

Managing JCL changes through the Development Life Cycle – a different approach

Christian Le Fils and Chris Reed
ASG

November 2019
Session **ML**

Place your
custom session
QR code here.
Please remove
the border and
text beforehand.

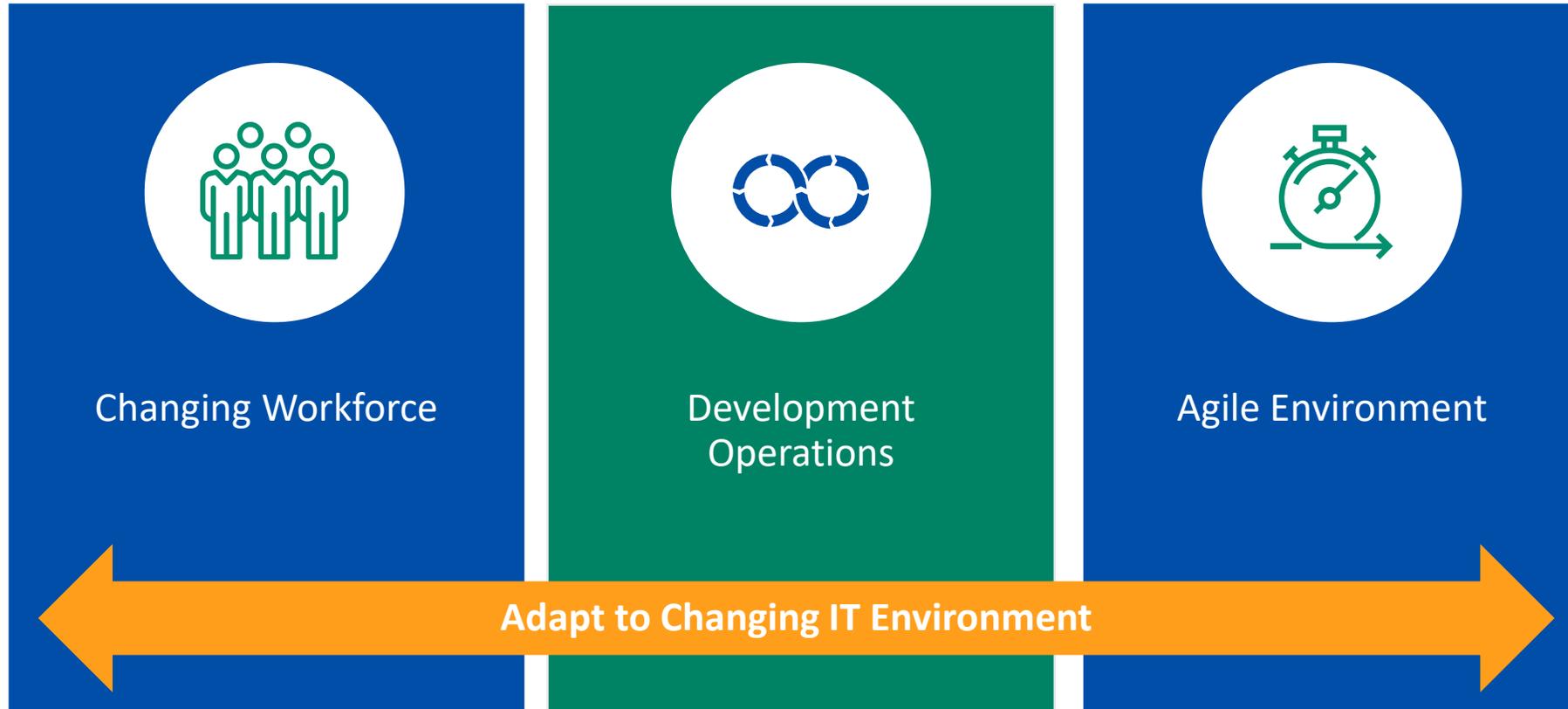


Managing JCL changes through the Development Life Cycle – a different approach

- Setting the Scene – the challenges of Managing JCL
- An Integrated and Automated Approach
- A Radical Approach – No JCL!
- Conclusions

