A System z Developer's Journey through the Application Lifecycle.

Set up

This document how to do some stuff to set up for the day in the life demo. Starting from a clean Eclipse workspace.

1. Make sure CICS region CICSTS41 is up and running

Let's start with the Business Analyst

1. Login to CICS region to show error. Using a MVS269 PCOMM type LOGON APPLID(DBDCCICS)
2. Clear screen (Pause key or right click and select Pause on mini pad)
3. Type EPSP as the transaction. Mortgage Calculator should start up.
4. Show the error. In this case the Length of Loan is only 1 digit and it needs to be 2. If you enter some values the transaction will actually abend.
5. Log into RTC repository with **guy2** userid. Password is **fun2test**. Connect to the Scrum Demo Project
6. Show the current Sprint 1 plan just to get a feel for the planning side. Show that as usual Rosalind is behind on her work. Maybe drag and drop some of the existing Developer A items to a different developer? Just to show how flexible the planning is.
7. Create a defect and assign it to Developer A

Now take over as the developer

8. Log into RTC repository with **deva** userid. Password is **fun2test**. Connect to the Scrum Demo Project
9. Log into RSE on MVS269 with **DOHERTL**.
10. Go to My Work and see what is on my plate. See the new defect, open it, changed planned for, and start working.
11. Open Source Control and create a new RWS off **MVS Dev Stream** that contains the **Mortgage Application.**
12. When the Load option starts choose the normal find and load projects. We are going to load the files in our Eclipse workspace.
13. In RTC Enterprise Extensions, System Definitions make the COBOL Compilation (CICS&DB2) the translator to be used for Syntax Check.

14. If not already open, open the Enterprise Projects View

We are going to digress for a while to have a play around in RDz

15. Open the EPSCMORT program and show that there are a bunch of yellow markers. Scroll down and show that it is unable to find the COPYBOOK. This is because wwe don't have a property group. Explain that a property group defines the SYSLIB so that copybooks can be resolved as well as providing input for compilation JCL.

16. Close EPSCMORT.

17. On the **MortgageApplication-EPSCMORT** project right click, select **Enterprise Extensions → Generate a property group…**

18. Open EPSCMORT again and show that the yellow markers are gone.

19. Show the different editors, LPEX and COBOL Editor. In LPEX show that the command line can be moved to the top and the appearance can be set to ISPF. However there are many more features in the new COBOL editor (such as TODO and Snippets) so we should use that.

20. Do a Find on COPY to find a copybook. Highlight the copybook name, then right click and open copybook member.

21. Show the outline view. Drill into the tree to show the divisions etc. Show that by clicking on a procedure you are taken there in the program.

22. Open EPSCMORT and add some TODOs in the code. Close the code. Mention that on a Friday I normally send myself an email to remind me what I was working on. Now go open the tasks view and we will see the TODOs listed. Double click and it will open the module and position on the code.

23. Using the new COBOL editor. Highlight some code. Right click and **Add to Snippets…** Also mention that you can define variables in place of real names. See A100.

24. Now go back to the code and pick somewhere to add new code. You will have the snippets view. By positioning in the code and clicking on a snippet it will be inserted into the code. By clicking on one that has a variable it will pop a dialog to enter the variable value.

25. Make some changes in the program to show real time syntax checking and content assist. Find a COPY then start to type COPY but use CTRL-Space to bring up the content assist box. Enter an invalid copybook, one that does not exists. Because we are doing real time syntax check it will look for the copybook in the SYSLIB as defined by the processor group.

26. Right click in the program and select **Show in → Program Control Flow.** Show that by clicking on the procedure it takes you to that code.

27. Other cool stuff. Select a block of code and right click and select **Source**.

28. Show User Build. Useful to run a compile on your code in isolation. You will need to go to the JES view to see the output.

29. Software analysis. On the Eclipse project name right click **Software Analysis → Software Analysis Configurations**. Right click on **Software Analysis** and select **New**…
30. Show the Rules sets that are provided, and the fact that you can write and import your own rules sets.
31. Look at the rules and how the ones related to the rule set are highlighted.
32. Show the icons that show if it will become a warning or an error
33. Click set on the rule set and it will set those rules. You can check others you wish as well.
34. Click **Analyze** to run code analysis. Look at the COBOL Code Review box. Drill into the Program Structures → Avoid GO TO. Double click on one, and show that it jumps to the code.
35. Click the right most Icon to generate a report. Generate a PDF with all the broken rules.
36. Show zUnit if possible

Ok back to the defect

37. Open the BMS for EPSMORT using **Open With → BMS Map Editor**
38. We can see the error in the screen that is painted there. Now go to the source and make the change. Show in the preview that the field has changed.
39. At this point when we make the change, and insert the extra 9 it pushes the line out by 1 putting it in error. We can see it flagged the error. But if we miss that and check-in/Deliver the deliver will fail because of a compilation pre-condition. Go back and fix.
40. Save the file and check it in. Do not deliver.
41. We are going to run a personal build to make sure it compiles OK. Show the build output and the fact that a BMS change has generated a COPYBOOK and forced a compile of the program using that COPYBOOK.
42. Talk to the fact that the developer might like to test this in their own CICS region. But that's not how we roll…
43. Let's deliver the change and associate it with the defect workitem. Show the Pending Changes view. Related Artifacts → Associate workitem, then deliver.
44. Developer marks the defect as done.

This next bit might be done by the Build and Release Admin

45. Now let's kick off an RTC team Build. Show the build has gone to a different target data set
46. Now we need to test the change. So in this scenario we are using RTCs package and deploy. Run a package, show the output.
47. Run a deploy and again show the output.

48. At this point mention UrbanCode Deploy and how we are integrating it with RTC.
49. Jump back to the CICS region and do a clear screen, then CEMT I PROG(EPS*) and newcopy the 2 affected programs.

Back to the business analyst

50. Run transaction EPSP again and this time the years field is 2 and we can enter a valid calculation without abending.