CHAPTER

6

Solution Design

During conceptual design, requirements (functional and nonfunctional) are gathered. Templates are provided in the Appendix, and clear explanations are detailed in this chapter.

Conceptual design can be seen as the solution blueprint. A high-level determination will be made of the automation solution to include what the Bot will do, the systems it must access, the schedule on which it must run, etc. This will be further detailed, but in this phase, the basics will be determined.

Also during this phase, a solution design document is created. This document includes process and outcome, overall structure, logical components and interrelationships of both the current and possible future states. We provide a template, with a clear explanation (please see Appendix 12).

RPA can be introduced with either a conventional "Waterfall" approach or with a more Agile methodology. In the "Waterfall" approach, requirements and conceptual solution design are created and documented before development begins. More and more companies are adopting an Agile or concurrent engineering approach, where requirements and solution development activities are ongoing in parallel. Each of these approaches has pros and cons, and some organizations adapt them to a hybrid approach.

Regardless of the methodology selected, the tasks described within this chapter must be completed.

© Robert Fantina, Andriy Storozhuk, Kamal Goyal 2022
R. Fantina et al., *Introducing Robotic Process Automation to Your Organization*,
https://doi.org/10.1007/978-1-4842-7416-3_6

Once conceptual design is finished, a readiness checklist must be completed. This assures that all the components required for development (the next stage) are complete. A template with a detailed explanation is included.

Once the readiness checklist is completed, it is forwarded to the governing body for a decision on whether or not to move to the next phase, development. The governing body generally consists of the following roles:

- RPA technical delivery lead
- Architect
- Developers
- Others as deemed necessary

Pause and Consider Names of roles differ from company to company. What are the coinciding roles within your organization? Also, consider what "others as deemed necessary" might be in your company. This could be a director, area manager, system analyst, or others. The governing body should be small, but should have the right people to decide whether or not to move to the next phase.

Please note that one of the advantages of this model is that work is done gradually, and there are very specific checkpoints for the governing body (decision-making body) to review the work and determine the continued feasibility of any project. A request that, during opportunity assessment, when only preliminary information is gathered, may seem promising in terms of ROI, competitive advantage, or any other factor may seem less so upon the deeper investigation done during conceptual design. This is true as the project moves from any phase to the next one. At any point, the governing body may decide that earlier evaluations have not proven true upon closer scrutiny and cancel the project. A project once given the go ahead in one phase, and later cancelled, does not indicate a failure in the earlier phase. This is the nature of RPA project progress.

Pause and Consider In many companies, projects often continue even when stakeholders know they are no longer needed. Cancelling a project is seen as a waste of the money already spent and the failure of various decision-makers. How will you convince management that using this model of periodic evaluation of a project is beneficial?

In RPA, the requirements document is generally referred to as the Process Design Document (PDD). This is described in the section, 'Governing Body Approval', below. If your organization is good with requirements gathering and documentation, continue to use that process; details of a typical requirements document follow.

But if the process of gathering requirements in your organization does not exist or is not efficient, and you need to capture requirements for RPA only, then PDD is the preferred way of gathering and documenting requirements. See the section, Governing Body Approval, of this chapter for that information.

Requirements Gathering

We will start with a discussion about functional requirements and the template we recommend.

Please refer to the template in Appendix 9, remembering that it is simply a model. If it works for your organization, fine, but feel free to adapt it to your organization's needs.

1 Level 1 Details

1.1 System

What systems must be accessed, during input, throughput, or output? List them here.

1.2 Goal

In this section, briefly but clearly describe the goal of the project. This might include a short description of the current situation and why it is a problem, along with a brief description of what the outcome of the project will be and its advantages. This is NOT where the goal is defined; that was done in earlier phases. This simply describes it in more detail. It is important not to fall into the trap of redefining the goal here and moving away from the initial request. This description must satisfy what the process owner requested. They must approve this document, so the description must be accurate.

1.3 Triggering Event

In this section, list what event causes the process to start. For example, client seeks to change beneficiary information.

1.4 Triggers

This is the specific action that causes the process to start. To follow the example in Section 1.3, the event is the client wanting to change beneficiary information. The specific action, or trigger, might be the receipt of an email from the client advising of the new beneficiary information. Or it might be notification from an advisor to make the change on behalf of the client. All possible triggers should be listed.

1.5 Precondition

In order for the process to start, what circumstances must exist for the Bot to begin? For example, the Bot may need to have access to certain software, the request may have to be in a particular format, etc. List all required preconditions here.

1.6 Post-condition

List here everything that is changed or produced as a result of the process. In the example earlier, post-conditions might include the following:

- Beneficiary information has been changed.
- Client has been notified that the change has been made.
- Advisor has been notified of the change.
- A report has been issued, listing all changes within the last day.

Tip Talk to subject matter experts (SMEs) to fully understand triggering events, triggers, and pre- and post-conditions.

2. Level 2 Details

2.1 Input Parameters

Parameters Name	**Description/Value**	**Mandatory?**	**Meaning if omitted (only enter if mandatory = no)**
What is the input to the process? It could be an email, the completion of another process, or any number of other things. Insert as many input parameters as there are. Add extra rows as necessary.	Briefly describe the input.	Is it mandatory or not?	