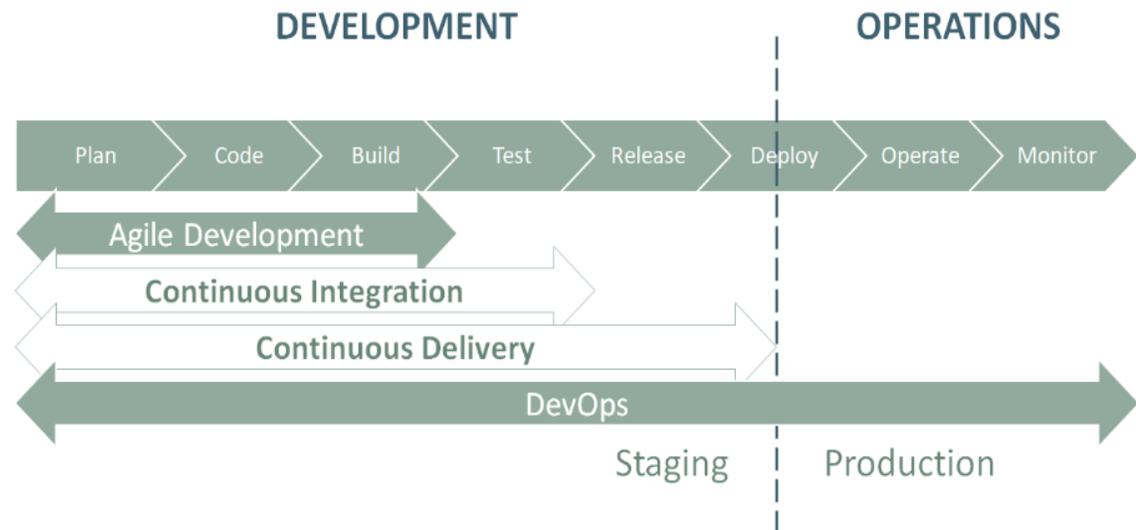
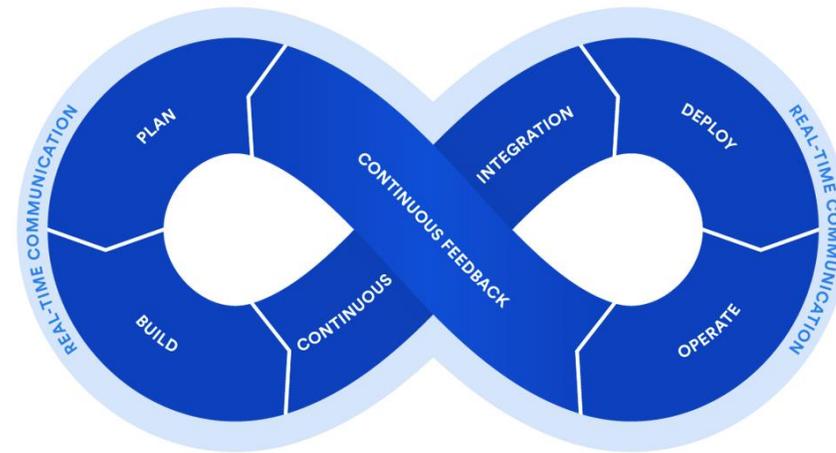
Setting the Scene – The challenges of Managing JCL

