無人運転を実現する
RPA のシステム開発

Robotic Process Automation 2.0

無人運転こそが RPA のゴール

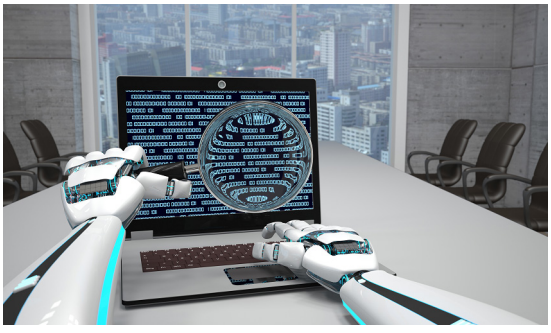
RPA（Robotic Process Automation）は、バックオフィス業務に代表される人手に頼っていた PC 操作等の処理を自動化します。
つまり、ソフトウェアロボットに業務を代行させ、業務担当者の仕事を減らします。

大ブームとなった RPA は、多くの企業で様々な RPA ツールが採用され実践されています。
その結果、ソフトウェアロボットによって業務効率の改善につながったという評価がある半面、所詮 RPA は人手がかかる仕事をロボットで部分的に最適化するツールにすぎない、という声もあります。
多くの RPA を採用した企業は、一部は自動化できたけれど、ロボットの自動化には適さない処理だからと、ベンダー等のアドバイスのまま、人が介入しなければならない作業を残したまま、実際はその業務から実務担当者を解放できないでいる企業が大多数のようです。
果たして、このような状況で満足してよいのでしょうか？

AI 時代に入って、RPA の可能性はまだまだ期待されています。今後更に RPA が活躍するマーケットは拡大していきますが、採用する企業は、どこを目指して RPA を推進しているのでしょうか？

眠らないロボットへの期待は、24 時間 365 日止まらず働き続けられる能力です。
残念ながら、多くの企業はまだまだ中途半端な状態なのに、RPA の実績を妥協して評価してしまっているようです。

RPA2.0 の時代では、RPA への最大の期待が、自働化したい業務の全ての作業をソフトウェアロボットに代行してもらう事です。そうであれば、RPA のゴールは人の介入が要らない完全自動化である無人運転を目指すべきでしょう。



ロボットは眠らない

どんなに働いても、ロボットは眠らないし疲れません。スピーディで、正確で、愚痴も言わないロボットは、夜間でも仕事を依頼できます。
それなのに、構築した RPA のソフトウェアロボットへの指示が十分でないと、業務担当者が介入すべき処理が残り、一緒にそばについていなくてはいけないため、結局、ロボットに時間外労働をさせられないことになってしまいます。

ロボットに仕事をさせるために、業務担当者の残業も増えてしまったとなっては、本末転倒です。ソフトウェアロボットに仕事を任せ、業務担当者は、あまりやることがなかったとしても、会社にいることを強制すれば、残業代が発生してしまいます。

だからこそ、RPA の構築を考える時、無人運転できる仕組みを目指さなければならないのです。

もちろん、業務の目的が、部分的な自働化だけでいい場合もあるかもしれません。また、日中しかできない処理もあるかもしれません。
それでも会社全体では、無人運転の RPA を目指せば、次期システムの更改や、新規業務を立ち上げる時に、ソフトウェアロボットを人の採用と同じように戦力として当てにして事業計画を組むことができ、ロボットが大いに貢献するはずです。
そのためには、段階的に時間をかけてでも、無人運転を目指して RPA の構築を継続し、決してあきらめないことです。

現実には、まだまだ人の判断に頼らなければ無理と思われる業務の一部が残っていたとしても、それは近い将来 AI が解決するかもしれません。
新人に初めから難しい仕事を任せられないのと同様、まずはロボットにできることから始め、将来はベテラン社員と同様のレベルの業務もこなせるように育てていくのです。そうすれば、人はもっと有効で有意義な時間の使い方ができるようになるでしょう。

FA に学ぶべき RPA

無人運転といえば、それを古くから追求してきたフルオートメーションの自動車工場の生産ラインなどに代表される FA（Factory Automation）を思い浮かべるでしょう。FA は、工場の生産工程の自動化を図るシステムの事で、まさしく、人によって行われていた産業活動を機械によって無人化することを意味します。現在も、FA によって産業用ロボットが大活躍して、生活を豊かにしてくれていることは言うまでもありません。

