RANOREX MIGRATION TO ASCENTIALTEST

Matryxsoft Tech LLP
Sept 2023

Background and Introduction:

One of the customers having PowerBuilder application used Ranorex tool to automate around 3000 Testcases, over a period of time they had controls identifying issue for the latest version of PowerBuilder which led to maintenance issues. They were looking for an alternative testing tool and approached Zeenyx who owns AscentialTest™ (AT) Product, known as the Next Generation Management tool which stands as No.1 support for PowerBuilder applications.

Zeenyx knew about Matryxsoft Tech's experience of migrating UFT, TOSCA, Selenium code to AT code and also about the migration techniques of Ranorex to AT. Hence, they recommended the customer to approach us for migration of their entire Ranorex Project.

Matryxsoft Tech, an independent software testing company well known for its specialization in Migration and localization techniques had analyzed the requirement from the customer and worked towards the same with its partner Zeenyx and migrated the Ranorex code to AT code which gave more robust and reliable tests with an increased performance.

The purpose of this paper is to provide a detailed understanding on how the migration was successfully performed from Ranorex to AT w.r.t to Object Repository, Datatable, Steps/Functions and Tests.

Setup:

AscentialTest Tool and Project built on Ranorex are the basic requirement for migration.

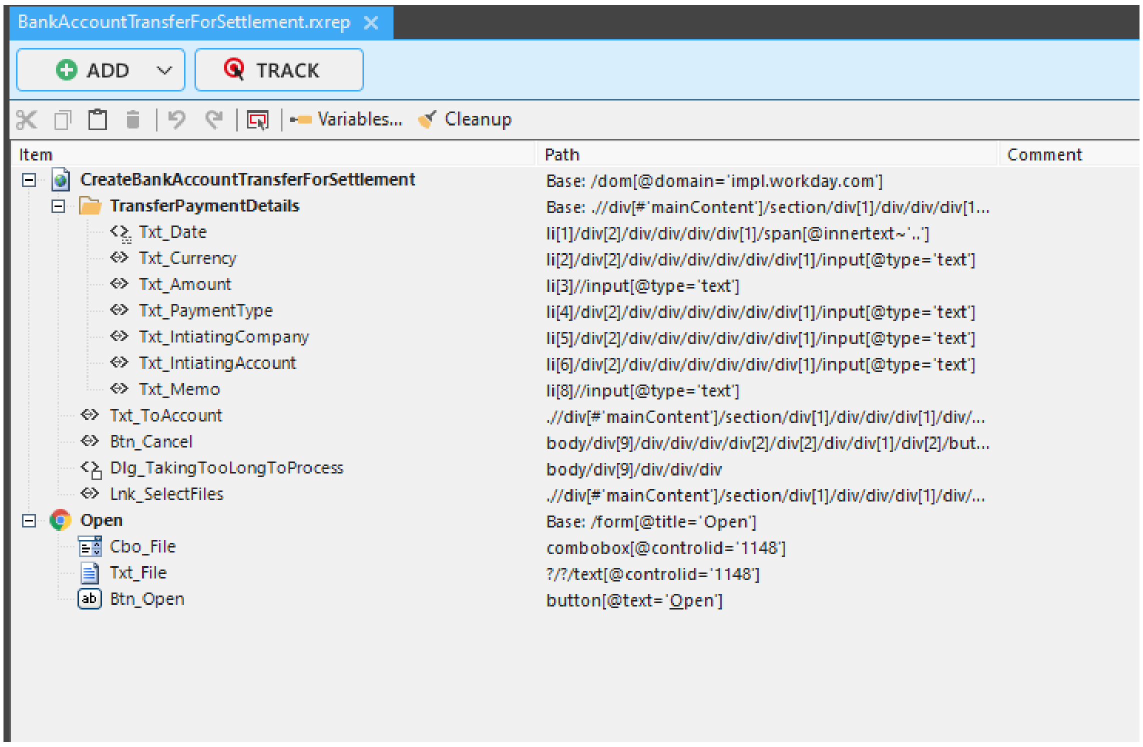
All the required files are segregated w.r.t to extensions according to the conversion required to AscentialTest. One can go right away for the migration with the utilities that are prepared for migration but for a better migration rate, it is good to understand the Ranorex architecture and the framework built around for the application under test.