Initiatives in a Mainframe World



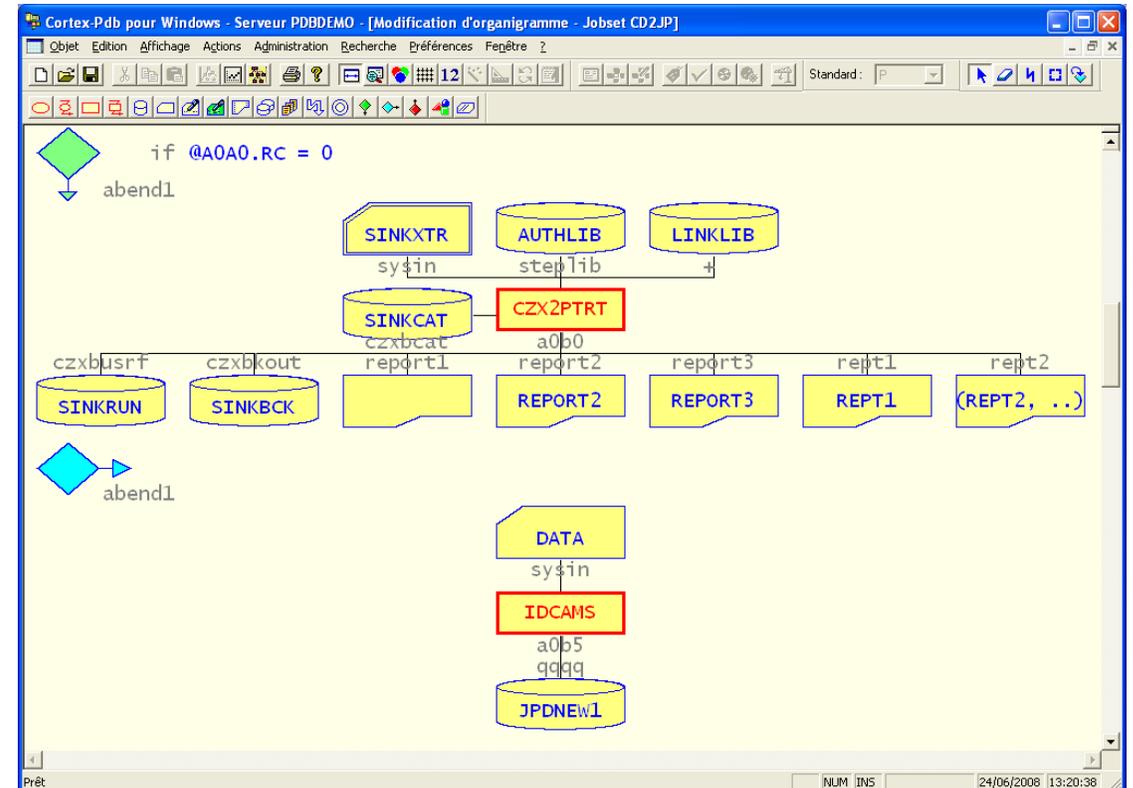
DEVOPS

- DevOps
 - The set of practices that emphasize the collaboration and communication of developers, testers, IT, automating the process of delivery and infrastructure changes
- DevOps Toolchain
 - The products or “tools” that work together to enable the DevOps philosophy
- CI/CD Solutions
 - Continuous Integration/Continuous Delivery solutions are part of the DevOps Toolchain



What is a JCL stream

- JCL is the “glue” that connects programs and files on z/OS.
- The language to run a sequence of application programs:
 - reading a bunch of sequential files
 - updating some databases
 - extracting data from those databases
 - building some new sequential files and new batch reports
- More or less something like :



What is a JCL stream for Operations staff ?

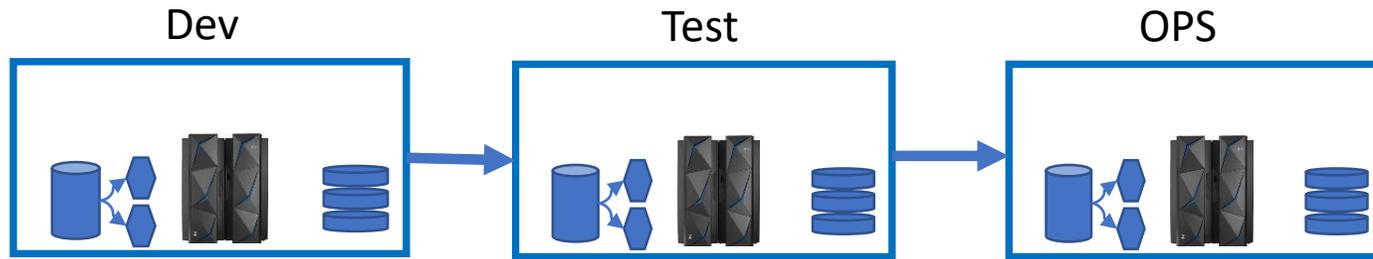
- **The language to run a sequence of application programs, but also :**
