RPA Developer foundation

Introduction to the RPA Developer Role

* Welcome to RPA
  + What is RPA/What does it do
    - Mimics human actions
    - Quick to implement and powerful to scale
    - Works without making mistakes and without rest
    - Operates any application
    - Reads and processes data in structured form
  + Automation First Approach
    - A robot for every person
    - Open and free collaboration
    - Robots learn skills
* UiPath Platform
  + Studio
    - Where processes are built
  + Orchestrator
    - Server application where robots are controlled managed and monitored
      * Connections with robots are created and maintained and robots are grouped (control)
      * Processes are distributed as tasks (management)
      * Execution of tasks logged and kept track of (monitoring)
  + Robot
    - Executes workflows and instructions sent locally or through orchestrator
    - 2 types
      * Attended: triggered by user events, works with human on same workstation
      * Unattended: run in virtual environments and can automate any number of processes
* What is a business process?
  + Definition: set of interrelated or interacting activities transforming inputs into outputs
  + Components:
    - Inputs: data going in
    - Process flows: sequences of sub-processes or activities in process
    - Source applications: applications or systems used to perform sub-processes or activities of process
    - Outputs: result generated by process
  + Things to remember: output can serve as input for another process
  + Procedure explains:
    - Who is responsible for each part
    - When each part needs to happen
    - How to handle exceptions
    - Specifications applicable to each part of the process
  + What makes a process a good candidate for automation?
    - Process fitness
      * Rule-based
        + Can use pre-defined logic
        + Exception rate is low or can be included in the business logic
      * Automatable and/or repetitive process
        + Manual and non-repetitive

Steps are performed by people and can be different every time process is executed

* + - * + Manual and repetitive

Steps are performed by user and at least some of them are the same every time

* + - * + Semi-automated and repetitive

Some repetitive steps are automated already

* + - * + Automated

Have already been automated using non-RPA technologies

* + - * Standard Input
        + Input should be electronic and easily readable or readable using RPA technology like OCR
      * Stable
        + Have been the same for a while and aren’t expected to change within next few months
        + Good candidates for automation assuming the rest as well
    - Automation complexity
      * Number of screens
        + Higher the number of screens higher the number of elements to be captured and configured before automation
      * Types of applications
        + Some applications are easier to automate than others
      * Business logic scenarios
        + Complexity increases with the number of decision points
      * Types and number of inputs
        + Standard input desirable
  + Assessing automation potential
    - No RPA
      * Change is frequent, system is volatile, multiple manual (even non-digital) actions required
    - Semi-Automation
      * Can be broken down into steps that can be automated and steps that must stay manual
    - High-Cost RPA
      * Digital and can be automated but use complex technologies (like OCR) or require advanced programming skills
    - Zero-Touch Automation
      * Digital and involve static system and process environment
      * Can be broken into instructions and simple triggers can be defined
* The RPA Journey
  + 6 stages
    - Prepare RPA
      * processes defined, assessed, prioritized, and implementation planned
    - Solution Design
      * Each process to be automated is documented (“as is” and “to be”)
      * Architecture created and reviewed
      * Test scenarios and environments prepared and solution design created and documented for each process
    - Build RPA
      * Processes automated workflow tested and validated and UAT (User Acceptance Test) prepared
    - Test RPA
      * UAT performed, workflow debugged and process signed off
    - Stabilize RPA
      * Go-Live prepared, process moved to production, monitored, measured, and lessons learned are documented
    - Constant Improvement
      * Process automation performance assessed benefits tracked and changes managed
  + Who will you be working with/roles?
    - Solution Architect
      * In charge of defining architecture of solution
      * Translates requirements captured by functional analysts, creating architecture and design artifacts
      * Lead, advise, and responsible for team delivery
    - Business Analyst
      * Responsible for mapping AS IS and proposed TO BE processes
      * Have knowledge of process to be automated, general business process theory, and RPA capabilities
      * Responsible for listing process requirements for automation, clarifying inputs and expected outputs, creating RPA documentation (Process Design Documents[PDD], Process maps, etc.)
      * Implementation Manager/Project Manager
        + Form and manage RPA team
        + Does resource planning and teams availability to hit automation goals
        + Most of the time is Single Point Of Contact (SPOC) for questions, RPA initiatives, or parallel RPA product projects
      * RPA developers
        + Several will collaborate to automate all processes on complex projects
      * Infrastructure & IT Security admin
        + Responsible for setting up and maintaining hardware & software resources for UiPath product installations
        + Set up accounts for all devs, end users, and robots
      * Process Owner
        + Key stakeholder and beneficiaries of RPA solution
        + Usually Senior Management with 10+ years of experience
        + Can be multiple people based on department
      * RPA support
        + Manage robots after process moved to production
        + With support from original devs who performed automation
        + May have multiple levels of support