人による生産では、人件費、教育費、労働問題対策などのコストに加え、人的ミス等による品質悪化の問題がありました。よって FA の目的は、生産性の向上やコスト削減、品質向上などが挙げられますが、工場の設備など初期コストが膨大に必要であるからこそ、圧倒的な効率化が必要で、そのためにも無人運転、つまりロボット化が必須でした。

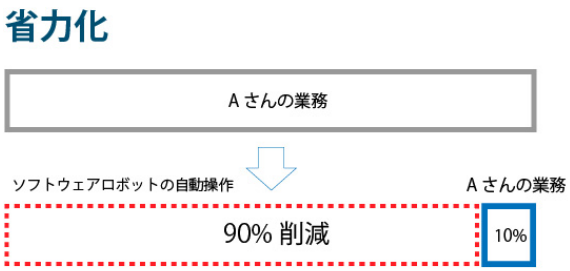
FA では、主にそれぞれの工場の環境に対応した特殊なコンピュータ PLC（プログラマブルロジックコントローラ）が使われており、ソフトウェアで動作する点は RPA に似ています。
PLC のプログラムは、電気回路を記号化したもので、ソフトウェアはラダー・ロジック（ラダー言語）というリレー回路を記号化したものが使われています。現在も工場の生産工程における PLC やロボットの制御では、API を使用したプログラム開発をしています。PLC の代わりに PC に置き換えて考えれば、使用する言語は違いますが、RPA の開発でも参考にできる部分は多いです。

現在の FA を見れば、通信ネットワークを相互に結び、情報システム部門に接続して集中監視、管理をすることで、複数の様々な産業用ロボットが行なう工程を自動化しています。各産業用ロボットは、何でもできる汎用的な人型ロボットとは違い、作業する工程は限定的です。
RPA も、それぞれの業務のフローに合わせて、限定した目的のソフトウェアロボットを量産して、それらをセンター側から指示して連携し、監視を行なえるようになれば、無人運転が可能になります。
FA で参考になるのは、その仕組みだけでなく、無人運転に対するその考え方も大いに参考になります。

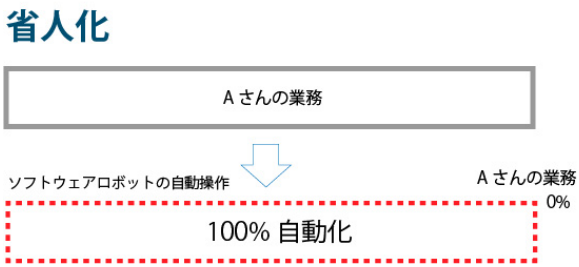
省力化と省人化

FA の分野でよく出てくる言葉に、「省力化」という言葉と「省人化」という言葉があります。
この言葉の違いを説明できますか？

生産ラインを思い浮かべて見てください。
ある作業者 A さんの仕事を、産業用ロボットに置き換えたとします。このロボットは優秀なので、9 割強の割合でロボットによって自動化することに成功しました。これで、A さんの仕事量は格段に減り、圧倒的に「省力化」できました。
しかしながら、見方を変えればたったの 1 割ですが、A さんの作業が残ってしまいました。
それゆえ、A さんは毎日工場へ出勤しなければなりません。もし、ロボットが A さんの仕事をすべて自動化することができれば、その業務のためには工場出勤が不要になり、その時はじめて「省人化」が実現できます。



さて、これを RPA に当てはめて考えてみてください。
A さんの業務も、100% ソフトウェアロボットが代行してくれて、はじめて「省人化」です。
ここで大切なのは、RPA のゴールをしっかり無人運転に置いているかどうかということです。
「省人化」できなければ、在宅勤務も夢で、新規事業の業務への配置転換もままならないかもしれません。



有能な人材を、いつまでもロボットでもできそうな仕事に縛り付けておくのですか？
工場での生産ラインで、他社が「省人化」を実現している場合、9 割自動化できて満足している会社が生き残れるでしょうか？