 - Some technical steps to save the database before update, notify the scheduler, facilitate the restart in case of failure, ...
 - Implementing the datasets with the constraints of the production
 - Adding some scheduling specific statements
 - Implementing GDG ...
- **More or less something like :**

```

//UPDTA5B0 EXEC PGM=PGKSUP
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=X
//SORTF DD DSN=PUPDT.XP.UPDTA5A5.STRECRDS(&Gà1),DISP=SHR
//KSDSUPS DD DSN=PUPDT.XS.UPDTA5B0.UPDTSMMR(+1),
//          DISP=(NEW,CATLG,DELETE),DSORG=PS,
//          RECFM=FB,LRECL=50,VOL=(,RETAIN),LABEL=(RETPD=0010)
//KSDSF DD DSN=P.DM.KSDSFILE,DISP=SHR
//SNAPDD DD SYSOUT=A
//*****
//**          BACKUP OF KSDS          **
//*****
//UPDTA5B5 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//KSDSF DD DSN=P.DM.KSDSFILE,DISP=SHR
//KSDSBAC DD DSN=PUPDT.BU.UPDTA5B5.KSDSBCKP(+1),
//          DISP=(NEW,CATLG,DELETE),DSORG=PS,
//          RECFM=FB,LRECL=80,LABEL=(RETPD=0021)
//SYSIN DD DSN=CORTEX.CLF.P.MTXTPRM1(UPDTA5B5),DISP=SHR

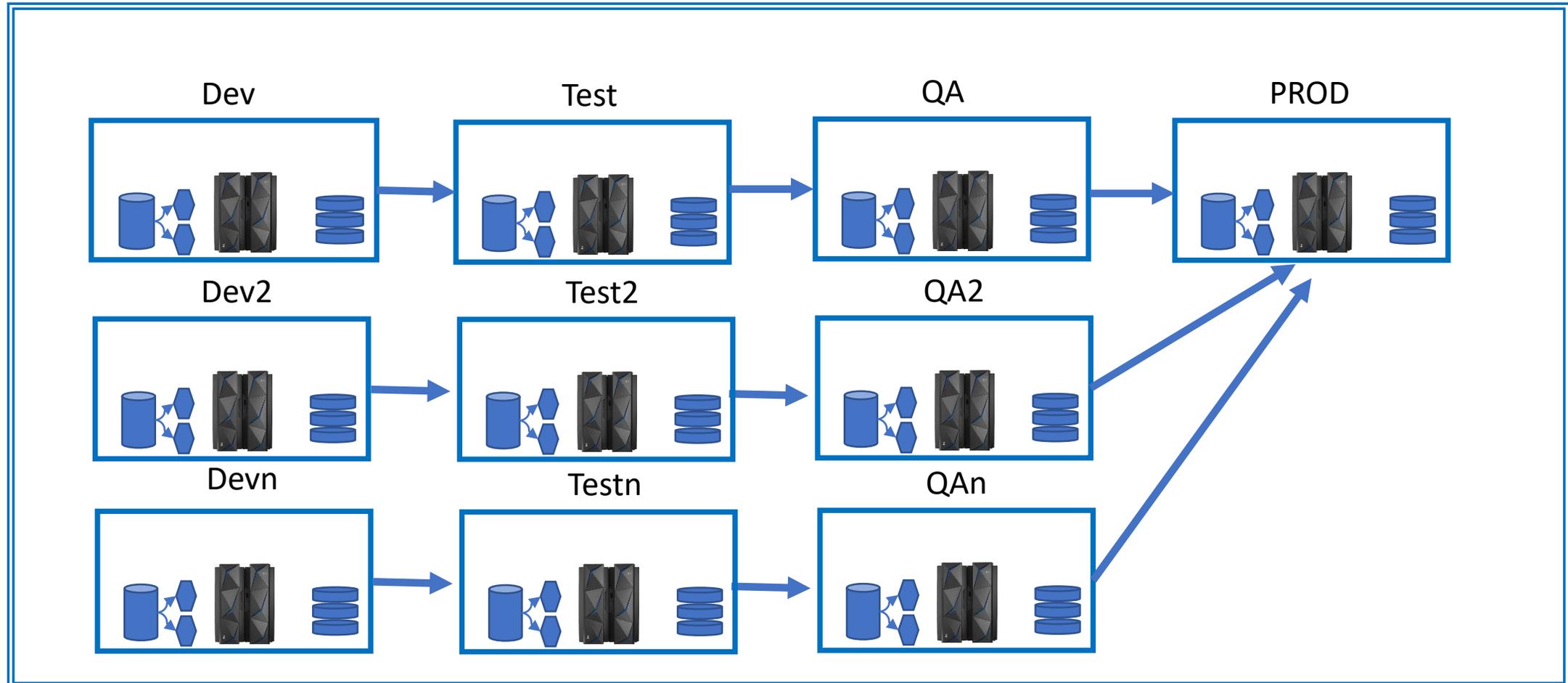
```