Variables, Data Types, and Control Flow

* Variables & Arguments
  + Variables
    - Configured through properties which are found in variables and properties panels
    - Main properties
      * Name
        + Should be descriptive to make automation readable
      * Type
        + Declared when variable created
        + There are some types that are more generic and can accommodate different data types
      * Default Value
        + Only used if no initial value is assigned when created
      * Scope
        + Where a variable can be used
        + Unnecessary global variables can cause efficiency issues and a possibility for confusion
    - Creating Variables
      * From variables panel
        + Accessed by entering name in designer panel or desired properties field
      * From designer panel
        + Assign activity press Ctrl+K first and change properties in variables panel
      * From properties panel
        + Press Ctrl+K in output of activity in properties panel
      * Good Case Practices
        + Use clear and consistent naming conventions (typically Camel case)
        + Define scopes correctly (just where they are used not outside of that)
  + Arguments
    - Store data dynamically
    - Same types and same methods as variables
    - Difference is pass data between workflows and have directional property instead of scope (in, out, in/out)
* Data Types
  + Borrowed from VB.Net
  + Common types used:
    - Numeric (category)
      * Have sub-types (int32, long, double)
    - Boolean
    - Date and Time (category)
      * DateTime
        + Specific time coordinates
      * TimeSpan
        + Duration
    - String
      * Used to store text
    - Collection
      * Each object can be identified by index in collection
      * Largely used to handle and process complex data
      * Common types
        + Array: ArrayOf<T> or System.DataType[]

Store multiple values of same data type

Size defined at creation

* + - * + List: System.Collection.Generic.List<T>

Used to store values of same data type like arrays

Difference with arrays is size is dynamic

* + - * + Dictionary: System.Collections.Generic.Dictionary<TKey, TValue>

Used to store objects in form of (key, value) pairs

Pairs can have different data types but all keys and all values must have the same data types

* + - * + Generic Value

UiPath proprietary variable can store any kind of data

Mainly used when you don’t know what type of data you will receive

Use this temporarily

* + Array Variables
    - Type of variable that allows storing values of same type
    - When to use them?
      * Fixed collection of something has to be stored and used
      * Use when there is a predefined size use lists if the size of the collection needs to change
  + GenericValue Variables
    - Type of variable particular to UiPath that can store any kind of data
    - Automatic conversion mechanism of variables
      * First element in expression is same type as what GenericValue will be treated as
    - When to use them?
      * Data is extracted and forwarded without processing
      * 2 versions of same Excel file being compared column by column
        + Only relevant thing is which entries have changed not how they have changed
* Control Flow Overview
  + Order of execution or evaluation of individual statements, instructions, or function calls
  + 2 concepts by which flow is enacted
    - Type of project
      * 4 predefined types
        + Sequences

Have process flow in clear succession and decision trees are rarely used

Highly recommended for simple linear workflows

* + - * + Flowcharts

Slightly more difficult to read and edit but flows are clear

Used when decision points and branching are needed to accommodate complex scenarios, workarounds, and decision mechanisms

* + - Control Flow Statements
      * Activities and methods used to define decisions
      * Most common types
        + If/else

Condition with 2 potential outcomes

Based on condition what happens

Based on type of project use either If statement or Flow Decision

When to use it?

2 courses of action

* + - * + Loops

Repetitions of set of operations based on condition

Most important ones in UiPath

Do While

Executes specific sequence while condition is met

Condition evaluated after each execution

Always runs at least 1 time

While

Like do while but condition evaluated before each execution

May never run

For each

Executes on each element in collection

* + - * + Switch

Executes set of statements out of multiple based on value of expression

Used when need at least 3 potential courses of action

Data Manipulation

* What is it?
  + Process of modifying, structuring, formatting, or sorting through data to facilitate useage and increase management capabilities
* When to use it?
  + Extract and view specific information from web server logs
  + Allows you to extract only relevant data
* Strings
  + Text data type
  + Methods
    - Concat
      * Concatenates string of 2 specified objects
      * String.Concat(VarName1, VarName2)
    - Contains
      * Checks if substring within string
      * VarName.Contains(“text”)
    - Format
      * Converts value of objects to strings and inserts them into text
      * String.Format(“{0} is {1}”, VarName1, VarName2)
    - IndexOf
      * Returns 0 based index of first occurrence of character in string
      * VarName1.IndexOf(“a”)
    - Join
      * Concatenates elements in collection and displays them as String
      * String.Join(“|”,CollVarName1)
    - Replace
      * Replaces all occurrences of substring in a string
      * VarName.Replace(“original”,”replaced”)
    - Split
      * Splits string into substring using delimiter
      * VarName.Split(“|”c)(index)
    - Substring
      * Extracts substring from string using starting index and length
      * VarName1.Substring(startIndex, length)
* Lists
  + List<T>
  + Objects must have same data type
  + Objects can be indexed
  + Provides specific methods to manipulate objects
    - Add to collection: List.Add()
    - Remove from collection: outputs Boolean to confirm success
    - Exists in collection: outputs Boolean to confirm success or failure
    - Clear collection: clears collection
    - Can also loop through items using for each loop
    - Can be sorted
    - Items can be extracted and converted to another data type
* Dictionaries
  + Dictionary<TKey,TValue>
  + Keys must be unique
  + Key and value types can be different but must be consistent across dictionary
  + Methods
    - Initialization
      * Can be done in either the variables or designer panel
    - Adding
      * VarName.Add(Key,Value) adds item to dictionary
      * Does not return a value so must be used in invoke code not assign activities
    - Removing
      * VarName.Remove(Key) removes item
      * Can be done in assign activities
    - Retrieving
      * VarName.Item(Key) returns the dictionary item by key
      * VarName.Count returns Int32 value of number of items
      * VarName.ContainsKey(Key) checks if item with key exists in dictionary and returns Boolean
      * VarName.TryGetValue(Key,Value) checks if item with key exists and returns Boolean result and value if found
* RegEx Builder
  + Regular Expression (REGEX) is search pattern that can be used to match, locate, and manage text
  + Methods/Activities
    - Matches
      * Searches input string for all occurrences and returns all successful matches
    - IsMatch
      * Indicates whether or not the regular expression finds a match
    - Replace
      * Replaces strings matching regex with specified replacement