2.2 Output Parameters

Parameters Name	Description/ Value	Mandatory?	Meaning if omitted (only enter if mandatory = no)
What does the process produce? List its name here.	Briefly describe the output.	Is it mandatory or not?	
Include all outputs. Add extra rows as necessary.			

3. Level 3 Details

3.1 Main Flow Steps

In this section, list the steps of the normal process. For example, if in 70% of the cases, there is one way the process flows, list those steps here. In using the same example as before, the steps might be as follows:

1. Email received
2. System ABC accessed
3. Account number found
4. Beneficiary changed per instructions
5. Email sent to client
6. Email sent to advisor
7. Information added to daily report

3.2 Alternative Flows

There are some inputs that will have slightly different steps, but can still be handled by the Bot. List those process flows here.

3.3 Exception Flows

There may be some circumstance that the Bot will be unable to process. Using the example earlier, this could be because the account number couldn't be identified, or the current beneficiary name was wrong, etc. Explain what happens in those circumstances. Using the same example, the flow might be as follows:

1. Email received
2. System ABC accessed
3. Account number not found

4. Email sent to processing clerk for manual handling
5. Adding information to an exception report

■ **Pause and Consider** In order for the developers to create efficient Bots, this information is vital. How will you get it? Who do you need to speak with to obtain it and then validate it? Remember, spending the time now to get it right will prevent costly rework later.

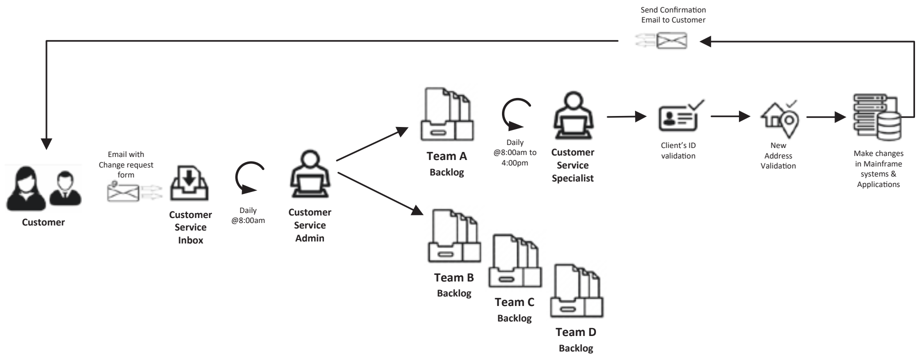
We will now discuss **nonfunctional requirements** (see Appendix 10 for a template).

"Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs."[1]

1. Automation service standards: This is a "boilerplate" section that simply states the date that current standards were set and by whom (example: COE). If there are no established standards at your organization, please disregard this section.
2. Privacy, data retention, and purge requirements: Include information on protecting customer privacy and also where the data will be stored. Also indicate when the document can be deleted.

Process Flow

Include a process flow for visual clarity. Using the same example, the process flow might resemble this:



[1] www.scaledagileframework.com/nonfunctional-requirements/.

The process flow is an at-a-glance view of the process from end to end. It enables management to have the necessary understanding of the process and assists designers in knowing what aspect of the process can be automated and how best to design the automation.

■ **Tip** The terms "process flow" and "process map" are often used interchangeably. Please keep in mind how they are used in this book.

Screen Captures

In order for the developers to create the Bot, they must know exactly how the process functions. Remember, RPA is to automate an existing process. While there might be some opportunities for improving the process itself during automation, no major process changes will be made. If they are required, the process is not a good candidate for RPA.

■ **Pause and Consider** During this phase, the request will be examined in more detail. Is it still a good candidate for RPA? Remember, at the end of this phase, there will be another governing body meeting, and you will be asked for a "go/no-go" recommendation. You may have recommended "go" at the conclusion of opportunity assessment, but following a deeper look, you may recommend "no go." That is perfectly legitimate.

In order to get screen captures, the most effective method is to record a person actually performing the process manually. This can be done through any of a variety of online meeting apps or, lacking anything else, using a cell phone.

After identifying the correct SME, schedule a meeting with them, and record the entire process. After the meeting, grab screenshots and place them in a word document, using arrows to indicate selections, and have a brief description of what is happening on each screen. Remember, in order to correctly automate the process, the developers must see every last step in the process.

Finally, include a screenshot of a typical output. What is it that the process produces? Is it a report in Excel format? A letter to a client or advisor? Whatever it is, include a screenshot of it.

■ **Pause and Consider** Your company may have standards that must be followed for development. Your new RPA program must conform to those standards. If there is a Center of Excellence (COE) in your company, those standards may be housed there. Bring in a COE representative early in the process, to assure that you have all the necessary steps in place to satisfy company requirements.

Planning

Formal planning begins once requirements are gathered. For a "Waterfall" methodology, this means that they have been fully captured, documented, and validated for the entire project, start to finish. If you are using an Agile methodology, or a hybrid of both, you must determine at what point you have requirements sufficient to begin planning.

Once you have reached the point where requirements have been sufficiently captured to satisfy the methodology used (Waterfall, Agile, or a hybrid), planning is the next step. Planning is a vital step in the process of automation. The old adage "If you fail to plan, plan to fail" is certainly true with RPA.

In planning, the first step is to schedule meetings with the pertinent stakeholders. Most of these will have been identified by now as you have talked to the business owner(s) and subject matter experts. Assure that you have all the necessary stakeholders identified. This can be validated in a brief discussion with the business owner(s). This initial planning meeting should include product (business) owner(s), key SMEs, a business analyst, developers, and/or a deployment team member.