Is JCL the the same from Dev to Ops ?



- The applicative part (what we call the functional part) is the same in both Dev and Ops, but the Technical stuff is not the same because the Production constraints are not the same.
- There are usually more strict rules in Production :
 - Dataset naming conventions, job and step naming conventions ...
 - Media (Disk, VTS, ...) choice are different because volume of data is usually less in Dev
 - Usage of GDG, with a appropriate number of versions (to avoid wasting disk space)
 - Structure of Jobs (start with a header step, finish with an ending step)
 - Scheduling constraints
 - And more....

Typical Mainframe Life Cycle



What does that mean for your JCL?

CHANGE

CHANGE

CHANGE

CHANGE

```
//BKUPLLD EXEC PGM=IDCAMS,COND=(0,NE)
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//IN DD DSN=IDS.VS.A0.LLD.IDSACCT,DISP=SHR
//OUT DD DSN=IDS.QS.A0.LLD.IDSACCT.HARILALU,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=43,BLKSIZE=0),
// UNIT=3390,SPACE=(CYL,(10,10),RLSE)
//SYSIN DD *
        REPRO INFILE(IN) -
            OUTFILE(OUT) -
            REPLACE

/*
//*****
//* @D DELETE DEFINE IDS ACCOUNT DATASET (COPY FROM INTL) *
//* @R RUN FROM SORTLLD *
//*****
//DELVSLLD EXEC PGM=IDCAMS,REGION=2048K,COND=(0,NE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DELETE IDS.VS.A0.LLD.IDSACCT CL PRG
DEFINE CLUSTER -
        (NAME (IDS.VS.A0.LLD.IDSACCT) -
        INDEXED -
        SHAREOPTIONS(2,3) -
        NOERASE -
        RECOVERY -
        NOWRITECHECK -
        NONSPANNED) -
```

What does that mean for your JCL?