Excel and Data Tables

* What are datatables?
  + Variable that can store data in table form like spreadsheets
  + Allow pieces to be identified by unique column and row coordinates
  + How to create them?
    - Build data table
      * Choose number of columns and data types
      * Can add specific options like null or unique values, auto-increment, default value, and length
    - Read range
      * Reads data from worksheet storing it in datatable
    - Read csv
      * Reads csv file storing it in datatable
      * Not used much now but there are legacy and internal-built applications that use them
    - Data scraping
      * Extracts structured data from browser, application, or document
* DataTable activities
  + Add data column
    - Requires existing table
    - Can be column type or be added empty
    - Can specify data type and configure options
  + Add data row
    - Requires existing table
    - Can be data row type or array row
    - Each object must match corresponding column data type
  + Build data table
    - Creates table in window
    - Allows customization
  + Clear data table
    - Clears all data
  + Filter data table
    - Can use filter wizard using various conditions
    - Can create new table in output or delete entries not matching conditions
  + For each row
    - Loops through table
  + Generate data table
    - Creates table from unstructured data
    - User indicates row and column delimiters
  + Join data tables
    - Combines rows using common values according to join rule
    - How does it work?
      * 3 variables must be specified
        + 2 input and 1 output
        + Order of first 2 is important only 1 option keeps values from table 1 and cannot be changed
      * Join type has to be chosen 3 options
        + Inner

keep all rows meeting join rule

removes rows not meeting the rule

* + - * + left

keeps all rows from table 1 and only rows from table 2 matching rule

null values inserted for rows without matches

* + - * + Full

Keep all rows from both

Null values added into rows from both tables without match

* + - * Have to be configured can have multiple
        + One column from each table has to be specified by names, index, or ExcelColumn variables
        + Operator has to be chosen: =, !=, >, <, >=, <=
  + Lookup data table
    - Like vLookup
    - Allows searching for value in table returning row index
    - Can be configured to return value with coordinates (RowIndex and target column)
  + Merge Data table
    - Appends 2 data tables
    - Simpler than join data table
    - 4 predefined actions can be used on missing values
  + Output datatable
    - Writes table to string returns CSV
  + Remove data row
    - Removes row with input which can have index or data row variable
  + Remove duplicate rows
    - Removes duplicates from table only keeps first entry
  + Sort data table
    - Sorts ascending or descending based on values in column
* Workbooks and Common Activities
  + File Access Level
    - Workbook activities executed in the background
    - Doesn’t require Excel to be installed
    - Can be faster and more reliable for operations just not opening the file
    - Works only for .xlsx files
  + Excel App Integration
    - Opens excel
    - Works with .xls and .xlsm
    - Has some specific activities for csv
    - All activities can be visible or run in the background
    - Excel must be installed
    - Opens file and saves and closes it for each activity
  + Share some activities but Excel integration has more
  + Common activities
    - Append range
      * Adds information from datatable to end spreadsheet
      * Creates sheet if It doesn’t exist
    - Get table range
      * Locates and extracts range of table from spreadsheet using table name
    - Read cell
      * Reads content of cell storing it as string
    - Read cell formula
      * Reads formula from cell storing it as string
    - Read column
      * Reads column starting with cell inputted by user storing it as IEnumberable <object> variable
    - Read range
      * Stores range in datatable variable
      * If “use filter” is checked only filtered data will be read
      * Filter option only available for Excel Application Scope
    - Read row
      * Starts with cell input by user stores it as IEnumerable <object> variable
    - Write cell
      * Overwrites data in cell if it exists
    - Write range
      * Writes data from datatable starting with indicated cell
* Excel Application Scope and Specific Actions
  + Container and all activities used to work with file have to be inside container
  + Opens workbook providing scope for activities workbook and application closed when execution ends
  + Specific activities
    - CSV
      * Can read and write from CSV files with datatable variables
      * Work outside excel scope
      * Append to CSV
        + Adds info to file creating it if it doesn’t exist
        + Doesn’t overwrite existing data
      * Read CSV
        + Reads all entries storing them in datatable
      * Write CSV
        + Overwrites CSV with info from datatable
    - Range
      * Delete column: removes column based on name
      * Insert column: inserts blank column at position
      * Insert/delete columns: adds blank column or removes existing columns based on change type
      * Insert/delete rows: adds blank rows or removes existing rows based on change
      * Select range: generally paired with another activity performing manipulation on data
      * Get selected range: outputs range as string
      * Delete range: removes range
      * Auto fill range: applies formula over given range
      * Copy paste range: copies and pastes entire range from source sheet to destination sheet
      * Lookup range: searches for value in all cells in given range
      * Remove duplicate range: deletes duplicate rows
    - Table
      * Filter table
        + Applies filter on all values
        + Only rows meeting filter will be displayed
        + Does not delete entries not meeting criteria only hides them
      * Sort table
        + Sorts by column
      * Create table
        + Creates table in range specified in properties panel
    - File
      * Close and save workbook
    - Cell color
      * Get cell color: reads background color storing it as color variable
      * Set range color: changes background color of all cells in range; input is color variable
    - Sheet
      * Get workbook sheet: reads name by index
      * Get workbook sheets: extracts sheet names and stores them by index
      * Copy sheet: copies sheet and pastes it in same file or different specified file
    - Pivot table
      * Refresh pivot table:
        + refreshes table in excel
        + useful when pivot table source data changes as refresh is not automatic
      * create pivot table
        + creates table using specified sheet and given parameters
    - macro
      * work only with .xlsm files
      * execute macro: executes macros defined in file
      * invoke VBA: invokes macros from another file