RPA も同じです。自動化は 9 割できたからといって満足していてよいのでしょうか？
これから続くロボットや AI の時代に、他社に打ち勝つためには、100% の自動化を目指さなければいけないのです。すこしでも妥協していてはその会社は負けてしまいます。

目指すべきは活人化

「省人化」という言葉に、単なる人減らしのように悪いイメージを持ってしまうといけないのでその同義語で使われる「活人化」という言葉を取り上げます。

FA において、「省人化」する目的は「活人化」のためです。生産の現場では、「省人化」が人員整理のリストラをイメージしてしまうので、かわりに「活人化」という言葉を使っているようです。「活人化」は、言うまでもなく人を活かすということで、自動化できる部分は、可能な限り自動化して、もっと人にしかできない仕事で活躍してもらうことを意図しています。

RPA でいえば、経験上その人しかできないと思われていた業務をソフトウェアロボットに代替させることができれば、その人はもっと違った業務で活躍することができます。ベテラン社員を、単純な業務から解放できなければ、会社にとっても損ですし、本人もモチベーションが上がらないでしょう。
就職したら、一生同じような業務に縛られることなく、ロボットも進化していけば、それにも増して人も次のステージに上って活躍していくという理想的な世界です。
そうはいっても、こうした考えに前向きに取り組めない人は、同一労働同一賃金の時代を迎え、ロボットのコストの合わせられて、どんどん給料が減ることでしょう。それにより、ますます格差社会が進んでしまうかもしれません。

しかし、この「活人化」が進めば、時間や場所にとらわれない働き方が実現でき、人の暮らしがもっともっと豊かになるでしょう。きっと、その頃には、ロボットと比較されてしまうので、時間単価で働くという考え方自体がナンセンスになっているかもしれません。

なぜなら、ロボットを基準にした最低賃金では時給数円にしかならず、ロボットでもできる仕事では生活していけなくなるからです。

省人化　＝　活人化

ロボットに業務操作を任せ
負荷価値の高い仕事に就く

ジャスト・イン・タイムの少人化

さらに、事務系の現場では聞きなれない言葉に「少人化」という言葉があります。
有名な TPS（Toyota Production System：トヨタ生産方式）の中の、各工程が必要なものだけを停滞なくスムーズに生産する考え方である「ジャスト・イン・タイム」の５つの基本原則の中に、「少人化」は登場します。
「少人化」は、市場の変化や季節変動等に対応するために、必要とされる量に見合った生産を極力少ない人員構成で生産出来るようにすることで、ピーク時に合わせて労務費が無駄にならないように、臨機応変に対応できる仕組みを作ります。

「省人化」が、ロボットによる自動化で人を減らすのが目的であって、作業のムダは考慮しないのに対し、「少人化」は、生産性を維持しながら最少の人数で作業することを目的にしているため、徹底したムダの排除が行われます。

ここで、この言葉を取り上げた理由は、RPA を実装していく過程で、必ずしもすぐに「省人化」が実現できない業務においては、「少人化」の考えも重要だからです。
自動化してから、その作業工程のムダが発見されるよりは、ムダを省いてから自動化すべきですし、たとえ今の時点では自動化できない部分が残ったとしても、可能な限り少ない介入で済むような工夫が必要です。
また、ピーク時など、業務量の違いを考慮したロボット化も必要です。データ量が増えたら、同時に平行して作業するソフトウェアロボットを稼働できる仕組みまで考慮されていれば、その時の人の介入の負担が軽減できます。ピーク時には、毎回いつも人が作業の応援に集められているようではもったいないです。
そのためには、業務量を判断して、動くロボット数を変動させたり、並行処理できる仕組みであったり、連携やリモート起動できる仕組みなどを構築しておくことが必要です。人であれば、当然臨機応変に対応できる事も、ソフトウェアロボットには様々なケースを予測して事前に準備しておくことが大切です。

自動化と自働化

TPS と言えば、「ジャスト・イン・タイム」とともに 2 本柱として有名なのが、「自働化」です。「自働化」の語源は、自動織機に源流があり、トヨタ自動車の社祖である豊田佐吉が 1924 年に発明した糸切れなどの異常を検知すると自動的に止まる世界初の「無停止杼換式豊田自動織機（G 型自動織機）」などが初期の自働化の例です。