CHANGE

```
//DELDEFNM EXEC PGM=IDCAMS,REGION=2048K,COND=(0,NE)
//*
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SYSIN DD *
```

```
DELETE (IDS.VS.A0.LLD.IDSACCT.ALTNDX) AIX
```

CHANGE

```
DEFINE AIX -
(NAME (IDS.VS.A0.LLD.IDSACCT.ALTNDX) -
```

```
RELATE (IDS.VS.A0.LLD.IDSACCT) -
```

```
NONUNIQUEKEY -
```

```
UPGRADE -
```

```
SPEED -
```

```
KEYS (15 10) -
```

```
RECORDS (3000 1000) -
```

```
SHR (2 3))
```

CHANGE

```
DEFINE PATH -
```

```
(NAME (IDS.VS.A0.LLD.IDSACCT.IDSNAME) -
```

```
PATHENTRY (IDS.VS.A0.LLD.IDSACCT.ALTNDX) -
```

```
UPDATE)
```

CHANGE

```
BLDINDEX INDATASET (IDS.VS.A0.LLD.IDSACCT) -
```

```
OUTDATASET (IDS.VS.A0.LLD.IDSACCT.ALTNDX)
```

```
IF MAXCC = 08 THEN SET MAXCC = 0
```

What does that mean for your JCL?

CHANGE

CHANGE

CHANGE

CHANGE

CHANGE

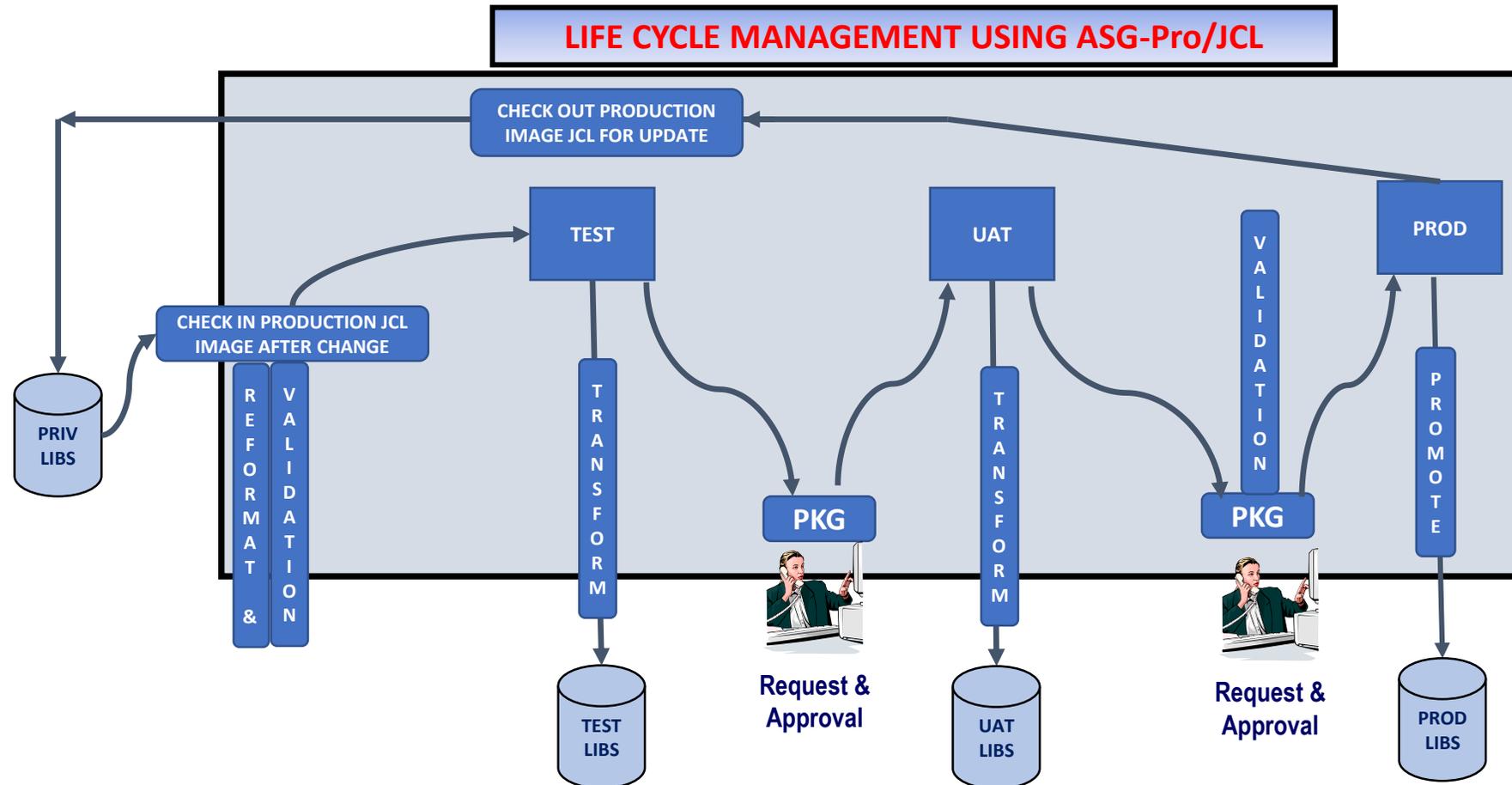
```

//CIBK0010 EXEC PGM=IKJEFT01.DYNAMNBR=20,COND=(0,NE)
//STEPLIB DD DSN=DB2PRD.SDSNLOAD,DISP=SHR
//          DD DSN=MNI.BCA.P.DB2LOAD,DISP=SHR
//          DD DSN=MNI.BCA.P.LOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSOUT   DD SYSOUT=A
//SYSOUD   DD SYSOUT=A
//BULKIN   DD DSN=LKS.GS.A0.CIBK0010.BULKIN,DISP=SHR
//BANKDTE  DD DSN=IDS.QS.A0.CIBK0010.CIBK0010.BANKDTE,DISP=SHR
//BULKCTLO DD DSN=IDS.QS.A0.CIBK0010.BULKCNTL,
//          DISP=(,CATLG,DELETE),
//          UNIT=SYSDA,
//          SPACE=(CYL,(20,10),RLSE),
//          DCB=(RECFM=FB,LRECL=1000,BLKSIZE=0)
//CI001AGS DD DSN=IDS.GS.A0.CIBK0010.CI001AGS,
//          DISP=(,CATLG,DELETE),
//          UNIT=SYSDA,
//          SPACE=(CYL,(20,10),RLSE),
//          DCB=(RECFM=FB,LRECL=1542,BLKSIZE=0)
//SYSTSIN  DD *
DSN SYSTEM(DSN) RRTY(0) TEST(0)
RUN PROGRAM(CIBK0010) PLAN(DPBTCHPL) -
LIB('MNI.BCA.P.LOAD')
/*
//*-----*
//* EOF THE CIS BATCH CUSTOMER OPEN REPORT GSAM FILES *