The main purpose of this first formal planning meeting is to assess the size and complexity of the process for which automation has been requested. With the key stakeholders who will attend this meeting, you will be able to determine if the level of requirements detail collected is sufficient to start development or if additional detail is needed. This is a key outcome of this meeting.

Some information may be reviewed at this meeting, but if additional requirements are needed, they should be obtained first. You want to be sure to discuss what applications the automation is going to work with and understand if these applications are used in any existing automations. If they are, identify which automations are using the applications. You also need to determine what access is required to use these applications. You may need to engage with your organization's security team to get the right authorization. Lastly, you should determine if a proof of concept (POC) is required in order to determine technical feasibility.

If additional requirements are needed, or these questions cannot be answered at this meeting, do not proceed. Obtain the missing information, and meet again to finalize these issues.

For a Waterfall methodology, once the requirements are completed, reviewed, and signed off by the required stakeholders, the development team will meet regularly.

If using an Agile methodology, once sufficient information is obtained and the user stories are developed, the Agile project management process begins.

■ **Pause and Consider** The development team will meet "regularly." What will "regularly" mean in your organization? For an Agile team, what meetings additional to the daily stand-up will be required? If using a "Waterfall" methodology, how often must you meet? It is likely that your team may require more meetings at the start of the project and fewer as the project progresses.

During initial development team meetings, requirements must be divided into smaller tasks, functions, or stories. During these meetings, the team will identify dependencies and decide the sequence of development.

Once this information is developed, and the team has a good idea of complexity and a timeline, it is time to once again review the project with the governing body (governing body). The team will briefly present the information they have identified and make a recommendation on whether or not to proceed with the project. Remember, as more information is gained, a project may be validated, or it may be found to not have the promise that it initially appeared to have.

■ **Tip** Remember, a project cancelled at this point does not indicate failure! Based on your earlier knowledge and recommendations, the governing body has agreed to proceed. If new information has come to light that causes you to think the project is no longer feasible, that just demonstrates that the RPA process itself is working as it should.

Governing Body Approval

■ **Tip** We refer to the governing body as the "governing body," to indicate that it is the "gatekeeper" to the next phase. Use whatever terminology fits your organization.

As with all the phases, if the governing body decides not to proceed with the project to the next phase, the business owner will be advised of this decision and the reasons for it. If, during discussions with the team, any other possible solution was identified (e.g., a new software tool, a significant improvement to the manual process, etc.), the business owner should be informed of those possibilities.

If the governing body has approved moving to the next phase, the development team now starts working on developing process design documents (PDD).

As mentioned earlier in this chapter, PDD is the preferred method of documenting requirements for RPA projects. If your organization doesn't have an effective method of gathering and documenting requirements, use the following method for your RPA projects.

A process definition document is fairly straightforward. It answers some basic questions to assist designers and developers in automating the process it describes. It conveys what the process does, its inputs and outputs, the steps currently performed to accomplish the process, and any decision points in the process. Regarding any decision points, you need to determine if they can be automated. If not, they will need to be sent for manual handling.

The PDD also conveys who owns the process and who actually performs it. The document will also specify how often the process is performed: it might be hourly, weekly, monthly, on demand, or any other configuration. Also, what systems or applications interact with it? Which ones either provide input to the process or receive its output.

We talked before about authorization; the PDD lists what authorizations are needed for any systems or applications that interact with the process.

The PDD will also list the currently known risks. What could go wrong? What happens if something goes wrong? Does the process revert to manual performance until the problem is resolved? Who is responsible?

Lastly, the PDD will explicitly state the endpoints. How do we know that the process has completed?

A PDD can be accompanied by a flow diagram showing the current process and how that same process will/would look if automated. It can also specify the endpoints of a manual as well as an automated process.[2]

Technical Feasibility

There are several components to be validated. These include the following:

- Defined logics
- Rule-based steps
- Input and output data

[2] For more information, please see The Process Definition Document (robocorp.com).

- Complexity analysis
- Volume of transactions
- Development effort

The following steps are required to do the technical feasibility:

a) List all the applications the automation is going to interact with or use; this can be obtained from the process design document (see previous discussion).

   i). Determine if automation tool provides functions of scrapping or interaction with these applications.

   ii). Determine if there is any application that the automated process would not be able to interact with (e.g., any website that has complex JavaScript that updates its classes too frequently).

b) Will the automated process provide the required security (again, refer to the PDD)?

c) Can the process, when automated, run at the speeds and times required by the business?

d) Are there any regulatory requirements?

Tip You can see how the PDD will be used extensively during design. It's vital to assure that the information it contains is complete.

Proof of Concept/Prototyping

In some cases, the RPA team may not have experience with the systems or applications with which the solution will interact. Or there might be very serious risks, such as privacy breach or risks to stability, that are involved. In these cases, having a proof of concept or developing a prototype can be highly beneficial. It may not even be necessary to create a prototype or proof of concept for the entire automation, but only for those parts that are particularly risky. Work with your SMEs and "risk and compliance" department to determine when this might be necessary.

Solution Design of Automation (What Will the BOT Look Like?)