UI Interactions

* Input Actions and Methods
  + Input Methods
    - Provide actions with means to input data
    - 3 available methods for each input action
    - Type used is picked in properties panel
    - Default
      * How it works
        + Clicks: mouse cursor moves
        + Typing: keyboard driver is used for each character
      * What are implications?
        + Attended user cannot touch mouse or keyboard
        + Lower speed and load times impact accuracy
      * What are strong points?
        + Supports special keys and hotkeys
        + 100% compatibility
      * What are limitations?
        + Does not automatically empty field
        + Does not work in background
    - Send Window messages
      * How it works
        + Replays window messages target application receives when mouse/keyboard used
        + Clicking and typing occur instantly
      * What are implications?
        + Works in background
        + Comparable to default in terms of speed
      * What are strong points?
        + Supports special keys and hotkeys
        + Users can work on other activities during execution of processes
      * What are limitations?
        + Does not automatically empty field
        + Works only with applications that respond to window messages
    - Simulate
      * How it works
        + Uses technology of application to send instructions (API level)
        + Clicking and typing occur instantly
      * What are implications?
        + Works in background
        + Faster but compatibility limitations
      * What are strong points?
        + Can empty field
        + Can work on other activities during process
      * What are limitations?
        + Does not support special keys and hot keys
        + Lower compatibility
  + Input Actions
    - Click, type into, send hotkey
    - All share several properties:
      * Delay: can be set before or after click
      * Wait for ready: can be configured to wait for target by verifying application tags
    - Click:
      * Clicktype: can be single or double
      * Mousebutton
        + configured to left middle or right button
        + timeout: specifies duration where activity is retried before error is thrown
      * key modifiers: can press alt, ctrl, shitft, and/or windows key while performing action
    - type into
      * Activate: Field where text is typed is brought to foreground and activated before typing
      * Click before typing: UI element where text is typed into will be clicked on first
      * Delay between keys: Delay between typing keys
      * Empty field: UI element emptied first
    - Send hotkey
      * Activate: field is brought to foreground and activated first
      * Click before typing: UI element will be clicked on before typing
      * Delay between keys: delay between typing
      * Empty field: UI element emptied first
  + Best Practices
    - Press F2 to pause for 3 seconds
    - Use send hotkey inside containers to avoid sending them to unintended places
* Output actions and methods
  + Output methods
    - Full text
      * Default method and usually good
      * Fastest and 100% accuracy also can work in background
      * Can extract hidden text
      * Does not support virtual environments or capture text position and formatting
      * Offers option to ignore hidden message and capture only visible text
    - Native
      * Compatible with applications using GDI (Microsoft API used to represent graphical objects)
      * Can extract text position and formatting including color
      * 100% accuracy on applications supporting GDI
      * Slower than full text and does not extract hidden text
      * Cannot work in background
      * Doesn’t support virtual environments
      * Can process all characters as separators but only when they are specified
    - OCR
      * Optical Character Recognition
      * Only output method that works with virtual environments and “reading” text from images
      * Relies on recognizing each character and position
      * Cannot work in background
      * Cannot extract hidden text
      * Slowest speed by far
      * Accuracy varies from one text to another and changing settings can improve result
      * Captures text position
      * 2 default engines (Google Tesseract and Microsoft MODI)
      * Additional engines can be installed for free or paid
    - Can switch between methods using screen scraping wizard in the preview window
  + Output Actions
    - Get Text
      * Extracts text value
    - Get full text
      * Extracts string and info from element using fulltext screen scraping method
      * Provides option to ignore hidden text otherwise extracts all
      * Automatically generated when using screen scraping with FullText method along with a container
    - Get visible text
      * Extracts string and info from element using native screen scraping method
      * Automatically generated along with container when performing screen scraping with native method chosen
    - Get OCR text
      * Extracts string and info from element using OCR screen scraping method
      * Automatically generated with container when OCR screen scraping method
      * Tesseract OCR engine is used by default
    - Data scraping wizard
      * Allows extraction of structured info from application, browser, or document to datatable variable
      * First element chosen is used to populate first column and option to extract URL is presented where it exists
      * Order of columns can be changed as can the maximum number of entries to be extracted
      * From preview can extract other data fields
    - Extract attributes
      * Get ancestor
        + Retrieves ancestor or parent of UI element
      * Get attribute
        + Allows user to indicate attribute and activity retrieves value of specific attribute
      * Get position
        + Retrieves actual position on screen of element
* Working with UI Elements
  + Find Element
    - Waits for element to appear on screen and in foreground before returning element variable
  + Element exists
    - Verifies if element exists even if not visible
    - Returns Boolean so good for if statements
  + Wait element vanish
    - Waits for element to disappear may be more reliable than appearance of another element when used for something like loading sign
  + On element appear
    - Waits for element to appear and enables user to perform actions within it
  + On element vanish
    - Enables you to perform 1+ actions after element vanishes
  + Text exists
    - Checks if text is found
    - Alternative OCR version
    - Useful when UI elements aren’t accessible other than as images