Application Knowledge and the manual flow testcase document will help in fine tuning the migration at a faster rate.

AppObjects to AppObjects:

Ranorex Objects are recognized by the identified UI elements are stored in the Object Repository. The repository acts as a collection of the recognized objects along with their properties and hierarchical relationships. At first all the files related to the app objects are copied and are converted to the AscentialTest AppObjects through the conversion utilities created. The conversion utilities handle all the necessary indentation and the syntax changes which brings exactly the way AscentialTest requires. The challenges are seen on the indentation of objects with multiple properties and some of the attributes which are not exactly the same with AT are handled through our utilities and brings them to the AscentialTest AppObjects Format. Please find the example of AppObjects conversion as shown in the AT Screenshot.

Ranorex AppObjects:



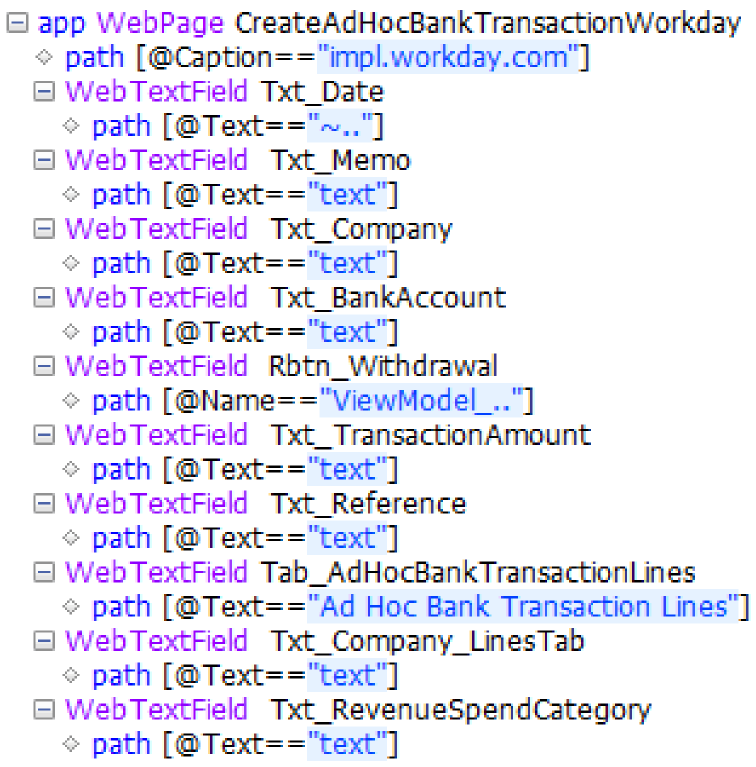
BankAccountTransferForSettlement.rxrep

ADD TRACK

Variables... Cleanup

Item	Path	Comment
CreateBankAccountTransferForSettlement	Base: /dom[@domain='impl.workday.com']	
TransferPaymentDetails	Base: .//div[#'mainContent']/section/div[1]/div/div/div[1...	
Txt_Date	li[1]/div[2]/div/div/div/div[1]/span[@innertext~'..']	
Txt_Currency	li[2]/div[2]/div/div/div/div/div/div[1]/input[@type='text']	
Txt_Amount	li[3]//input[@type='text']	
Txt_PaymentType	li[4]/div[2]/div/div/div/div/div/div[1]/input[@type='text']	
Txt_IntiatingCompany	li[5]/div[2]/div/div/div/div/div/div[1]/input[@type='text']	
Txt_IntiatingAccount	li[6]/div[2]/div/div/div/div/div/div[1]/input[@type='text']	
Txt_Memo	li[8]//input[@type='text']	
Txt_ToAccount	.//div[#'mainContent']/section/div[1]/div/div/div[1]/div/...	
Btn_Cancel	body/div[9]/div/div/div/div[2]/div[2]/div/div[1]/div[2]/but...	
Dlg_TakingTooLongToProcess	body/div[9]/div/div/div	
Lnk_SelectFiles	.//div[#'mainContent']/section/div[1]/div/div/div[1]/div/...	
Open	Base: /form[@title='Open']	
Cbo_File	combobox[@controlid='1148']	
Txt_File	?/?/text[@controlid='1148']	
Btn_Open	button[@text='Open']	