「自動」は、単に動くだけの事ですが、「自働」は機械を管理・監督する作業者の動きを「単なる動き」ではなく、「働き」にすることを意味します。具体的には、異常があれば機械が止まる仕組みによって、ひとりでも何台もの機械を運転することができ、そのまま知らずに不良品が生産され続けることを防ぎ、飛躍的に生産性を向上させることが可能になります。

これを RPA に置き換えて考えてみると、RPA ツールによって作成されたソフトウェアロボットは単に自動的に動くだけではいけません。データクレンジングの処理や、作成されたデータの整合性チェックなどを組み入れておかないと、ロボットが異常終了しなかったからといって、出力されたデータを信頼するしかなく、後で大事に至るという事になりかねません。

さらに、TPS では、自動停止装置を取り付けるだけではなく、「アンドン」と呼ばれる異常表示盤を、各ラインの監督者からよく見える位置に置き、異常発生と同時にアンドンを点滅させ、監督者が即座に対処できるようにする仕組みがあります。
これが、RPA にも重要で、作成したソフトウェアロボットを少人数で監視し、異常があればすぐに対処できる仕組みを作ることが、無人運転の実現には必須条件となります。

ロボットが判断できることは対処方法まで組み込み、規定外のデータが入力されていたり、通常より時間がかかってしまっている場合などの想定外の対処は、異常を検知するとロボットの処理を止めるというロジックを組み込んでおきます。
その異常終了に対して人による対処を行ったら、その経験をナレッジベースとして記録し、プログラムの見直しに活かします。

自動化　≠　自働化

異常を検知すると自動的に止まる
無人運転には　「自働化」が必要

記録型ツールの業務範囲

RPA の記録型ツールを使用すれば、マウスやキーボードなどの PC 操作の動きは簡単に自働化できます。
たとえば、QuickROBO は、様々なアプリケーションの画面操作を、マクロやフローチャートの作成も必要なく、もちろんプログラミングは不要で、主に操作をそのまま記録していけば自動化できます。しかしながら、これでは「省力化」には貢献しますが、「省人化」の実現は厳しいです。エラー処理等ができず、リスクを伴います。単純な文字入力などをキーボード操作や、ダブルクリック、右クリックなどのマウス操作だけであれば、誰でも簡単に自動化できるのでお勧めですが、それでは「自動化」はできても「自働化」にはなりません。

QuickROBO は、簡単なので現場の業務担当者が自分で自動化のジョブを作ることもできます。これは、業務担当者が今まで通りオフィスの机の前に座り、業務の一部のみを QuickROBO に任せるのであれば、かなり有効です。それでも、これは箒で掃く作業を、掃除機でやるのと同様に、ツールで自動的にしているだけで「活人化」にはほど遠く、単に楽になるというレベルかもしれません。

とは言え、現場にシステム的なことを求めるのは無理がありますから、RPA のゴールである無人化を目指すのであれば、システムに精通した方の力を借り、「自働化」できる仕組みを作成することが必要です。そのために、QuickROBO は、ROBOWARE のオプションという位置づけであり、QuickROBO は、ROBOWARE のプログラムからも呼び出せるようになっております。

ROBOWARE のような開発型 RPA ツールは、汎用プログラミング言語で記述できるので、「自働化」のシステム開発ができますが、他の RPA ツールではどうでしょうか？
残念ながら､特に RDA（ロボティック･デスクトップ･オートメーション）中心のツールでは「自動化」は可能ですが、「自働化」まで対応ができるツールが少ないです。
「自働化」を実現できるものは、大規模ユーザ向けのサーバにて集中管理するものが大半で、それ自体がひとまとまりのパッケージ化された自動化システムなので、導入と構築、維持にかなり高額な投資が必要になります。