Selectors

* Introduction to selectors
  + Enables identification of elements based on its address and attributes
  + Done to perform specific activities
  + Generated automatically every time use an activity that interacts with GUI elements
  + Selectors must contain structured and hierarchized details
  + Structure of Selectors
    - UIs are built using series of containers nested inside eachother
  + Tags and Attributes of Selectors
    - Selectors made of nodes each of which is made of tags and attributes
    - Tags
      * Correspond to visual element on screen
      * First node is app window last node is element itself
    - Attributes
      * Has a name and value
      * Should only use attributes with constant or known values
* The UI Explorer
  + What is it?
    - Functionality that allows analyzing and editing selectors
    - Contains status button showing state of selector, visual tree panel showing navigable UI of each application at the moment, and selected UI element
    - Displays available tags and attributes giving option to check them in or out
  + UI Frameworks
    - Default
      * Proprietary method which usually works correctly with all types of UIs
    - Active accessibility
      * Earlier solution from Microsoft making apps accessible
      * Recommended when using legacy software if default does not work
    - UI automation
      * Improved accessibility model from Microsoft
      * Recommended when using newer applications and default does not work
  + Property Explorer
    - Displays all attributes of UI element including those not displayed in selector (position, visibility, innertext, etc.)
* Types of Selectors
  + Full selectors
    - Contain all tags and attributes needed including top-level window
    - Basic recorder
    - Best suited when actions require switching between multiple windows
  + Partial selectors
    - Don’t contain tags and attributes of top-level window
    - Must be enclosed in container
    - Desktop recorder
    - Best for multiple actions in same window
* Fine-tuning
  + Required in certain situations:
    - Dynamically generated selectors: values of attributes change with each visit
    - Selectors being too specific: automatically generated with name of file or value that changes
    - System changes: contain version of application or another element that changes when application is updated
    - Selectors using IDX: index of current element in container with similar elements
      * May change when new element appears in same container
  + What is it?
    - Process of refining selectors to have workflow correctly executed where generated selector is unreliable, too specific, or too sensitive in regards to system changes
    - Consists of small simple changes with large impact on overall process
      * Add wildcards, use repair function, or use variables in selectors
* Managing Difficult Situations
  + Tools to help fix selectors
    - Anchor base
      * Attribute values not reliable but a UI element is stable and linked to target element
      * 2 parts 1 to locate anchor element and 1 to perform activity
    - Relative selector
      * Incorporates info about anchor’s selector in selector of UI element
      * New selector will probably need additional editing
      * Have to have part like dynamic ID removed
      * Selector stabilizes using anchor’s selector
    - Visual tree hierarchy
      * Can improve reliability of selector including tags and attributes of element
      * Useful when target element’s selector not reliable but selector right above is
      * Needs further editing as dynamic part needs to be removed and need to make sure target element can be identified with unique attribute
    - Find children
      * Can identify all children of element that is more stable
      * Outputs collection of children so must come up with a way to identify only element you want