We recommend using a layered design approach. The following steps will guide you:

1. Controller (process): This is the main controller that controls the Bot execution. It can be seen as the "supervisor" of other tasks within it.
2. Task container (business objects): This is the "nuts and bolts" of the Bot. Here are the various small tasks that will be used repeatedly by the Bot to execute the steps required. These steps are executed by the controller.
3. Communicator: Every automated process must issue some kinds of reports to show its results. Additionally, if transactions enter the Bot that it doesn't know how to handle, those transactions must be sent to someone for manual handling. As suggested by its name, the controller is used for these and other communication purposes as may be required.

Pause and Consider What are the communication needs for the processes you are automating? How much information, if any, does senior management need? Where will transactions that cannot be handled by the Bot be sent? What record of them will be kept, and where will it be sent, stored, etc.? How often will a report of the process execution be created, and where will it be sent, stored, etc.?

Let's prepare the solution design document (SDD).

Hemant Joshi, CEI and Managing Director of eGleis Technologies, detailed what a SDD is.[3]

To summarize, a solution design document is a document that describes at a high level the design and the optimal way of implementing a technical solution to your project. It is created for every business process that is automated using RPA technology. This document is created by the RPA developer who will automate the business process, with input from the RPA solution architect who will review it before handover to RPA operations. It comes under the design phase of the RPA development life cycle.

[3] www.linkedin.com/pulse/writing-solution-design-robotic-process-automation-project-joshi/.

A typical and effective solution design document (SDD) contains the following components (see Appendix 8 for a template):

- High-level design: This will include an automation flow diagram and an infra architecture diagram.
- Information security: What are the compliance requirements? This can be obtained from the process design document (PDD).
- Prerequisites to automation execution: Consider hardware, software, licenses, etc.
- Exception handling: What will be done with transactions that enter the process, but cannot be handled by it? Every process has some "exceptions to the rule," and you must determine explicitly how these will be handled.
- Debugging tips: What suggestions can you include to help developers in the future to resolve problems? Remember, the developer who initially develops the automation may not be available if problems arise with it at some future date.
- Success criteria: Lastly, document what needs to happen for the automation to be successful.

■ **Tip** While each automation will be different, keep in mind that there may be reusable components. This will add to the efficiency of the entire RPA initiative.

Writing a best-fit solution design document (which is sometimes referred to as a solution architect document) for any RPA solution is the most critical piece in the RPA development process.

■ **Tip** Assuring a complete and thorough PDD (process design document) will greatly assist in the creation of a good SDD (solution design document).

The following are the major points to keep in mind in the process of writing a good RPA solution design document.

Development/Implementation

Once the SDD has been completed, reviewed, approved, and signed off, development of the Bot can begin.

In developing an RPA solution, development or implementation activities are different from those in more traditional settings. For a newly established RPA development team, the work can be divided into three phases. The first is solutioning the process steps. What will the automated solution exactly look like? What will the Bot do? This establishes the vision or view of what the technical solution will be.

Secondly, the Bot is designed in full detail. Using the solution that has been envisioned, the design is now created in detail.

Finally, the actual coding and implementation of the Bot in the automation tool used by the organization is done. This, of course, follows creating the vision of the technical solution, and then the actual work of coding is performed according to the design that has been created.

As with any development effort, coding standards must be followed. If your organization does not have coding standards, these can be developed by the team.

Pause and Consider If your organization doesn't have coding standards, the introduction of an RPA initiative is a fine place to develop them. How these will be defined in your organization is up to you. But you need to assure that they are sufficiently rigorous and can be used across platforms and business units.

If your team is unsure of the importance of coding standards in software development, you might want to remind them of some compelling reasons. Coding standards help improve the quality of the overall software system. They reflect a harmonized style, as if a single developer wrote the code in one session. Coding standards also reflect more than just the software; they are a reflection on the organization itself. Additionally, they greatly help to improve the maintainability of code; any qualified developer following the established standards will be able to maintain the code. Having coding standards also helps to limit risks; there is less unknown because by adhering to the standards, anyone can read the code. Coding standards also improve code quality; all developers must adhere to a professional, pre-approved standard. And lastly, having these standards helps all developers be familiar with the code structure.

Nishant Goel and Satyendra Shinde of *BOTmantra* identify five (5) key coding standards and several subcategories. The following list shows them all:

Top 5:

1. Readability
    a. Name convention standards and compliance
    b. Zero usage of junk code
    c. Componentization
    d. Simplified logic
2. Configurability
    a. Performance parameters
    b. URLS, files, and folder paths
    c. Email IDs
    d. Credentials
    e. Business rule threshold parameters
    f. Log messages
    g. Email formats
    h. Generic design
3. Reliability
    a. Exception handling
    b. Best interaction technique
    c. Memory leakage avoidance
    d. Auto-recovery and auto-healing mechanisms
4. Security
    a. Authorization
    b. Authentication
    c. Credential management
    d. Business data storage
    e. Data sharing
5. Performance
    a. Delay management
    b. Parallel execution

c. Interaction technique usage

d. Memory management[4]

We will now look at each component in detail:

1. Readability: How easy is it to read the code? Do naming conventions exist? Have naming conventions been followed? Is the code as simple as it can be? Are there comments that explain modules or even separate lines of code, if necessary?

■ **Tip** The developer who creates the code may not always be available if problems arise. Any qualified programmer should be able to read and understand the code.