AscentialTest AppObjects:



```
app WebPage CreateAdHocBankTransactionWorkday
  path [@Caption=="impl.workday.com"]
  WebTextField Txt_Date
    path [@Text=="~.."]
  WebTextField  Txt_Memo
    path [@Text=="text"]
  WebTextField  Txt_Company
    path [@Text=="text"]
  WebTextField  Txt_BankAccount
    path [@Text=="text"]
  WebTextField  Rbtn_Withdrawal
    path [@Name=="ViewModel_.."]
  WebTextField  Txt_TransactionAmount
    path [@Text=="text"]
  WebTextField  Txt_Reference
    path [@Text=="text"]
  WebTextField Tab_AdHocBankTransactionLines
    path [@Text=="Ad Hoc Bank Transaction Lines"]
  WebTextField  Txt_Company_LinesTab
    path [@Text=="text"]
  WebTextField  Txt_RevenueSpendCategory
    path [@Text=="text"]
```

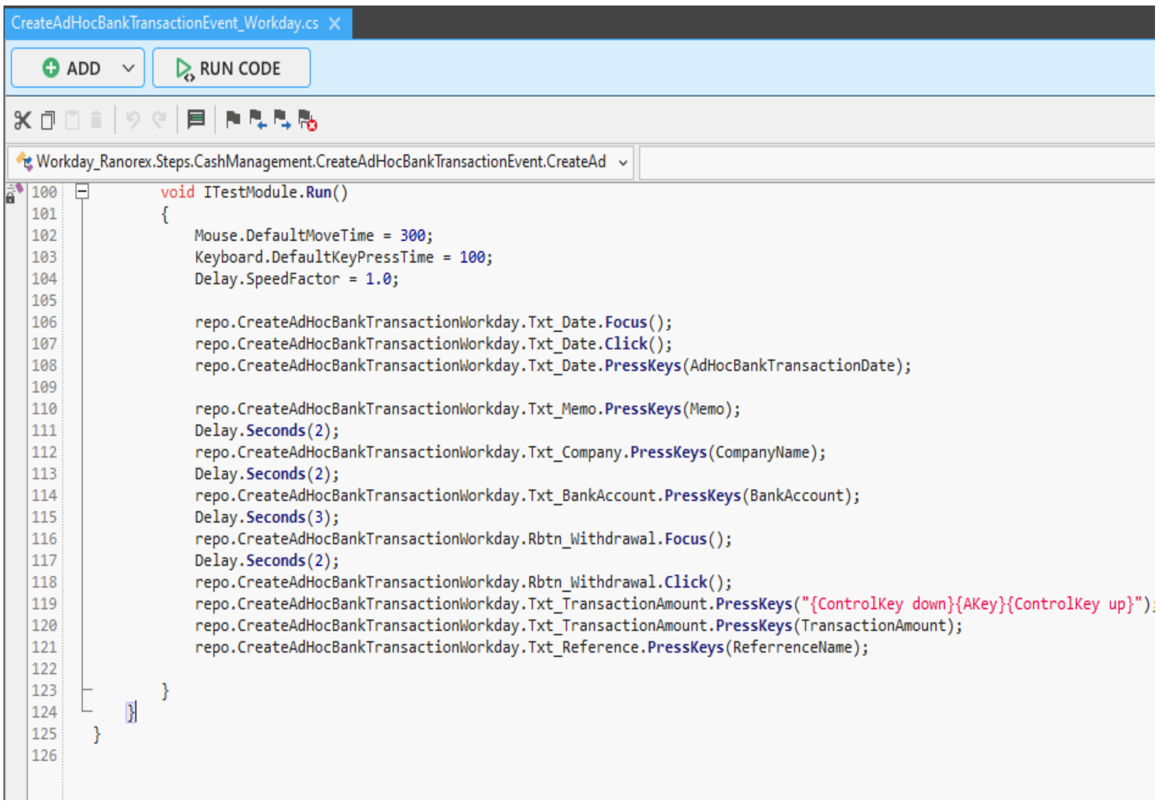
Challenges:

