KnowledgeLake
Intelligent Content Automation

Evaluating a Robotic Process Automation (RPA) Project

10 Factors to Consider When Assessing Project Feasibility And Complexity

Table of Contents

Whenever we engage a new customer, users spend a lot of time in the initial exploratory meeting talking about what their RPA targeted applications functionally do (CRM, ERP, EMR, etc.) rather than what they are. When we say, "What they are," we mean how does the application look to the operating system? If we're talking about a PC environment, which operating systems run in production? Is the application DOS (believe it or not) or Windows-based…16, 32 or 64 bit? Is the application browser-based? If so, which browsers and which versions need to be supported in production? If the system is a legacy application hosted in a terminal emulator, which emulator is being used? Does the emulator have an API for screen element access and remoting? Is the application being served up in a virtual environment like Citrix or RDP? The bottom line is that most Automation Builders (the role of the person who actually builds the bot) don't really care what the RPA involved applications functionally do or what the process is: Those details are the domain of the Process Designer (the person who constructs the project specification) and described in detail in the specification document. Rather, the Automation Builder needs to know which tricks in their bag needs to be used to reliably read screens, write to fields, click buttons, extract data from grids, etc.

The only way for an Automation Builder to fully understand a project's complexity is to spend some time with the applications involved. At KnowledgeLake, we call this step in our methodology "the screen test". We usually require a screen test be performed in order to produce an automation feasibility statement and estimate for a given project. We do this because most users don't know the answers to questions like those posed above. However, often there is not enough time to perform an exhaustive screen test before the customer wants to know, "Can this automation be done?", and "What will it cost?" In those cases, the Automation Builder must rely on gut and experience to render an opinion.

The following 10 factors are the most important points our Automation Builders consider when forced to render opinions and estimates with incomplete information.

1. Which applications are harder or easier to automate?

Which applications are harder or easier to automate? Browser-based and legacy applications (think green screen) are generally easier to automate than traditional Windows-based client applications. This is because, regardless of what the application functionally does, there is significantly less variability between browser applications or legacy applications versus traditional Windows clients. While it is true that client-side scripts executed in the browser or Java-based frameworks may complicate things a bit, most applications running in a specific version of a browser or emulator look the same from the Automation Builder's perspective. Browsers make available several "access points" that allow Automation Builders to read from, and write to, web page screen controls. These access points include: MSAA and UI Automation, the WinAPI, and the Document Object Model (DOM). Likewise, terminal emulators are generally designed with remoting as a priority, so their APIs are usually robust in this area and the "textual" nature of green screens lend themselves to being easily scraped using the WinAPI. On the other hand, traditional Windows-based client applications can have a lot of variability – especially if the developer bypassed commonly used Microsoft Foundation Classes and rolled their own custom UI controls, thus rendering many accessibility techniques ineffective or inconsistent.

2. Will an application require screen OCR (optical character recognition)?

Will an application require screen OCR (optical character recognition)? Screen OCR can be an Automation Builder's best friend or worst enemy. When all operating system screen extraction techniques fail, screen OCR is the last, best hope. When it works, it is a life saver. But used carelessly or in the wrong situations, it can be a nightmare of frustration.

Most software applications share a consistent and common anchor point (upper, left-corner of the window). Resolution and DPI can be easily determined through the operating system and differences from workstation-to-workstation can be derived from a baseline via a simple calculation. Screens are not skewed, speckled or streaked. There may be low color contrast issues, but underlying color codes reported by the operating system are consistent so dropping out colors is generally easy to do. All these issues, which can make paper-based OCR results inconsistent, are not issues for screen OCR. However, screen OCR has its own challenges. For starters, the best DPI rates used on a workstation's display are generally lower than that encountered in low-resolution fax documents. Next, most users have font smoothing turned on which makes it easier for humans to read screens but poses problems for bots trying to OCR those screens. Finally screen font sizes tend to be smaller than fonts on printed forms. While none of these issues are insurmountable, they must be considered when screen OCR is being used. If any of these configuration parameters are gotten wrong, results can be inconsistent and doom a project – especially for attended bot automations. This is especially true if the inconsistent results are on the same field (e.g. sometimes the engine returns a "6" and sometimes it returns a "G" for the same character). If screen OCR must be used, the Automation Builder must build in plenty of extra time for exploration and testing. Also keep in mind the OCR engine may require font "training" to achieve reliable results, and that takes lots of samples and time.

3. Does the automation involve lots of data grids?

Does the automation involve lots of data grids? If the automation needs to integrate with data grids, extra time will be required. While grids are a great UI control for humans because they allow users to quickly view large data sets in summary, they present several challenges for bots trying to read them. For starters, most Windows client applications tend not to handle them well from an accessibility perspective which makes them somewhat opaque to the bot. Further, grids scroll vertically and horizontally which makes it difficult to establish the grid's state (i.e., what part of the grid is currently displayed or has focus on the desktop?). Finally, many grids allow users to re-order and resize columns which also creates a bit of mayhem. While this is usually not a problem when a bot is running unattended because the state of the grid can be reset each time the automation is executed, when the bot is running in attended mode on a user's workstation, things can get tricky.

If a grid is inaccessible through the operating system, the Automation Builder must employ some less reliable techniques to: a) determine which record is selected and where the user is positioned in the grid, both vertically and horizontally; b) acquire data being truncated due to narrow column width settings; and c) select functions available from column-specific context menus. These same issues may also pertain to data tables in the browser, but usually to a lesser extent. The good news is that most applications possess a "details" screen subordinate to the grid that displays the selected row's data in a consistent manner. When available, the details screen should always be used.