---

2. Configurability: Bots must be created with the possibility of further changes. Perhaps new parameters will be added, or different systems will need to "communicate" with the Bot. Are you including performance parameters, files paths, folder paths, various credentials and business rules, etc.?

■ **Tip** Always design with the future in mind.

---

3. Reliability: What degree of accuracy is required? Remember, we said that automating a manual process with RPA is best done when there are few events that must be removed from the process for manual handling. What exception rate is acceptable for the particular process being addressed? How will those exceptions be handled? Are there auto-recovery mechanisms in place to avoid downtime?

■ **Tip** It is sometimes believed that customer-facing processes require a higher degree of accuracy than internal processes. Don't cut corners on any process. Doing so will damage your credibility.

---

4. Security: A rigorous risk assessment will greatly increase security. As mentioned previously, the risk assessment will be ongoing: in the early stages, only limited information is available, so a complete view of all risks will be impossible to attain. That is fine! But as you progress with the project, additional risks will be identified. Each must be either mitigated or accepted, unless they are considered to be so great that the Governance Committee

[4] https://botmantra.com/rpa-coding-bestpractices/Accessed on April 15, 2021.

decides to cancel the project. Is it important to know: What authorizations are required to use, change, or stop the automation? How will the Bot "know" if it should run or stop? How will authentication be accomplished? Where will data going in and coming out of the process be stored?

■ **Tip** As mentioned previously, early involvement of your risk and compliance team will be highly beneficial. Keep in mind that that team might have a different name in your organization.

---

5. Performance: How will you optimize performance? What interaction technique best suits this particular process? Can the Bot run on two or more processes simultaneously? What is the most efficient business logic configuration for this process?

■ **Pause and Consider** The four previous components all impact performance. What are your performance standards? What is required by your organization? Defining these requirements clearly will greatly assist you in defining what is required for readability, configurability, reliability, and security.

At this point, you may be thinking that there is an awful lot to accomplish to introduce and implement a successful RPA program in your organization. Yes, there is a lot to do. But what you are doing is nothing short of revolutionizing the way your organization operates. Such an undertaking will take time and effort but, if done properly, will bring significant benefits. So don't be intimidated by the work required; follow the steps we have included. A task is never quite so daunting when taken in small steps.

■ **Pause and Consider** As with each of these standards, you need to determine which ones are required for your organization and how rigorous each should be. What configurability components are vitally necessary within your organization? Which are "nice to haves"? How detailed should each be? The answers to these questions will vary from company to company and even within companies, from one business unit to another. Also, what is required for one automation may be more or less than will be required for another.

■ **Tip** As the team gains more experience, technical feasibility will be folded into design activities, but for the present time, it should be separate.

Code Reviews

For many organizations, code reviews may be informally performed, if done at all. Establishing a code review practice in your development activity or team is very important. Dan Radigan, writing in *ATLASSIAN Agile Coach*, says that "Code review helps developers learn the code base, as well as help them learn new technologies and techniques that grow their skill sets."[5]

In the process of code reviews, other developers review the code to assure quality; remember, there is no place for "spaghetti code today". Code reviews help ensure the readability of the code.

The code review also ensures that coding standards, which we've already discussed, are strictly followed.

Another advantage of code reviews is that they help to assure that code can be easily maintained.

They also assure that exception handling standards for the particular process being automated are followed.

Code reviews, like all aspects of development, are more efficient and effective when certain guidelines exist. Having clearly documented coding standards is key. Also, it is helpful to provide and follow checklists; create these from your code review standards.

Additionally, it is often beneficial to provide pre-review training, to assure that developers are familiar with the standards they must follow.

Having code templates is also very helpful.

Tip If you reduce code reviews to an afterthought, something you will do "if time allows," be prepared to fail. Code reviews are an integral part of any RPA project.

Testing

Once code reviews are completed, and any changes to code have been made and re-reviewed, you are ready to test.

For successful testing, the testing teams must be completely familiar with the process. Knowing and understanding the relevant PDD (process design documents) and SDD (solution design documents) are key to success.

[5] www.atlassian.com/agile/software-development/code-reviews#:~:text=Code%20review%20helps%20developers%20learn,that%20grow%20their%20skill%20sets.

An exhaustive list of test use cases must be created; this could be an overall test plan. This will include testing data for the automation.

■ **Tip** As you progress with the RPA initiative, you will learn from defects and thus improve testing procedures. Even the most comprehensive test plans will not be bulletproof; there will be continued fine-tuning of testing procedures.

Testing automation can be a challenge for several reasons. A major one is that many organizations don't have testing environments. Also, it is sometimes difficult to get data, and the Bot may behave differently based on the data it is given.

Additionally, the more applications with which the Bot needs to interact, the more difficult it becomes to build and manage parallel environments for testing. Because updates to the applications happen during Bot execution, sometimes testing can only be done in the production environment. This includes the Bot needing to update some screens that may have wrong data or the possibility of it making incorrect or partial updates.

Following established testing techniques is vital when transitioning to RPA. We will look at some of the aspects of RPA testing that you may find different from testing in other methodologies.

Also, please remember that the people performing testing must be trained. Someone with minimal training may be able to execute the testing (see step 4), but steps 1, 2, and 3 require a degree of knowledge that if learned "on the job" carries high risks. Be sure that the people performing these steps are trained in doing them. If a less experienced person is going to perform step 4, be sure they are trained on what's required for step 5.