- Working on indentation on two – three levels and so on.
- Parent and child object identification in Ranorex AppObjects is quite a challenge. Parent object opening tags and child object closing tags are quite similar.
- It is difficult to transform Ranorex properties into AT Properties. Some Ranorex properties differs from AT properties.

Steps to Steps:

Ranorex functions are written on multiple files that are created w.r.t features or modules. There could be 'n' number of functions with 'n' number of parameters which may or may not have return data type. For all the functions which returns more than one datatype are kept as a function and the one with no return type or one return datatype are converted to steps in AscentialTest. Numerous challenges were faced while migrating the functions on the datatypes, parameters, function calls, Subfunction calls, indentation, return datatypes and reading input data. All these challenges were successfully handled using the steps and function utilities created by Matryxsoft. There are multiple utilities that were written to handle the conversion techniques. With the help of this Utilities, we converted Actions of Ranorex functions to AT Steps. A Sample of the Action written in Ranorex is migrated to AT as shown in the below picture.

Ranorex Steps:



CreateAdHocBankTransactionEvent_Workday.cs
ADD
RUN CODE
Workday_Ranorex.Steps.CashManagement.CreateAdHocBankTransactionEvent.CreateAd

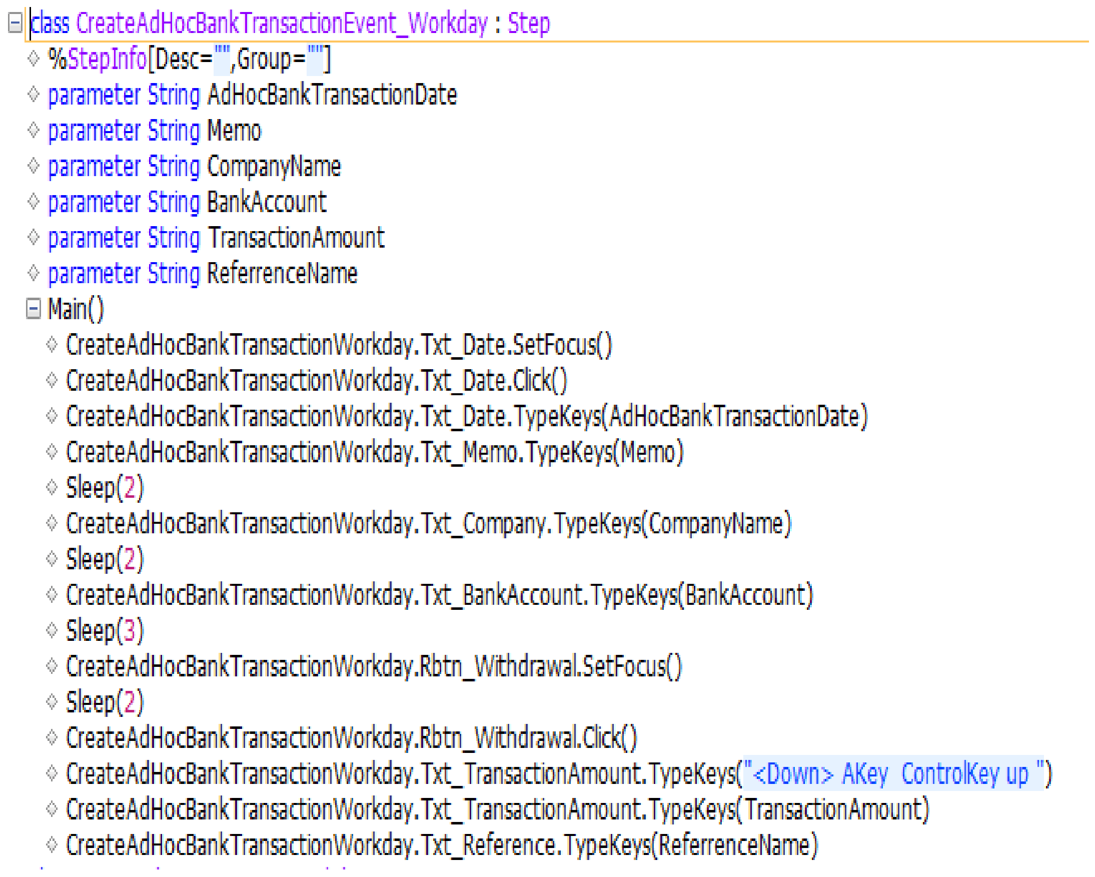
```
        void ITestModule.Run()
        {
            Mouse.DefaultMoveTime = 300;
            Keyboard.DefaultKeyPressTime = 100;
            Delay.SpeedFactor = 1.0;

            repo.CreateAdHocBankTransactionWorkday.Txt_Date.Focus();
            repo.CreateAdHocBankTransactionWorkday.Txt_Date.Click();
            repo.CreateAdHocBankTransactionWorkday.Txt_Date.PressKeys(AdHocBankTransactionDate);

            repo.CreateAdHocBankTransactionWorkday.Txt_Memo.PressKeys(Memo);
            Delay.Seconds(2);