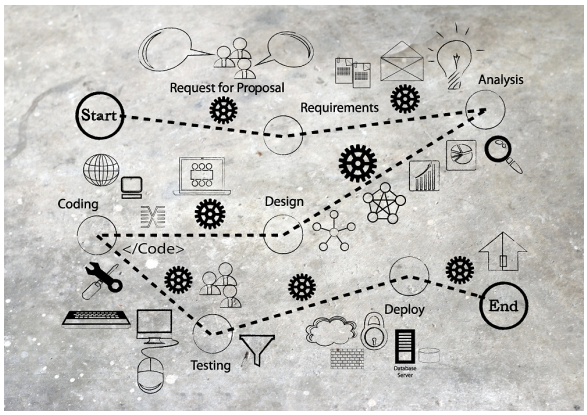
RPA に必要なシステム開発

高額な RPA ツールを大々的に導入し、サーバ集中管理によりロボットを監視、コントロールすれば、ある程度「自働化」に近づくことができます。但し、それは ERP システムなどを導入した時と同じように、そのパッケージされたシステムに業務を合わせ、その独自の運用管理体制を新たに追加しなければならない場合が多いです。
設定や運用方法の習得、そして開発に人員をあまり割けない会社は、RPA ベンダーや外部委託会社に業務の大切な部分を委ねるリスクが生じます。現在、こうした大規模な RPA システムは、トップダウンで導入されることが多く、そうした大がかりなロボット化が、リストラ目的でなく「活人化」が目的であることを願うばかりです。

一方、RDA 機能が中心の小規模な RPA ツールは、無人運転の実現のためには「自働化」の仕組みを開発する必要があります。RPA ツールの機能でカバーできる場合は良いのですが、
多くのツールは、多様なアプリケーションとの連携や、多数の条件分岐などを追加したりすることが苦手です。

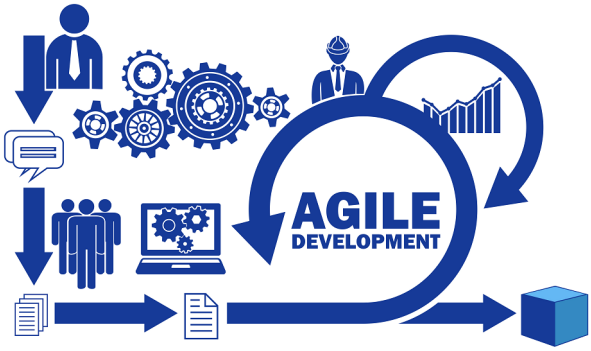
無人運転は、RPA についても「自働化」の手法を十分取り入れ、システム運用に精通した方の意見を取り入れてシステム開発すれば実現できます。
もともと、PC 内でのロボット開発は、OS レベルまでロボット化するために C 言語などを使ってプログラム開発されていました。それでは、流石に一般のプログラム開発にはハードルが高すぎます。しかし、ROBOWARE のような開発型 RPA ツールを使用すれは、Ruby、PHP、Java、C# というような高級言語で、API によって比較的簡単にソフトウェアロボットの開発が可能です。また、すでに他の RPA ツールで作成されたソフトウェアロボットについても ROBOWARE で、スケジュール実行、監視や通知をする仕組みを作成することも可能です。

バックオフィス業務といえども、その業務のシステムを開発した人がいるはずです。その人たちを RPA のプロジェクトに参加してもらうことができれば、その業務を RPA 化する際に最も効率的な無人運転の開発が可能となります。



アジャイル開発と RPA

システム開発の手法に、「アジャイルソフトウェア開発」というものがあります。
よく、ウォーターフォール型開発の手法と比較されて取り上げられますが、RPA には、このアジャイル型の方が向いていると言われています。



アジャイルとは、敏捷、素早いという意味で、ソフトウェア開発においては、エンドユーザの要求を適宜取り入れ、イテレーションと呼ばれる短い期間単位でリリース可能なアプリケーションを提供します。
このイテレーションは、そのまま訳すと反復の意味ですが 1 週間から 2 か月ぐらいを目途に、一般的には 2 週間ほどに設定されます。その期間が終了すると、作業量などを振り返り、それ以降のイテレーションに組み込んでいきます。こうして、短い期間で見直しながら反復のサイクルを継続して開発を進めることで、リスクを軽減できます。

アジャイルの手法が、RPA のシステム開発に向いている理由は、現場の業務担当者の協力が仰ぎやすく、その上こまめに短い期間で最良の設定が確認ができるからです。
ウォーターフォール型のような開発手法では、最初の要件定義や設計の段階で重要なことが決定され、それ以降の開発の途中では、仕様変更などを臨機応変に取り込むことが難しいです。
RPA の場合、多くはすでに人が行なっている業務ですから、現場の担当とフェーズごとに確認しながら開発していった方が、リスクも少なく、スピィーディに開発できます。また、ROBOWARE などのツールを使えば、短期間で開発できるのでイテレーションはさらに短く設定が可能です。