```

AN INTEGRATED AND AUTOMATED APPROACH

Integration of JCL Management into Life Cycle Management Software





JCL Reformatting

```
000001 //REFJCL1 JOB (A,P,GL), 'MESS', CLASS=P, MSGCLASS=H
000002 //*
000003 //PS010 EXEC PGM=IEBGENER
000004 //SYSPRINT DD SYSOUT=A
000005 //SYSUT1 DD DSN=TAPE9, UNIT=TAPE9, DISP=OLD
000006 //SYSUT2 DD DSN=&&TEMP, DISP=(,PASS), UNIT=SYSDA,
000007 // DCB=BLKSIZE=6160, SPACE=(TRK, (5,1))
000008 //SYSUDUMP DD SYSOUT=A
```

BEFORE REFORMATTING

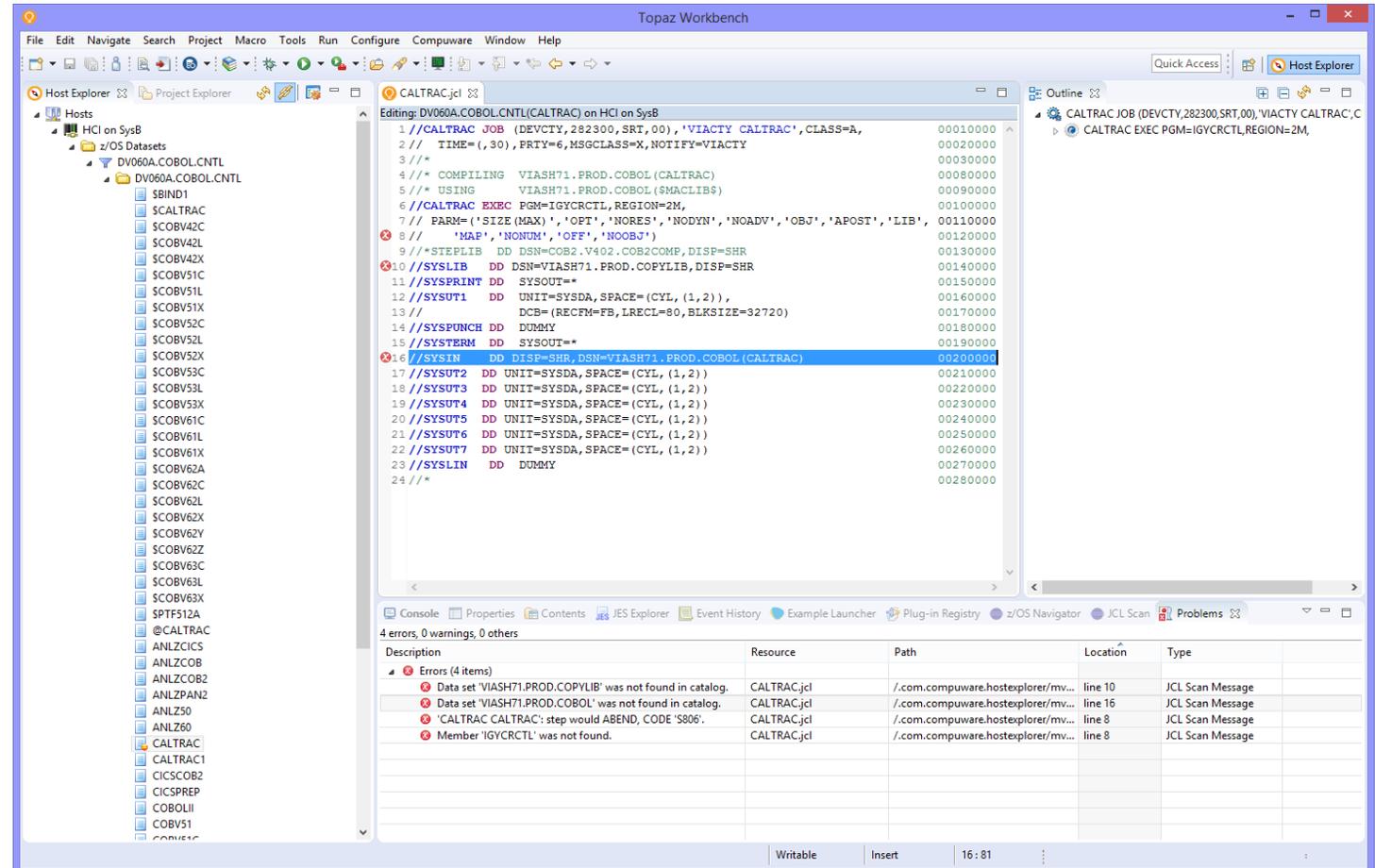
```
000001 //REFJCL1 JOB (A,P,GL), 'MESS', CLASS=P,
000002 //          MSGCLASS=H
000003 //*
000004 //PS010      EXEC PGM=IEBGENER
000005 //SYSPRINT DD SYSOUT=A
000006 //SYSUT1     DD DSN=TAPE9, DISP=OLD,
000007 //          UNIT=TAPE9
000008 //SYSUT2     DD DSN=&&TEMP, DISP=(,PASS),
000009 //          UNIT=SYSDA,
000010 //          DCB=BLKSIZE=6160,
000011 //          SPACE=(TRK, (5,1))
000012 //SYSUDUMP DD SYSOUT=A
```

AFTER REFORMATTING



JCL validation

- Validation – application of standards to ensure quality and consistency & checks for errors.
- New release of Pro/JCL comes with an Eclipse front end.
- Eclipse is the foundation of popular development tools like Compuware Topaz® and IBM iDZ. Offers integration to an IDE.



The screenshot shows the Topaz Workbench interface with a JCL file open. The JCL code is as follows:

```

1 //CALTRAC JOB (DEVCTY,282300,SRT,00),'VIACTY CALTRAC',CLASS=A, 00010000
2 // TIME=(,30),PRTY=6,MSGCLASS=X,NOTIFY=VIACTY 00020000
3 /** 00030000
4 /** COMPILING VIASH71.PROD.COBOL (CALTRAC) 00080000
5 /** USING VIASH71.PROD.COBOL ($MACLIB$) 00090000
6 //CALTRAC EXEC PGM=IGYCRCTL,REGION=2M, 00100000
7 // PARM=('SIZE (MAX)', 'OPT', 'NORES', 'NODYN', 'NOADV', 'OBJ', 'APOST', 'LIB', 00110000
8 // 'MAP', 'NONUM', 'OFF', 'NOOBJ') 00120000
9 /** STEPLIB DD DSN=COB2.V402.COB2COMP, DISP=SHR 00130000
10 //SYSLIB DD DSN=VIASH71.PROD.COPYLIB, DISP=SHR 00140000
11 //SYSPRINT DD SYSOUT=* 00150000
12 //SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)), 00160000
13 // DCB=(RECFM=FB, LRECL=80, BLKSIZE=32720) 00170000
14 //SYSPPUNCH DD DUMMY 00180000
15 //SYSTEM DD SYSOUT=* 00190000
16 //SYSIN DD DISP=SHR, DSN=VIASH71.PROD.COBOL (CALTRAC) 00200000
17 //SYSUT2 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)) 00210000
18 //SYSUT3 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)) 00220000
19 //SYSUT4 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)) 00230000
20 //SYSUT5 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)) 00240000
21 //SYSUT6 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)) 00250000
22 //SYSUT7 DD UNIT=SYSDA, SPACE=(CYL, (1, 2)) 00260000
23 //SYSLIN DD DUMMY 00270000
24 /** 00280000
  
```

The console at the bottom shows 4 errors:

Description	Resource	Path	Location	Type
Data set 'VIASH71.PROD.COPYLIB' was not found in catalog.	CALTRAC.jcl	/.com.compuware.hostexplorer/mv...	line 10	JCL Scan Message
Data set 'VIASH71.PROD.COBOL' was not found in catalog.	CALTRAC.jcl	/.com.compuware.hostexplorer/mv...	line 16	JCL Scan Message
'CALTRAC CALTRAC': step would ABEND, CODE 'S806'.	CALTRAC.jcl	/.com.compuware.hostexplorer/mv...	line 8	JCL Scan Message
Member 'IGYCRCTL' was not found.	CALTRAC.jcl	/.com.compuware.hostexplorer/mv...	line 8	JCL Scan Message

JCL Transformation

Why Transformation

Production Image cannot execute in other stages. Must be modified in many place

When

Transformation takes place at the moment the JCL is written from the Life Cycle Manager tool into the JCL execution library.