Project Organization

* How to choose best project layout
  + Sequence
    - When to use it?
      * Clear succession of steps without many conditions
      * Used to nest workflows and high-level logic handled through flowcharts or state machines
    - What are advantages?
      * Easy to understand and follow having top down approach
      * Great for simple logic
    - What are disadvantages?
      * Nesting too many conditions in same sequence makes process hard to read
      * Not suitable for continuous flows
  + Flowchart
    - When to use it?
      * Complex flow with several conditions
      * Flow runs continuously or terminates only in several conditions
    - What are advantages?
      * Easy to understand
      * Can be used for continuous workflows
    - What are disadvantages?
      * Can be used only as general workflow (with sequences nested inside)
      * Cant be used for individual parts of projects (nested in other workflows)
  + State machine
    - When to use it?
      * Abstract machine consisting of finite- number of pre0defined states and transitions between states
      * Process can only be in 1 state at a time based on external inputs and conditions
      * Can be used with finite number of clear and stable states to go through
    - What are advantages?
      * Can be used for more complex continuous workflows
      * Transitions can be easily defined and offer flexibility
      * Can accommodate processes that are more complex and cannot be captured by loops and if statements
      * Easier to cover all possible cases/transitions
    - What are disadvantages?
      * Longer development time due to complexity: splitting process into states, figuring out transitions, etc.
      * Should only define skeleton of the project
    - Important notes
      * Naming states is important for maintenance and future development
      * Order of transitions shown in each state is important as it matches order they are evaluated in
* How to break down a complex process
  + 3 factors should be considered as breakdown criteria:
    - Application being automated
    - Purpose of certain operation
    - Length of workflows
  + Handling data
    - Multiple workflows requires arguments
    - Arguments store data dynamically, have same types, and support same methods as variables
    - Direction is main difference between variables and arguments
      * In: used if you want to use value of argument inside invoked workflow
      * Out: if want to pass argument outside of invoked workflow
      * In/out: want to pass variable from parent workflow to invoked workflow, modify it in invoked workflow and pass it back
    - Extract as workflow automatically turns variables into arguments
    - Need to use import arguments first time invoking workflow
    - Make sure to instantiate arguments with corresponding values of variables in arguments panel
  + Good case practices- using arguments
    - Use direction as prefix when naming and renaming arguments
* How to reuse parts of a project
  + Process library: package containing multiple reusable components consisting of 1+ workflows acting as individual activities
    - Saved as nupkg files and installed in different workflows using Package Manager
  + Good case practices-Libraries
    - Each workflow can be set as private or public
    - Public can be used as activities in projects where libraries are added while private can only be used inside libraries
* How to manage versions of the same project
  + Git
    - Clone repository, add project, commit and push, copy project to Git, create and manage branches, solve conflicts with File Diff option
  + TFS
    - Have to set up TFS then can open project or add new project
  + SVN
    - Open and add project
* How to prevent and solve exceptions?
  + Predicting and treating exceptions can be done in 2 ways
    - Activity level using Try/Catch blocks or Retry Scope
    - At global level using Global Exception Handler
  + Categories of exceptions
    - Application exception
      * Error rooted in technical issue (application not responding)
      * Can be solved by retrying transaction
    - Business exception
      * Error rooted in problems with data such as incomplete, missing, outside of set boundaries or does not pass data validation criteria
* Best Practices
  + Analyze process and identify requirements and planning how solution should look before starting
  + Break process into smaller workflows for better understanding, independent testing, reusability; can also impact effectiveness
  + Group workflows into different folders based on application
  + Keep consistent naming convention
  + Use right type of argument when invoking workflow based on direction of information
  + No credentials should be stored in workflow directly but loaded from safer places
  + Use try/catch blocks to predict and handle exceptions (only finds error does not solve it)
  + Use global exception handler for less probable errors or errors that are project wide
  + Use libraries to create and store reusable components
  + Add annotations and use logs to get relevant information

Error and Exception Handling

* Common exceptions
  + NullReferenceException: usually occurs when using variable with no set value
  + IndexOutOfRangeException: index of object is out of the limits of the collection
  + ArgumentException: method is invoked and at least one passed argument does not meet parameter specification of method
  + SelectorNotFoundException: robot can’t find designated selector for activity in target application within TimeOut period
  + ImageOperationException: image not found within TimeOut period
  + TextNotFoundException: indicated text not found within TimeOut period
  + ApplicationException: error rooted in technical issue
  + Business rule exceptions are separate
    - Will not be thrown using generic System.Exception in Try Catch activity
    - Has to be handled separately from try catch or it can stop execution of process using Thro activity
* Try Catch
  + Either displays error notification or dismisses it continuing execution
  + Finally is only executed when no exceptions are thrown or when exception caught and handled without being rethrown
  + Multiple errors can be caught and lead to different activities
* Retry Scope
  + Retries contained activities as long as condition not met or error thrown
  + Used to catch and handle error similar to try catch difference being this retries instead of doing something else
  + Can be used without termination condition
  + Additional properties
    - Number of retries: number of times sequence is to be tried
    - Retry interval: specifies amount of time between retries
* ContinueOnError
  + Specifies execution should continue when activity throws error
  + If set on activity with a scope all errors that occur in other activities inside scope ignored
  + Not recommended in all situations but is recommended when:
    - Data scraping so error not thrown on last page when “Next” selector is no longer found
    - Not interested in capturing error just the execution of activity
  + Default value is false
* Global Exception Handler
  + Workflow designed to determine behavior when encountering execution error at project level
  + Only 1 can be set per automation project
  + Only uncaught exceptions reach handler
  + How does it work?
    - 2 predefined arguments that shouldn’t be removed:
      * errorInfo with In direction: contains error thrown and workflow that failed
      * result with Out direction: stores next behavior of process when it encounters error
    - actions that can be added:
      * log error: logs error, developer chooses logging level
      * choose next behavior
        + Continue: exception re-thrown
        + Ignore: exception ignored, execution continues from next activity
        + Retry: activity throwing exception retried
        + Abort: execution stops after running handler

Debugging

* Overview
  + Debugging actions and panels
    - Actions
      * Step into
        + Debugger highlights activity before executing it
        + Shortcut is F11
      * Step over
        + Skips current container and moves onto next
        + F10 shortcut
      * Step out
        + Pauses execution at container level
        + Completes execution of activities in container before pausing
      * Focus
        + Helps return to current breakpoint or activity causing an error
        + Can be used to return to breakpoint after navigating through other activities when debugging is paused
        + When debugging paused after using step into or step over and navigating through process, returns to activity pausing debugging process
      * Test Activity
        + Used to run test on current selected activity
        + Opens locals panel displaying variables and arguments in scope
        + Can be used in 2 ways

Add default values to properties and test

Add arguments and/or properties to activity properties and use local panel to add values after clicking option

* + - Debugging Panels
      * Locals
        + Only visible during debugging
        + Shows:

Exceptions: description and type

Arguments and variables can be modified when paused

Properties of previous and current activity (only input and output displayed)