リーン開発と TPS

アジャイル開発手法の中の 1 つに、「リーンソフトウェア開発」という手法があります。
これは、製造業を中心に展開されいるリーン生産方式の考え方、つまりリーン・シンキングをソフトウェアに応用した開発手法です。

リーン生産方式とは、1980 年代にアメリカのマサチューセッツ工科大学が TPS を研究して、体系化した生産管理手法です。
リーン（lean) には、ぜい肉の取れた体が締まった状態という意味もあり、TPS の生産工程のムダの排除をぜい肉に見立てたことが名前の由来のようです。

リーンソフトウェア開発の手法には、7 つの原則があります。

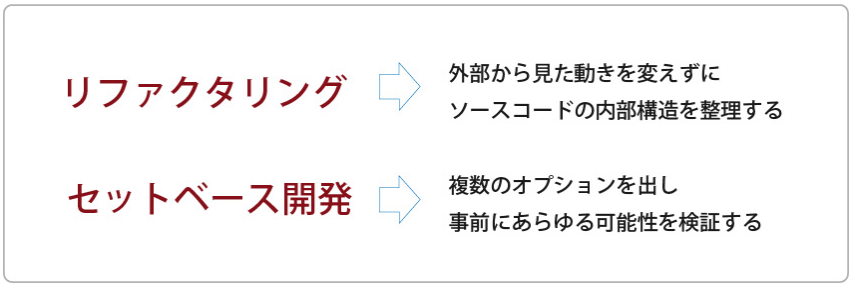
① ムダを排除する
② 学習効果を高める
③ 決定をできるだけ遅らせる
④ できるだけ早く提供する
⑤ チームに権限をあたえる
⑥ 統一性を作り込む
⑦ 全体を見る

［引用元：リーンソフトウエア開発～アジャイル開発を実践する 22 の方法～
メアリー・ポッペンディーク著］

トヨタの生産手法が、製品開発手法へつながった結果が、リーン開発であり、つまり、アジャイル開発手法の根底には、TPS のムダに対する考え方があることがわかります。
だからこそ、無人運転を目指す RPA のシステム開発には、「自動化」だけでは満足せず「自働化」の思想を大切にすることが必要なのです。

RPA 開発における知識の蓄積

リーン開発の中ではたくさんの手法がありますが、たとえば、ラーニングのプロセスは、リファクタリングと統合テストによる短いイテレーションのサイクルを使用することによって、スピードアップすることが可能になります。
リファクタリングとは、外見的な動きを変えず、内部のソースコードを最適化させることです。これにより、プログラムは、理解し易くなり、バグも見つけやすくなり、更にはより早くプログラミングができるようになります。オブジェクト指向の設計思想と深く関係しており、具体的には、メソッドを抽出して、引数オブジェクトを組み入れて、再利用可能な形に変更するなどです。
Ruby などのプログラミング言語では、何度も使用するロジックは外出しして共有するなど当たり前に行なっていることですが、RPA ツールによっては、このリファクタリングが難しい場合もあります。
RPA でもリーン開発では、イテレーションのサイクルを短くするため、開発者が常に業務担当者とコミュニケーションを取り、場合によっては画面を見ながらその場ですぐに修正し、そのノウハウを蓄積していって、次の開発のスピードアップを図ることが大切です。



リーン開発では、時にはセットベースの手法も有効な手段です。セットベースとは、いろいろな可能性を複数シミュレーションできる形で作成し、それを元に最適なものに絞り込んでいく手法です。
RPA では、RPA ツールで比較的簡単にプログラミングできるので、複数のパターンを業務担当者と共有しながら、一番自動化に向いたパターンを作り上げていく方法も可能です。これができれば、業務担当が詳細な業務フローをシステム開発者にうまく伝えられなくても、その動きを共有しながら確認できます。

