TARPON HEALTH

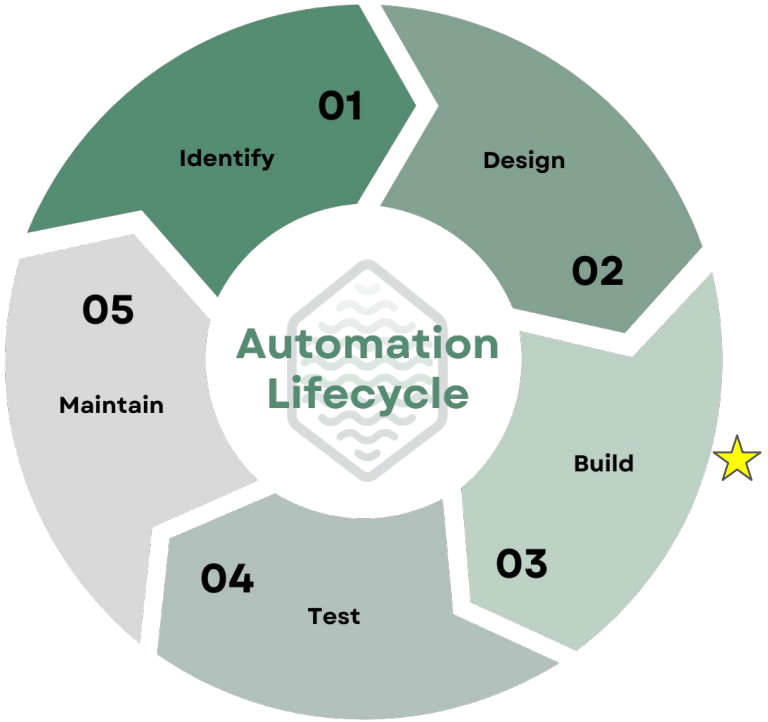
Managing Automation Development

Agenda

What we will cover today

1. Sprint planning
2. Standups
3. Development
4. Review
5. Retrospective

The Tarpon Health Automation Lifecycle



The development process

The following steps will help your organization scale.

The team meets to plan the sprint, including identifying the tasks that need to be completed and estimating the time required for each task.

The team meets daily to discuss their progress, identify any blockers, and make adjustments to their plans as needed.

The team works on the tasks that were identified in the sprint planning meeting.

The team presents their work to the stakeholders and gets feedback.

The team reflects on the sprint and identifies areas for improvement.

Why sprints?

TARPON HEALTH

Definition: A sprint is a fixed-length period of time (typically 1-4 weeks) during which a team works on a specific set of tasks to achieve a specific goal. Sprints are often used in Agile software development, but they can also be used in other types of projects, such as RPA.

Benefits:

1. **Focus and productivity:** Sprints force teams to focus on a specific set of tasks for a limited period of time. This can help to improve productivity and reduce distractions.
2. **Early feedback and iteration:** Sprints allow teams to get feedback on their work early and often. This can help to identify and fix problems early on, before they become too costly or difficult to address.
3. **Agility and adaptability:** Sprints allow teams to be more agile and adaptable to change. If new requirements or priorities emerge, teams can adjust their sprint plans accordingly.

How do you do it well?

Agile framework is a great guide for development.

The flow

1. The RPA manager / lead developer presents the sprint goals (based on business needs).
2. The team discusses the sprint backlog and identifies the work that is needed to achieve the sprint goal.
3. The team estimates the time required to complete each piece of work.
4. The team prioritizes the work and selects the work that will be completed in the sprint.
5. The team updates the sprint backlog.
6. The work gets built into a tool to track the progress.

Discussion

- How strict do follow the agile framework?
- How do you define the goals for each sprint?
- How do you estimate how much time a given task will take?
- What is a typical sprint timeframe for an RPA project?

Running a standup

Once the sprint is planned, you have to manage it.

Goals of the standup

- Understand the status of the project
  - For example: Will the tasks will be completed within the sprint timeline?
- Provide transparency into what was completed and what is on deck
- Highlight blockers that need to be resolved quickly
- Outline next steps to keep the project on track

How to make it effective:

- **Brief:** Standup meetings should be short and to the point. Aim for 15 - 30 minutes.
- **Focused:** The focus of the meeting should be on the work that will be done today, next priorities, and any blockers that are preventing the team from making progress.
- **Transparency:** Everyone should be able to see the work being done and the blockers preventing progress.
- **Actionable:** The standup meeting should end with a clear plan for how the team will address any blockers and make progress on the work that was discussed.

Example

Today, Tomorrow, Blockers are the essentials of a good standup.

What I plan on doing today (or what I did today):

- Service line logic
  - I've re-reviewed the logic and feel confident it is working as designed
- Using EHR service line service date field
  - Exclude "Extract Service Date" because we cannot guarantee it's the same for each line item
- Loop is set to handle however many lines are present
  - I've tested it on 6 service lines and plan to test 9 or 10 lines next week
- Removal of modifiers and accurate logging

Next day priority:

- Building a data table for post automation export

Blockers:

- Lack of facility data for testing

Align the conversations with the tasks in your project management tool (e.g., Kanban board).

