AI や IoT を連携した RPA2.0 で実現できます。

RPA2.0 を推進することにより、徐々により多くの業務が自働化できれば、会社の業績も上がり、働く人々も楽になり、そして生活にゆとりができれば、期待したライフワークバランスが実現でき、理想的なロボット社会が訪れることでしょう。

（編集：春日井）

CCS 株式会社シーイーシーカスタマサービス
■お問合せ　ESECinfo@cec-ltd.co.jp
〒 150-0022 東京都渋谷区恵比寿南 1-5-5 JR 恵比寿ビル 8F
TEL：03（5789）2443　FAX：03（5789）2575

（2018年11月14日公開の情報です）