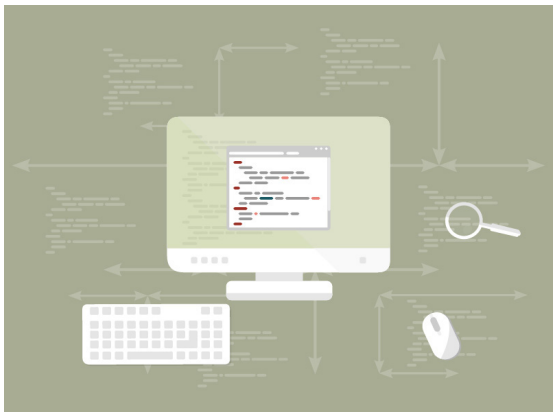
どのような手法を活用してもよいのですが、ポイントとなるのは、ソフトウェアロボットの開発はイテレーションのサイクルの中で、改善を試みながら、知識ベースを蓄積していきます。このノウハウが、その後の開発生産性と正確性を向上させ、新たなロボット開発に役立ちます。

従来のシステム開発との違い

従来のシステム開発では、成果物となる目的のアプリケーションの仕様は決まっていて、その設計仕様に従ってプログラミングされて出来上がったものを、業務担当者が使うというのが一般的でした。
アプリケーションを作成するために、要求事項の収集のヒアリングで業務担当者が呼ばれることがあっても、仕様が決まってしまえば、アプリケーションは、コンピュータシステムが処理する世界なので、業務担当者が開発に携わる必要はほとんどありませんでした。

しかし、RPA は、業務担当者が PC で操作していたその動きが、仕様決定の中心になります。
業務担当自らロボットを作るかどうかは関係なく、開発に深く関わってもらわないと、なかなか思ったようにシステム開発はできません。
しかし、業務担当者には、オブジェクト指向だとか、運用管理はこうあるべきだとか、システムの事はよくわかりません。システム開発者にとっては、システムのことが分かっていない人とシステム開発の話をすることは、相当なストレスとなります。
一方業務担当者も同様で、システム開発者にありがちなロジックを説明されても、自分の経験と勘でやってきた業務をプログラムに落とし込むために説明することは、たいへん難しいことだと実感してしまいます。

その開発プロジェクトのリーダーが、業務にもシステムにも精通した人であれば理想的ですが、実際は、開発担当者と業務担当者とのコミュニケーションが今まで以上に必要になります。
システム開発担当者は、システムの目線でムダを省き、業務担当者は「自働化」の目的を理解してロボット開発できるようにお互いが歩み寄るべきです。
2020 年には小学校でもプログラミングが必須になるわけですから、この機会に業務担当者もプログラミングの考え方に触れて、論理的思考を意識してもらい、システム開発担当者には、これからロボットを量産するためには、業務の現場ではどのような考えや判断でその業務を操作していたのかをお互い目線を変えて学んで、理解し合う姿勢が大切でしょう。



システム開発が必要な RPA

RPA をブームだからそれだけの理由で採用したという企業は少ないと思いますが、ソフトウェアロボットを導入すれば、生産性が上がるはずだと漠然と考えて RPA を始めているのでは、導入したあとに後悔するかもしれません。

RPA を構築するのであれば、働き方改革を見据えて、「活人化」を目標にすべきです。
テレワークの時代になって、時間も場所も超越して仕事が出来るようになった時、AI も発達してロボットに任せられる仕事はどんどん増えていくでしょう。
その時、人は、人間にしかできない付加価値の高い仕事を創造して、それに従事できる環境にしていかなければ、企業は生き残っていけないでしょう。

「活人化」を実践できるのは、無人運転ができる RPA だけです。
そのためには、「自働」のシステム構築が必要で、それを行うために RPA にはシステム開発が必要なのです。そして、その実践には、歴史ある FA で培われた TPS 等のノウハウが RPA のシステム開発手法としてたいへん役に立ちます。

産業用ロボットに比較すれば、RPA のソフトウェアロボットは比較的低コストで量産できます。
無人運転ができる RPA のシステム開発にいち早く着手し、多くのノウハウを貯め、優良なソフトウェアロボットを量産する会社が、多くの「活人化」を実現でき、他社のどこよりも高い生産性を上げることで業績も上がり、間違いなくその業界の勝ち組となっていくでしょう。

（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7月2日公開の情報です）