Stay action oriented

Tools and applications

Below are tools and applications that help with sprint planning.

Common Tools

- **Kanban boards:** Kanban boards are visual tools that can be used to track the progress of tasks throughout a sprint. Tasks are typically represented by cards that are moved through different columns on the board, representing different stages of the workflow.
- **Burndown charts:** Burndown charts show the remaining work to be completed in a sprint over time. This can be helpful for tracking progress and identifying any potential risks.
- **Velocity charts:** How many ‘story points’ are being completed on average. It allows you to estimate how much work you should tackle in a given sprint and how successful the team is in achieving the milestones.

Common Applications

- Jira
- Azure DevOps
- Smartsheets
- Others?

Before you start

Build scalable practices to avoid the ‘tribal knowledge’ pitfall.

The amount of time you invest now will payoff when you are maintaining the automation. Downstream use of documentation will improve the issue identification and the time required to fix it.

Best practices to avoid downstream issues

- Consistent logging
- Thorough ‘READMEs
- Version control and code storage using native RPA platform functionality or tools like Github
- A shared development strategy and framework, such as UiPath’s REFramework
  - Using prebuilt components during development allows for standardized, scalable, and “maintainable” automations

Logging

Don’t underestimate the importance of good logging.

Example: Adjustments

High quality logging

- Account Number (HAR/ETR)
- Result (Success/Failure)
- Adjustment/Txn Code
- Adjustment Amount
- Processed Time
- Account Balance
- Payer ID
- Approval Required Flag

Low quality logging

- Account Number (HAR/ETR)
- Result (Success/Failure)
- Adjustment/Txn Code
- Adjustment Amount

In order to have a high quality review, the **automation’s logs need to be capturing all the elements that constitute success** as defined by the business. More information will help you diagnosis the underlying issue (i.e., Is it a development or process issue?).

README

Documentation for the development team.

Sections

- **Overview** - a non-technical description of what the automation is doing.
- **Applications** - a list of software or applications being used along with a list of techniques.
  - E.g., Browser vs. native application, native selectors vs. computer vision, etc.
- **Version control / history** - a brief description of changes over time.
- **Common points of failure** - highlight areas that created issues when building.
  - E.g., Long load times, poor computer vision match %, etc.
- **Technical details** - Describe why you are taking the specific action, this includes the input and output of each workflow (e.g., import arguments)
  - Include where the logs live, how to access the production machine, flags within the automation specifically designed for testing, etc.
  - Organizations should try to document every workflow if they know multiple developers will be touching the automation.

May overlap with other documentation

Common development techniques

Make sure that these tactics are deployed to reduce the need for maintenance.

It's important that when the automation reaches a failure point, that it is able to redirect itself appropriately. These techniques ensure your automation will fail gracefully and keeps processing the queue.

Retry scope: Common development strategy to execute a particular action, or the same task again, if the desired action initially fails or encounters an error - **consistent with log-ins, MFA attempts, and launching applications.**

Switch case: allows a developer to specify multiple possible conditions, each associated with a specific set of actions - **consistent with decision points, especially where data variances in exports exist.**

Note: *It is important to test each decision path. Run high volume tests to identify exceptions, and use development techniques to handle them.*

Managing exceptions and enhancements

Making in the moment choices to build or log as an enhancement.

Your team will undoubtedly run into exceptions or enhancement opportunities.

When this happens, ask yourself…

- Does the volume justify building it as an exception?
- Or, should it be removed from the process completely?

If you do decide to build…

- What questions need to be answered by subject matter experts?
  - How quickly can you get a concrete answer?
- What is the impact to your current sprint?

Quality Assurance

Two approaches to quality assurance reviews.

Characteristic	Peer-to-peer	Manager-to-peer
Relationship between reviewer and developer	Colleague	Manager
Focus	Technical quality	Technical quality and overall performance
Formality	Less formal	More formal
Benefits	Improved code quality Knowledge sharing Collaboration	Improved code quality Feedback on performance
Drawbacks	Can be difficult to remain objective Can be seen as threatening	Can be seen as micromanaging Can discourage feedback

Discussion:

- Does your team have a manager to review code?
- What does your team consider when choosing a peer, for peer-to-peer review?

After Action Review

Build a culture of continuous improvement.

By routinely conducting AARs and incorporating the learnings into operations, teams can improve their effectiveness and build confidence to take on the next challenge.

AARs follow a simple process:

Purpose and ground rules - Establish the purpose and ground rules for the conversation to build a culture of objectivity and openness.

Overview - Discuss the intent of the project, expected end state, the actual end state, the events, and relevant metrics.

Sustain - Discuss what went well and what to keep doing.

Improve - Make decisions about how to improve performance in the future.

Share and implement - Develop a plan for sharing the learnings broadly and implementing decisions.