            repo.CreateAdHocBankTransactionWorkday.Txt_Company.PressKeys(CompanyName);
            Delay.Seconds(2);
            repo.CreateAdHocBankTransactionWorkday.Txt_BankAccount.PressKeys(BankAccount);
            Delay.Seconds(3);
            repo.CreateAdHocBankTransactionWorkday.Rbtn_Withdrawal.Focus();
            Delay.Seconds(2);
            repo.CreateAdHocBankTransactionWorkday.Rbtn_Withdrawal.Click();
            repo.CreateAdHocBankTransactionWorkday.Txt_TransactionAmount.PressKeys("{ControlKey down}{AKey}{ControlKey up}");
            repo.CreateAdHocBankTransactionWorkday.Txt_TransactionAmount.PressKeys(TransactionAmount);
            repo.CreateAdHocBankTransactionWorkday.Txt_Reference.PressKeys(RefferenceName);

        }
    }
}
```

AscentialTest Steps:



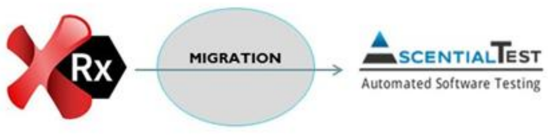
```
class CreateAdHocBankTransactionEvent_Workday : Step
  %StepInfo[Desc="",Group=""]
  parameter String AdHocBankTransactionDate
  parameter String Memo
  parameter String CompanyName
  parameter String BankAccount
  parameter String TransactionAmount
  parameter String ReferrenceName
  Main()
    CreateAdHocBankTransactionWorkday.Txt_Date.SetFocus()
    CreateAdHocBankTransactionWorkday.Txt_Date.Click()
    CreateAdHocBankTransactionWorkday.Txt_Date.TypeKeys(AdHocBankTransactionDate)
    CreateAdHocBankTransactionWorkday.Txt_Memo.TypeKeys(Memo)
    Sleep(2)
    CreateAdHocBankTransactionWorkday.Txt_Company.TypeKeys(CompanyName)
    Sleep(2)
    CreateAdHocBankTransactionWorkday.Txt_BankAccount.TypeKeys(BankAccount)
    Sleep(3)
    CreateAdHocBankTransactionWorkday.Rbtn_Withdrawal.SetFocus()
    Sleep(2)
    CreateAdHocBankTransactionWorkday.Rbtn_Withdrawal.Click()
    CreateAdHocBankTransactionWorkday.Txt_TransactionAmount.TypeKeys("<Down> AKey  ControlKey up ")
    CreateAdHocBankTransactionWorkday.Txt_TransactionAmount.TypeKeys(TransactionAmount)
    CreateAdHocBankTransactionWorkday.Txt_Reference.TypeKeys(ReferrenceName)
```