1. Know the business process: Those performing the testing must know and thoroughly understand the business process that is being automated. Without this knowledge, they will be unable to properly develop and implement effective tests.

   The best way to understand the process is to review the process definition document (PDD), the solution design document (SDD), and any other documentation that was created during the design phase of the automation. Once the team understands the business process, it can move on to the next step: creating the test scenarios to actually test the codes.

■ **Tip** If something is unclear, don't hesitate to contact the business owner for clarification. Do not cut corners in understanding the process.

---

2. Create test scenarios: This is another key area. Review the PDD and the SDD, with special attention paid to the SDD which, if done properly, will list the most important scenarios to be tested. But don't ignore the PDD; very useful information for building test scenarios can be gleaned from that document, too.
3. Write test scripts: Now that the testers thoroughly understand the process to be automated and have created test scenarios, the next step is to create a series of test cases. Test cases are very specific: they list the inputs, expected output (or outputs), and a column (often test scripts are written in a spreadsheet) to indicate if the test passed or failed and another column for any notes. This column is especially useful for tests that fail.

■ **Tip** To maximize the effectiveness of test scripts, they should be reviewed by the design team. This "second set of eyes" is very familiar with the process and can assist in finding any testing gaps.

---

4. Perform testing: Now that you have completed all the preliminary work, the actual testing is performed. Complete all the work of each test case, documenting if each passed or failed, with any relevant comments.
5. Rectify defects: Do not expect all test cases to succeed; if they do, it's likely you have not created sufficiently effective test cases. Defects are common, and identifying them is a main purpose of testing. All failed test cases should be fully documented; too much information is better than too little. There must be sufficient information for the development team to be able to correct the errors.

■ **Pause and Consider** What is the testing process in your organization today? How must it be adjusted for RPA testing? Will there be resistance to these changes? If so, how will you overcome it?

The test documentation must be stored in a repository that is easily accessible to the people needing access to it. Remember that the developers who will need to see it are probably working on multiple projects. You need to be sure they can quickly and easily find the information they need.

Pause and Consider Where is test documentation currently stored in your organization? Is it sufficiently accessible to suit the needs of your new RPA initiative? If not, what changes need to be made? Who will make them?

Much of what has been discussed in this session relates to quality assurance (QA) and quality control (QC). That is not coincidental! Let's look at the definitions of each:

Quality assurance: "Part of *quality management* focused on providing confidence that *quality requirements* will be fulfilled."[6]

Quality control: "Part of *quality management* focused on fulfilling *quality requirements*."[7]

So quality control can be seen as fulfilling the needs of quality assurance.

Pause and Consider Does your organization have formal quality assurance and quality control functions? If so, how can they best be integrated into your new RPA initiative? What, if anything, will need to be done differently? If there are no formal quality assurance and quality control functions, how can you still assure that the needs of both are met within your RPA program?

Automating Test Execution

When the RPA team is automating customer or client processes, it will be beneficial to automate its own processes also. This includes automating the testing process of Bots.

Since all Bots are unique in their function, before deciding to automate their testing, there are a few things that need to be considered. Determine how frequently will testing automation be used. For each particular Bot, how simple or how complex would it be to create automation testing. Also, determine if the testing automation you may create could be used to test other Bots in the future; try to determine its reusability.

If it is determined that automated Bot testing would be beneficial, there will be three components to do so. You will first do automation to set up the

[6] https://asq.org/quality-resources/quality-assurance-vs-control.
[7] Ibid.

environment and create the required test data. You will then execute the Bot automation. Finally, you will gather the output data after the Bot execution and validate the output to create a test summary.

Case Study

Now let's look at how solution design is implemented in our case study which was introduced in Chapter 4. We moved forward with it in Chapter 5. Now we will see how our hypothetical case would move through solution design.

Let's start by creating the SDD. In this document, we need to populate a variety of information. There are a variety of templates available online; use the one we provided or any other template that seems to meet your needs.

Solution Design Document

Phone and Address Change

		Comments
Author	John Williams, CSS Supervisor	
Date	July 28, 20xx	
Version	Draft 0.1	Initial draft

Approval	Signature
Mary Brown, Manager, Customer Service	

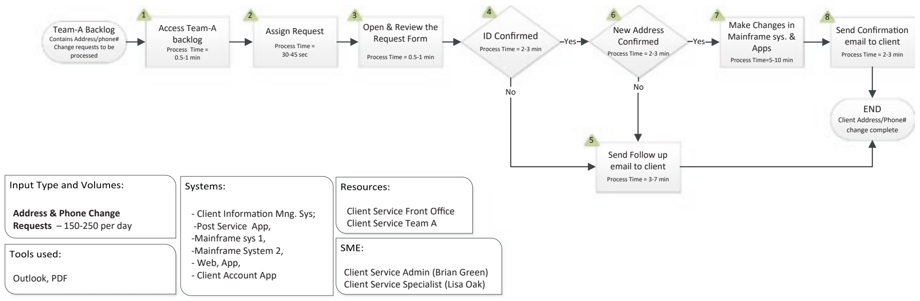
Purpose of This Document

This document contains the solution design for the phone and address update process. It presents the high-level "As Is" process steps and the "To Be" process steps that will be automated.

The current process is as follows:

- The request to change an address and/or phone number is received in the specified mailbox.
- The CSS verifies the client ID and updates the information as required.
- The CSS confirms the update with the client.
- The CSS maintains a record of updated information.