Can add stuff to watch panel to monitor it throughout execution

* + - * Immediate
        + Only visible during debugging
        + Can be used to inspect data available at certain point, evaluate variables, arguments, or statements
        + Keeps history of previous statements which can be removed
      * Watch
        + Only visible during debugging
        + Can display values of variables, arguments, and user-defined expressions
        + Updated after each activity execution
      * Designer
        + Displays workflow, breakpoints and highlights current activity in yellow and activities throwing exception in red
      * Call Stack
        + Displays next activity and parent containers when paused in debugging
        + Displayed during execution and populated after using step into, break, slow step, or after execution paused due to error or breakpoint encountered
        + Focus and highlight activity by double clicking

Orchestrator for Developers

* Introduction to Orchestrator
  + What is orchestrator?
    - Where workflows are published, assigned to robots, and executed
    - Web application enabling management of robots, activity packages, data to be processed, execution schedules, etc.
    - Ideal for large deployments of robots covering complex processes but can be deployed when dealing with short and repetitive processes and fewer robots
  + What are the capabilities?
    - Provisioning: creates and maintains connection with robots
    - Deployment: ensures delivery of workflows for execution immediately or using schedules
    - Configuration: enables creation, configuration, and maintenance of robots and execution of tasks
    - Queues: data to be processed which is broken down to indivisible operations (transactions); can store any number and facilitate distribution, execution, and monitoring
    - Monitoring: keeps track of robot identification data and maintains user permissions
    - Logging: stores and indexes logs to SQL database and/or ElasticSearch
    - Inter-connectivity: centralized point of communication for 3rd party solutions/applications and can be used for storing packages, libraries, and other assets
  + What are benefits?
    - Accessibility and version control
      * Workflows can be published and stored at version level with release notes
      * Different versions can be distributed to robots for execution
      * Libraries can be published and stored so they can always be accessed, used in development, and distributed to robots
    - Transactional processing
      * Queues allow efficient allocation between robots, continuous execution, and monitoring at transaction level
      * If transaction can’t be executed for some reason it wont stop or delay execution
    - Efficient planning and execution
      * Allows execution of tasks according to schedules and can accommodate various scenarios
      * Allows choosing execution targets from available robots or create continuous loops even outside business hours
    - Securely storing assets
      * Configuration assets, credentials, and data changing frequently must not be stored in workflows
      * Provides secure way to store and distribute assets according to different scenarios
    - Monitoring
      * Robots processes and task execution can be monitored
      * Can enable quick reaction if an error pops up and allows for accurate reporting and auditing of the robot work
* Orchestrator key concepts
  + Robot: enables running workflows
  + Environment: group of robots configured together
    - Processes can be allocated to individual robots but more effective to allocate them to environments
    - Each robot can be part of multiple environments as long as they are in same service
  + Organization Unit: entity corresponding to business unit
    - Same instance can have multiple organization units configured each with separate robots, environments, queues, assets, etc.
  + Package: published project; multiple versions can be stored and used
  + Process: package allocated to environment
  + Job: process that has been sent for execution
  + Schedule: process configured for execution to run at a time; multiple configurations are possible
  + Asset: piece of data stored for use of robots 4 types
    - Text: stored only strings
    - Bool: supports true or false
    - Integer: stores whole numbers
    - Credential: contains usernames and passwords robot requires to execute particular processes
  + Queue: sequence of transactions built in orchestrator and used to dispatch to robots
* Robots and environments in orchestrator
  + What is a robot?
    - Enables running workflows
    - Robot triggered when run is clicked
  + Types of robots
    - Attended
      * Triggered by user events and operates alongside a human on same workstation
      * Can be used for centralized process deployment
    - Unattended
      * Run in virtual environments and execute any number of processes
      * Covers execution, monitoring, scheduling, and providing support for queues
    - Development
      * Has features of unattended bot but should only be used to connect Studio and Orchestrator for development purposes
    - Non-production robot
      * Similar to unattended bot but should only be used for development and testing
  + Kinds of robots
    - Standard: works only on single standard machine; good choice when machine known
    - Floating: works on any machine defined; good choice when people work in shifts on same machine or different machines and when virtual machines being regenerated
  + Provisioning robots
    - Robots and machines have to be provisioned before they can perform jobs
    - Machine name mandatory for standard robots
    - Log in credentials may be needed for unattended robots
    - Robots can be connected:
      * Directly from orchestrator settings window
      * From command line
      * Automatically enrolled
  + Environments
    - Grouping of robots used to deploy processes
    - Robots can be added to multiple environments
    - Services have robots, environments, queues, processes, etc. totally separated
  + Important notes
    - Community edition doesn’t allow multiple services
    - Services allow separation of all data under same orchestrator instance
    - Template machines should be used when name changes such as for virtual desktop infrastructure
* Process Execution in Orchestrator
  + Packages
    - Consist of multiple automation workflows
    - Can be published on local machine or directly to orchestrator
    - Can be manually uploaded
    - Versions care stored automatically and release notes can be accessed anytime
      * Can be active or inactive
        + Active: deployed to 1+ environments
        + Inactive: not deployed (can be deleted)
  + Processes
    - Represents association between package and environment
    - Any arguments Main.xaml has are displayed on parameters tab and become input and output parameters that can be configured
  + Jobs and Schedules
    - 3 ways to trigger execution of process
      * Use jobs (immediately)
        + Job allocated to robots has priority over dynamically allocated jobs but dynamically allocated jobs are executed as soon as robot becomes available
        + Can set input parameters or display out parameters if they exist
        + 2 options to stop a job:

Use stop which will stop when should stop activity is encountered

Use kill job to stop job immediately

* + - * Use schedules (planned)
        + Offer multiple configuration options:

Regular interval to execute process

Setting up parameters

Selecting robots to execute (similar to jobs) – all robots, selected robots, or dynamically

Setting up non-working days and other parameters

* + - * From robot tray
        + Make sure package is assigned to environment attended robot is part of
        + Open robot
        + Update robot if available
        + Run processes by clicking “play”
* Assets
  + What are they?
    - Shared variables or credentials stored in orchestrator and used by robots
    - Data repository robots can access when running processes based on instructions
    - 4 types
      * Text: equivalent of string
      * Bool: true/false
      * Integer: whole numbers
      * Credential: contains usernames and passwords
  + How are they created and used?
    - In orchestrator
      * Can be created from area
      * Name and type have to be provided using tabs
      * Can be configured in 2 ways:
        + Single value accessed by all robots
        + Value per robot: each value provided can be accessed only by indicated robot
      * Can be modified or deleted
    - In studio
      * Use “Get Credential” activity to create credentials
      * Use “Get Asset” for all other types of assets
* Queues
  + What are queues?
    - Containers holding unlimited number of items
    - Store items and allow distribution for processing, and monitoring status based on outcomes
    - Advantages:
      * Centralized depository of work items
      * Reporting capabilities at individual item and queue level
      * Effectiveness of item distribution process when robot becomes available, item is dispatched
      * Unitary logic of item distribution: FIFO
  + Working with queues
    - Set maximum number of retries and unique reference field when creating queue as these cannot be modified
    - Important for Dispatcher & Performer model which has 2 stages
      * Stage where data taken and fed into queue from where it can be taken and processed by robots (Dispatcher)
      * Stage where data is processed (Performer)
    - Main activities
      * Add queue item: robot sends item to queue and configures time frame and other parameters
      * Add transaction item: adds item in queue starting transaction
        + Queue item cannot be sent for processing until robot finalizes activity and updates status to “In progress”
      * Get transaction item: gets item from queue to process, sets status to “In progress”
      * Postpone transaction item: adds time parameter between which transaction must be processed
      * Set transaction progress: custom progress statuses can be set for “in progress” transactions
      * Set transaction status: changes status of item to failed (with application/business exception) or successful
        + Transactions failing due to application exceptions will be retried
        + Possible statuses

New: just added (or item was postponed or deadline added to it)

In progress: item processed with Get/Add Transaction Item

Failed: did not meet business or application requirement

Successful: item processed

Abandoned: item was “in progress” for too long without being processed (24 hours ish)

Retried: item failed with application exception and was retried

Deleted: item manually deleted

Robotic Enterprise Framework Overview

* Transaction Processing
  + What is a transaction?
    - Represents minimum (atomic) amount of data and necessary steps to process data to fulfill section of business process
    - Assume data is no longer needed going forward after processed
  + Can divide business process into 3 categories:
    - Linear
      * Steps performed only once
      * Automation needs to be executed again if there is more data to process
      * Usually simple and easy to implement but not suitable for situations requiring repetitions using different data
    - Iterative
      * Steps performed multiple times with different data each time
      * Can be implemented with loop but if a problem occurs while processing 1 item the whole process is interrupted and other items cant run
    - Transactional
      * Steps repeat multiple times over different data
      * Automation designed so each repeatable part processed independently
      * Repeatable parts called transactions which are independent as they do not share any data or have order to be processed
* The REFramework
  + What is it?
    - Implemented as state machine where workflow has useful features:
      * States define actions to be taken according to input
      * Transitions that move execution between states depending on outcomes
    - 4 main states common to business processes
      * 1 initial state
        + where process initializes settings and performs application checks to make sure all prereqs for starting process in place
      * 2 get transaction data state
        + Gets next item (can be a queue item or any item of collection)
      * 3 process transaction state
        + Performs actions/applies logic in applications for transaction item obtained earlier
        + Continues with the next available item once transaction processed
      * 4 end process state
        + Process ends (all opened applications should be gracefully closed)
  + Features
    - Settings
      * Common to have certain settings and configuration values read during initialization phase
      * Keeps track of data by reading from config file shared among different states
      * Config file offers easy way to maintain projects by changing values in file not workflows directly
    - Logging
      * Built-in mechanism and framework template has log message activities
      * Can be used to create visualizations and reports about execution of process
    - Business exception and application exception
      * Need to be addressed for more robust automation
      * Application exceptions may be solved by restarting applications while business exceptions will never pass without changing the business rule or data input
* Dispatcher and Performer
  + Dispatcher
    - Process to push transaction items to queue
    - Extracts data from source(s) and uses it to create items to be processed
    - Advantage: can split processing of items between multiple robots
  + Performer
    - Used to pull transaction items from queue and process them
    - Processed one at a time
    - Uses error handling and retry mechanisms for each item
    - Advantage: scalability (multiple performers can be used with single queue)
  + Model advantages
    - Better separation of processes
    - Better separation and distinction between architecture and process layers
    - Better error handling and retry mechanism
    - Can run processes across multiple machines (availability)
    - Better re-usability within projects created components
    - Built-in configuration and orchestrator integration
    - Previous workflows can be adapted and deployed to use REF and this model