Challenges:

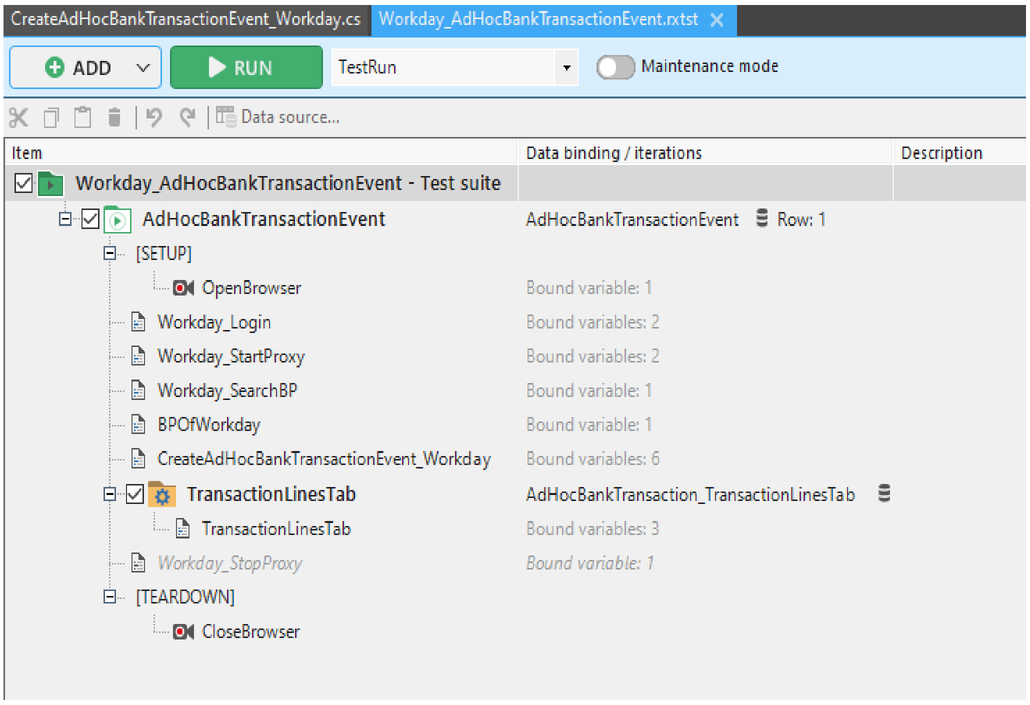
- Indentation of Converting Ranorex Actions to AT Steps is challenging.
- Handling parameters and function return datatype.
- Ranorex native methods are not straight forward as AT, hence need to develop own functions/utilities.

Test to Test:

Creating a Ranorex test involves several functions, including recording test scenarios, enhancing test functions, managing the Ranorex Repository, and executing the test. Calling its actions in sequence and every test is associated with the inbuilt Datatable by global sheet or external database like Excel and also has its own Setup and Teardown. In AscentialTest, we have a concept of AppState that has OnStart and OnFinish which works similar way of Ranorex. The challenges faced in the test were related to Function call, Indentation and Parametrization. All these were handled by the test utilities that were written by us in order to convert the tests of Ranorex to AscentialTest Test format. The successful transition of the tests is as shown in the below pictures.