Flow Diagram



Goal of the Automation

The goal is to create a Bot that will retrieve the emails, read and parse them, update the information as specified, and inform the client that the information has been updated.

Applications Involved

Name	Internal/ External	Type	Credentials Required?	Read/ Write
Web Application – Client CRM	Internal	Network-hosted application	Yes	Both
Mainframe 1	Internal	Mainframe	Yes	Both

Inputs

Name	Description	Initial Value
AddressUpdate_ ClientID	An email that will contain the client ID and a new phone number and/or address	Address and/or phone information

Process Steps

First Sub-process

Step Number	Description
1	Bot starts.
2	Bot logs into email account.
3	Bot reads emails.
4	Bot extracts client information.
5	When all emails have been read, Bot sends completion message.

Second Sub-process

Step Number	Description
6	Bot logs into Web CRM.
7	Bot searches with client ID.
8	Bot doesn't find client ID – "not found" message sent.
9	Bot finds client ID.
10	Bot populates client information in appropriate fields.
11	Bot activates update.
12	Bot sends completion message.

Third Sub-process

Step Number	Description
13	Bot logs into Mainframe.
14	Bot searches with client ID.
15	Bot doesn't find client ID – "not found" message sent
16	Bot finds client ID.
17	Bot populates client information in appropriate fields.
18	Bot activates update.
19	Bot sends completion message.

Fourth Sub-process

Step Number	Description
20	Bot sends email to Client.

Exceptions

- Business
  - Client ID is missing in the email: Send exception message EmailMissingClientID.
  - Last name or first name is missing in the email: Send exception message EmailMissingFirstOrLastName.

- Technical
  - Web CRM URL not reachable: Send exception message CRMInReachable.
  - Login password failed – Web CRM: Send exception message LoginFailedPWDExpired.

Success Criteria

The process is complete when the Bot sends a job completion notification daily at (time – TBD).

■ **Debugging Tips** If the Bot fails to log in to email, run the email subtask. If there is an error in updating the address or phone number in the web application, run the web application series of tasks, and check if updates are done successfully by the Bot. The same procedure can be followed to check to assure that the mainframe is updated.

Maintenance

This Bot has normal priority. If Bot fails or errored out, run the database script to reset the Bot and restart the Bot.

Bot Scheduling

Bot will run every day, Monday through Friday, at 11:00 in the morning.

Disaster Recovery

In case of complete failure of the Bot, the team will need to revert to the manual process:

- The operations team will perform the process manually, until notified that the issues have been resolved.
- The RPA team will be responsible for operational tracking, monitoring, and maintenance. In case of any temporary outages or planned downtime, the COE will communicate to business well in advance and expedite the restoration of services.
- Process owners are expected to systematically review and audit the results to ensure the functional value is achieved.

- Any changes to host systems will be proactively communicated to members of the RPA team far enough in advance to assure they can make any required adjustments.
- The process owner will receive notification emails for all failures.

Glossary of Terms

Term	Description
RPA	**Robotics process automation**
SIT	**System integration testing**
SLA	**Service-level agreement**
SME	**Subject matter expert**
UAT	**User acceptance testing**

Appendices

Appendix A: Exception messages

– EmailMissingClientID

– EmailMissingFirstOrLastName

Let's look at these components in detail.

The cover page includes several components: name of document. This is the SDD, and it is necessary to specify what project it pertains to. The document author is the person responsible for creating the document. That may be the architect or designer, but it could also be another member of the team. The document version should indicate whether it is the first or a later draft, if it is final, and if it has been approved. Generally, draft versions start with 0 and are incremented as the draft is reviewed and updated: 0.1 is the initial draft, 0.2 would be a draft that has been updated, etc. The "Comments" column would be used to make any explanations. For example, Draft 0.2 might have a comment such as this: "Revised following review with Mary Brown, who requested that XYZ be altered."

Also on the cover page is the space for the approval. The Name/Role is the name and role of the person who is responsible for approving the document. That could be the product owner, business owner, delivery manager, or another role. The "Signature" would be electronic when the document is approved by the person specified.

Pause and Consider Remember that versions (draft, final, etc.) and role names (product owner, business owner, etc.) differ from one company to another. Use the names that pertain to your organization.

The "Purpose of This Document" provides a very short description of process automation overview. This is very helpful for understanding what the Bot is going to at a high level; additional detail is included in later parts of the document.

The "Flow Diagram," which was developed in the PDD, is included here, to provide developers with an "at-a-glance" understanding of the current manual process that they will be automating.

The "Goal of the Automation" starts to drill down to state more specifically what the Bot will do.

In "Applications Involved," you will list which applications the automation is going to interact with. By identifying all applications, you can discover what kind of access the automation will need to interact with each of them.

The developers need to know what the "Inputs" are. In our example, there is only one input (email from the client), but this is just a simple case study. Your "real-world" work may include more.

While a very high-level description of the process has been provided, that is not something that the developers can build from. Now you need to list the individual "Process Steps" in detail. In our case study, there are four, simple sub-processes. As you can see, they are very specific and at a very low level.

Any process is going to have exceptions. In our case study, that could be an email that doesn't include the client ID, or the client ID may have too few or too many numbers; the new telephone number could be missing a number, etc. These "Exceptions" must be dealt with by the Bot, and this section explains how that will be done.