4. Do any or all the RPA involved applications run in a virtual environment?

Do any or all the RPA involved applications run in a virtual environment? If all the applications run in a virtual environment, this is not a problem as long as you can run your RPA runtime client within that same virtual desktop. If this is not the case, then the Automation Builder is stuck automating what amounts to application screenshots rather than live screens, which will require screen OCR and lots of keyboard buffer stuffing (i.e. key flow). This also holds true if some of the applications run in the VM while others run on the physical desktop (we call this "crossing the virtual barrier"). If you can run the RPA runtime client tool in both environments, the project will present some challenges but nothing insurmountable. If not, the project gets even more dicey so more time should be allocated.

5. What is the data fidelity between the RPA involved applications?

What is the data fidelity between the RPA involved applications? In a perfect organization, all systems will handle constrained lists the same way. However, this is often not the case. Whether we are dealing with US state names and abbreviations or general ledger codes and descriptions, very often, different applications refer to the same entities in very different ways. If this is the case, the Automation Builder will need to allocate extra time for coding data conversion and translation rules. This may even involve establishing and accessing external arbitration data sources. If so, how will the sources be accessed? Via spreadsheet? Web services? ODBC? Regardless of the data source access method required, the task just got more complex and requires proper planning.

It is also incumbent upon the Process Designer to make sure all systems involved contain a common key(s) that link records together. If this is not the case, then the Automation Builder may have to consider employing "embedded link keys" in unused or partially used fields. Though sometimes necessary, use of embedded link keys should be used with caution.

6. Will the automation involve parsing unstructured data?

Will the automation involve parsing unstructured data? If the automation involves working with unstructured data such as documents and emails, the Automation Builder must build in additional time and get comfortable with a text parsing technology like regular expressions (regex). If the only way to find something in a document (or sometimes on a highly dynamic screen) is to find a fixed, related string, regex becomes an important arrow in the Automation Builder's quiver. However, many find regex a bit arcane and difficult to work with so extra time should be allocated. Build in even more time if you choose to incorporate a cognitive service that can help determine things like transaction type and sentiment. If you go this route, make sure you have plenty of test data.

7. What is the release cycle of the RPA-involved applications and how likely are they subject to changes that may impact the automation?

What is the release cycle of the RPA-involved applications and how likely are they subject to changes that may impact the automation? If the applications change infrequently, the Automation Builder can spend less time on making the automations flexible and durable. However, if the applications change often, especially without a lot of lead time as is frequently the case with auto updates, the Automation Builder should spend considerably more time creating an automation that is more durable and adaptive to changes such as field placement and label changes. Otherwise, when changes occur, the Automation Builder will be under the gun to make the necessary changes with users waiting. One thing we've learned is that, once users get accustomed to bots performing mundane tasks on their behalf, there is no turning back. All work comes to a halt as users wait for the changes to be made.

8. How difficult will it be to get an appropriate amount of test data from the users?

How difficult will it be to get an appropriate amount of test data from the users? In order to properly test an automation, there needs to be sample data made available that tests every rule and condition defined in the specification. Insufficient test data and improper testing is the number one reason why RPA solutions are recalled from production and refactored. It is better for everyone involved if the automation is sufficiently tested before it is migrated into production. Frequent refactoring erodes user confidence and adds significant support costs to the process. Remember, users tend to underestimate the amount of variability within a process—especially when the variability happens infrequently. Having plenty of test data available will help unearth these varying conditions.

One of our best practices is requiring the Process Designer (or a user who does not normally perform the manual task) to enter ten test transactions into the system(s) using the specification as the guide. Entering ten test transactions usually flushes out 80% of the variability that causes an automation to be recalled from production. Also, the person entering these test transactions should record the session because that video artifact can be very helpful to the Automation Builder during the build process.

9. Where will development and testing take place?

Where will development and testing take place? If development and testing is to be performed on live production systems, the Automation Builder needs to factor in additional time to handle transaction rollback and data clean-up. Further, development and testing in production systems tends to slow the process down as Automation Builder's strive not to break anything and are forced to deal with maintenance cycles. If development and testing are being performed in a test environment, it is critical the test environment utilizes the same versions and configurations the applications use in the production environment. Slight differences between the environments will result in recalls and refactoring.

Also keep in mind that the duration of the transaction being automated will also impact testing time. Longer running transactions require longer testing cycles. We have seen long-running transaction automation add 30% - 50% time to normal development cycles.

10. The larger the number of users and varying user types, the more time the Automation Architect should allocate for variance testing.

The larger the number of users and varying user types, the more time the Automation Builder should allocate for variance testing. As the name implies, variance testing is the process by which Automation Builder and Production Managers (the people who help test and manage the runtime environment) seek, detect, and accommodate the variances an automation will encounter from workstation to workstation in production.

Though applicable more so to attended automations running on the user's workstation, a good project methodology should always incorporate variance testing as its own discreet step regardless of the bot mode. The rule of thumb here is the more workstations an automation runs on, the greater the chance of encountering: a) differing performance speeds; b) differing display configurations; c) conflicts with non-RPA involved software; and d) application behavior variances due to different access privileges (e.g., managers may encounter different screens, prompts, and messages than regular users).

While this is not an exhaustive list of all the factors judicious Automation Builders should consider when determining how complex a project may be, it is a great place to start. These points demonstrate that, although RPA solutions can be developed and implemented faster than traditional applications or integrations, they are not as easy as some vendors would like customers to believe. A successful deployment requires serious thought and careful planning. Further, Automation Builders need to be flexible and creative. In our experience, RPA is as much art as it is science. There are no user guides or technical manuals one can read to determine the best way to build a given automation. A lot of it is based on feel, intuition, and a willingness to embrace the kludge when necessary. Remember: By definition, an RPA automation "IS" a custom application.

KnowledgeLake

Intelligent Content Automation

www.knowledgelake.com
888-898-0555