Ranorex Test:



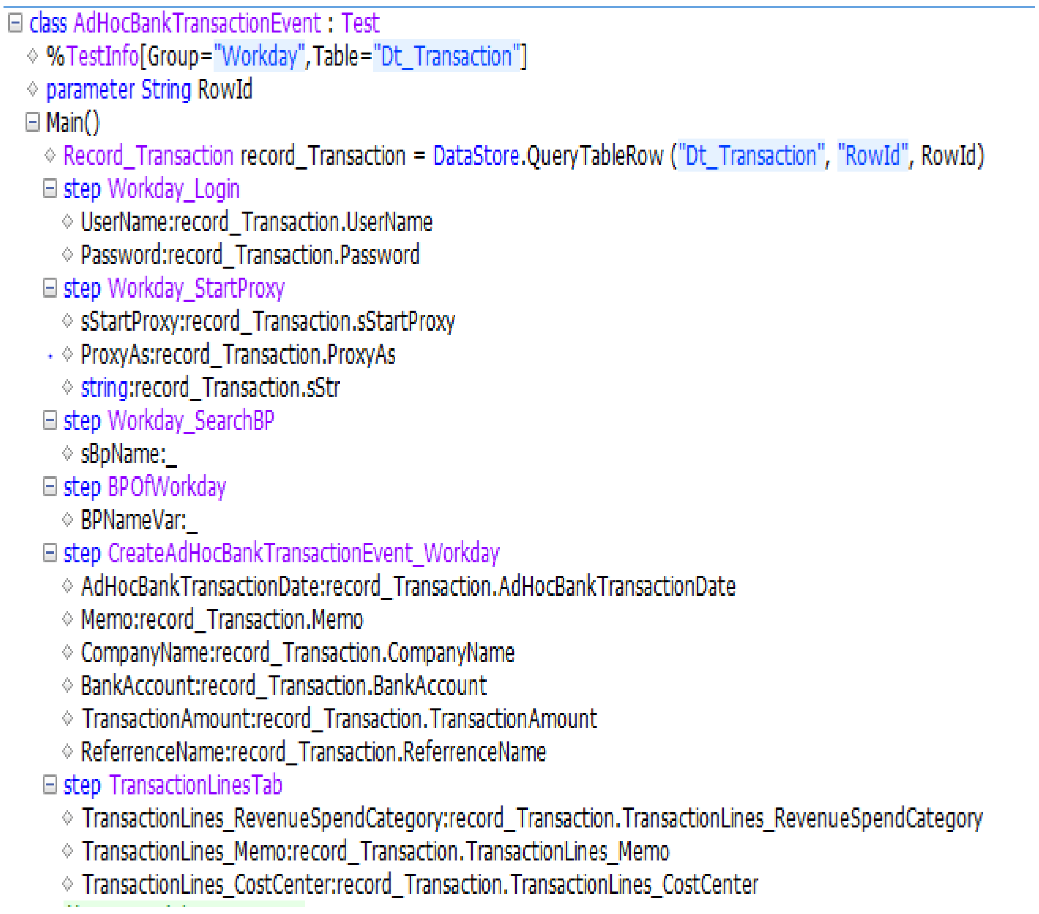
CreateAdHocBankTransactionEvent_Workday.cs | Workday_AdHocBankTransactionEvent.rxtst

ADD | RUN | TestRun | Maintenance mode

Data source...

Item	Data binding / iterations	Description
Workday_AdHocBankTransactionEvent - Test suite		
AdHocBankTransactionEvent	AdHocBankTransactionEvent Row: 1	
[SETUP]		
OpenBrowser	Bound variable: 1	
Workday_Login	Bound variables: 2	
Workday_StartProxy	Bound variables: 2	
Workday_SearchBP	Bound variable: 1	
BPOfWorkday	Bound variable: 1	
CreateAdHocBankTransactionEvent_Workday	Bound variables: 6	
TransactionLinesTab	AdHocBankTransaction_TransactionLinesTab	
TransactionLinesTab	Bound variables: 3	
Workday_StopProxy	*Bound variable: 1*	
[TEARDOWN]		
CloseBrowser		