The Bot has a beginning point and an endpoint, and it's necessary to assure that anyone monitoring it, either in real time or periodically, knows that it has completed its work for each cycle (hourly, daily, etc., as specified). This "Success Criteria" must be defined. Will it be an email to a manager, sent by the Bot, to say that all emails in the specified mailbox on July 29, 20xx, have been processed? Will it be a notation on a larger report or dashboard? Whatever it is, this is the place to specify it.

No application will ever run perfectly all the time. Because of that, the SDD should include some basic "Debugging Tips." These are suggestions you can include to help developers in the future to resolve problems. Remember, the developer who initially develops the automation may not be available if problems arise with it at some future date.

Maintenance

Once the Bot is deployed into the production environment and is live, it starts following its schedule. Then maintenance comes into picture. Every Bot has an SLA (service-level agreement) to provide support that sets expectations of the business in case the Bot fails to perform as required. Mainly, there are two aspects to it. One aspect is technical and the other is related to the business side. Technical aspects cover when the Bot is unable to perform because of any technical reasons, for example, servers are down, infrastructure maintenance, etc. Business issues might be related to the Bot being unable to perform as per requirements because data is not available or when input systems that feed the Bot are not ready at the time of Bot execution.

In case of technical reasons, the main responsibility of the technical team or COE is to communicate to the business information about the outage and also share the situation with any other relevant stakeholders. They should all be kept apprised of the progress and expected time when Bot will be available as per schedule. For any business reasons, the technical team needs to coordinate with the business team so that the Bot can be stopped or suspended until the business reasons are resolved; the technical team will also work closely with the business team and/or COE team to keep everyone on the same page.

Scheduling: The developers need to know when the Bot is to run. This may be daily at a certain time or multiple times; it might be weekly, monthly, etc. This will have been determined early on in discussions with the business and will be documented here for the development team.

Disaster Recovery: What happens if the Bot ceases to work? Either it stops work altogether or is malfunctioning to the point that it isn't doing its job. Again, what happens in this case will have been discussed with the business much earlier, usually when initially creating the risk assessment. Whatever the determination is should be included here. At a minimum, reverting to the manual process until the Bot has been repaired is one aspect of disaster recovery.

After completing SDD, it can be sent for sign-off as per the process in your organization.

Once SDD approval has been received, you are ready for developing the Bot in your selected automation tool. The steps included in this book will be effective with whatever tool you use.

Testing

After building the Bot to automate the process in our case study, the next vital activity is testing the Bot.

To capture all tests, testing execution, and its results, a test summary document can be created. Keep the test summary document as simple as possible so it will be easy to understand. Here in our hypothetical automation, we are going to follow a test summary that has three tabs:

– The first tab contains information about the Bot and test results.

– The second tab has test scenarios with results.

– The third tab has test data.

Test Summary

VERSION HISTORY			
Author	Version	Date	Comments
Allan Brent	0.1	03-Jan-21	Initial version

APPROVAL DETAILS	
ID	Automation - 001
Business Lead	John Patterson
Business Sponsor	Alex Macleod
Approval Date	
Approval Evidence lin	N/A

TEST RESULTS	
Number of Test Cases	3
Test Cases Passed	3
Test Cases Failed	0
Test Cases Blocked	0

ENVIRONMENT DETAILS				
Version	Environment	Environment Description	Risk	Risk Mitigation
1	Test	Testing environment for the bot	No	N/A

** If tested in Production, then Risk and Risk Mitigation should be stated and signed-off by the Business Sponsor prior to the testing*

Note that this tab has a Version History, Approval Details, Test Results and Environment Details. The information required in them is self-explanatory. For "Environment Details," include any risks that pertain to testing in that environment. This is especially critical if testing in a production environment.

Test Scenarios and Results

Prerequisites	Process Step Number	Test Case Description	Expected Result	Actual Result	Test Case Actual Status
Email account is set up on email server.	1	Bot logs in to email inbox.	Bot successful login to email account.	Bot able to log in	passed
Web CRM application is up and running.	2	Bot logs in to Web CRM and performs update to one address and phone number.	Bot successfully updated one record.	Bot able to update record	passed
Mainframe 1 is up and running.	3	Bot logs in to Mainframe and performs update to one address and phone number.	Bot successfully updated one record.	Bot able to update record	passed

This section is also self-explanatory.

Pause and Consider You will note that in keeping processes and documentation as simple as possible, you will have more success with your RPA initiative. But remember not to compromise quality as you simplify.

Test Cases

ID	FRD-Based Scenarios
Auto-001	Bot successfully logs in to email inbox and reads emails.
Auto-002	Bot successfully logs in to Web CRM and updates address and phone number.
Auto-003	Bot successfully logged in to Mainframe 1 and updated address and phone number.

This section shows the specific test cases that must be executed.

(For the purposes of the case study, we are only showing user acceptance testing).

Following testing, business sign-off is required. This is done by forwarding this testing document to the business owner and receiving and then storing their approval, usually as an email.

Testing can be completed in iterations; it can also be done more than once to assure that the Bot delivers what is expected and is able to handle exception situations as per requirements.

At this point, the Governance Committee is advised of business approval and solution design; no meeting is required. If the committee members have any questions, they will notify the RPA team.

After completing testing and obtaining approval, preparation for packaging and deployment can start; this is explained in detail in the next chapter.