AscentialTest Test:



```
class AdHocBankTransactionEvent : Test
  %TestInfo[Group="Workday",Table="Dt_Transaction"]
  parameter String RowId
  Main()
    Record_Transaction record_Transaction = DataStore.QueryTableRow ("Dt_Transaction", "RowId", RowId)
    step Workday_Login
      UserName:record_Transaction.UserName
      Password:record_Transaction.Password
    step Workday_StartProxy
      sStartProxy:record_Transaction.sStartProxy
      ProxyAs:record_Transaction.ProxyAs
      string:record_Transaction.sStr
    step Workday_SearchBP
      sBpName:_
    step BPOfWorkday
      BPNameVar:_
    step CreateAdHocBankTransactionEvent_Workday
      AdHocBankTransactionDate:record_Transaction.AdHocBankTransactionDate
      Memo:record_Transaction.Memo
      CompanyName:record_Transaction.CompanyName
      BankAccount:record_Transaction.BankAccount
      TransactionAmount:record_Transaction.TransactionAmount
      ReferrenceName:record_Transaction.ReferrenceName
    step TransactionLinesTab
      TransactionLines_RevenueSpendCategory:record_Transaction.TransactionLines_RevenueSpendCategory
      TransactionLines_Memo:record_Transaction.TransactionLines_Memo
      TransactionLines_CostCenter:record_Transaction.TransactionLines_CostCenter
```

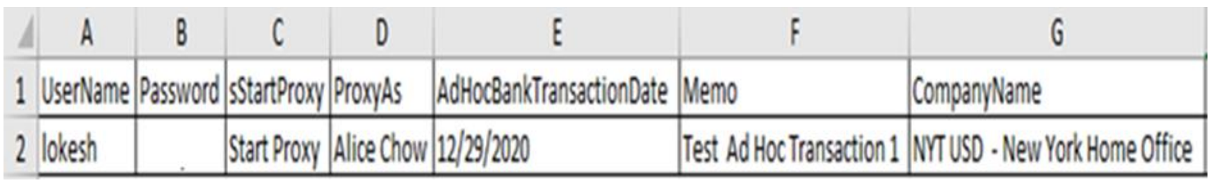
Challenges:

- Indentation of Converting Ranorex Tests to AT Tests is more challenging.
- Handling parameters.
- Calling Functions to Step in a sequential Manner along with Parameters
- Converting and passing the input values from the data table to the test was a challenge.

DataTable to DataTable:

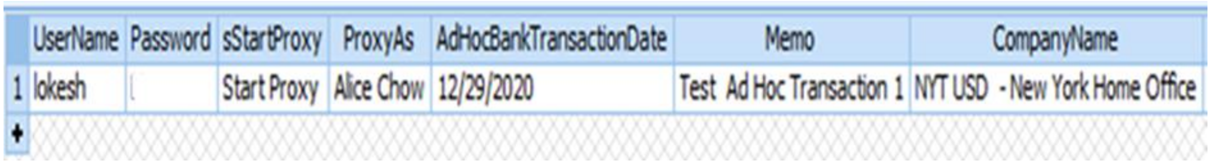
In Ranorex, inbuilt DataTable is a data structure used to organize and store test data for data-driven testing and also has external datatable like Excel. These are converted to AT inbuilt datable using the product Data Editor of Zeenyx. This creates a datatable of the same format with the column names and the values. Here in AT a unique column is automatically created as "RowID" which enables the test to take different set of parameters (Data Testing) input as shown in the example below:

Ranorex DataTable:



	A	B	C	D	E	F	G
1	UserName	Password	sStartProxy	ProxyAs	AdHocBankTransactionDate	Memo	CompanyName
2	lokesh		Start Proxy	Alice Chow	12/29/2020	Test Ad Hoc Transaction 1	NYT USD - New York Home Office

AscentialTest DataTable:



	UserName	Password	sStartProxy	ProxyAs	AdHocBankTransactionDate	Memo	CompanyName
1	lokesh		Start Proxy	Alice Chow	12/29/2020	Test Ad Hoc Transaction 1	NYT USD - New York Home Office
+							

Conclusion:

Over years of effort built over Ranorex project which is currently unstable/unreliable to maintain larger tests can be migrated quickly and efficiently to AscentialTest using the utilities created by Matryxsoft in a very short span of time which improves the performance, reliability and quality of the tests in turn improves the software quality, the utmost priority for one and all.

For more information on migrating Ranorex to AT test automation tool.
Visit Matryxsoft Tech at www.matryxsoft.com.