How

By a fixed set of transformation rules. Transforming Production Image JCL into DEV, System Test or QA style JCL using a set of rules means that in effect the Production JCL has been validated at each stage of the life cycle. What works in Test or QA will work in Production

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2  DSN                      DSN3                      BT12
DB2  DSN                      DSN2                      *
DB2  DSNR                    DSN3                      *
DBU  BTUSER                  BTPUSER
IMS  IMS                     IMX
IMS  IMR                     IMRX
IMS  IVP3                    IVP3X
HLQ  ATM                     ATMX
HLQ  IDS                     IDSX
HLQ  LKS                     LKSX
HLQ2 IDS.VS                  IDSX.VST
HLQ  ATMX                    ATMXX
LIB  MNI.BC.P.LOAD          LCM.ATM.UAT1.LOADLIB      MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2  DSN                DSN3                BT12
DB2  DSN                DSN2                *
DB2  DSNR              DSN3                *
DBU  BTUSER            BTPUSER
IMS  IMS               IMX
IMS  IMR               IMRX
IMS  IVP3              IVP3X
HLQ  ATM               ATMX
HLQ  IDS               IDSX
HLQ2 LKS.GS            LKSX.GST
HLQ  ATMX              ATMXX
LIB  MNI.BC.P.LOAD    LCM.ATM.UAT1.LOADLIB  MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2  DSN          DSN3          BT12
DB2  DSN          DSN2          *
DB2  DSNR        DSN3          *
DBU  BTUSER      BTPUSER
IMS  IMS         IMX
IMS  IMR         IMRX
IMS  IVP3        IVP3X
HLQ  ATM         ATMX
HLQ  IDS         IDSX
HLQ2 LKS.GS      LKSX.GST
HLQ  ATMX        ATMXX
LIB  MNI.BC.P.LOAD  LCM.ATM.UAT1.LOADLIB  MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2 DSN DSN3 BT12
DB2 DSN DSN2 *
DB2 DSNR DSN3 *
DEU BTUSER BTPUSER
IMS IMS IMX
IMS IMR IMRX
IMS IVP3 IVP3X
HLQ ATM ATMX
HLQ IDS IDSX
HLQ2 LKS.GS LKSX.GST
HLQ ATMX ATMXX
LIB MNI.BC.P.LOAD LCM.ATM.UAT1.LOADLIB MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2  DSN          DSN3          BT12
DB2  DSN          DSN2          *
DB2  DSNR        DSN3          *
DBU  BTUSER      BTPUSER
IMS  IMS         IMX
IMS  IMR         IMRX
IMS  IVP3        IVP3X
HLQ  ATM         ATMX
HLQ  IDS         IDSX
HLQ2 LKS.GS      LKSX.GST
HLQ  ATMX        ATMXX
LIB  MNI.BC.P.LOAD  LCM.ATM.UAT1.LOADLIB  MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2  DSN                DSN3                BT12
DB2  DSN                DSN2                *
DB2  DSNR              DSN3                *
DBU  BTUSER            BTPUSER
IMS  IMS               IMX
IMS  IMR               IMRX
IMS  IVP3              IVP3X
HLQ  ATM               ATMX
HLQ  IDS               IDSX
HLQ2 LKS.GS            LKSX.GST
HLQ  ATMX              ATMXX
LIB  MNI.BC.P.LOAD     LCM.ATM.UAT1.LOADLIB  MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Transformation Rules

```

***** Top of Data *****
* TABLE 01 - CONFIG=UAT*, ALL TYPES, ALL SYSTEMS
DB2  DSN                DSN3                BT12
DB2  DSN                DSN2                *
DB2  DSNR              DSN3                *
DBU  BTUSER            BTPUSER
IMS  IMS               IMX
IMS  IMR               IMRX
IMS  IVP3              IVP3X
HLQ  ATM               ATMX
HLQ  IDS               IDSX
HLQ2 LKS.GS            LKSX.GST
HLQ  ATMX              ATMXX
LIB  MNI.BC.P.LOAD     LCM.ATM.UAT1.LOADLIB  MNI.BC.PCOPY.LOADLIB
***** Bottom of Data *****

```

Before and after example of transformed JCL

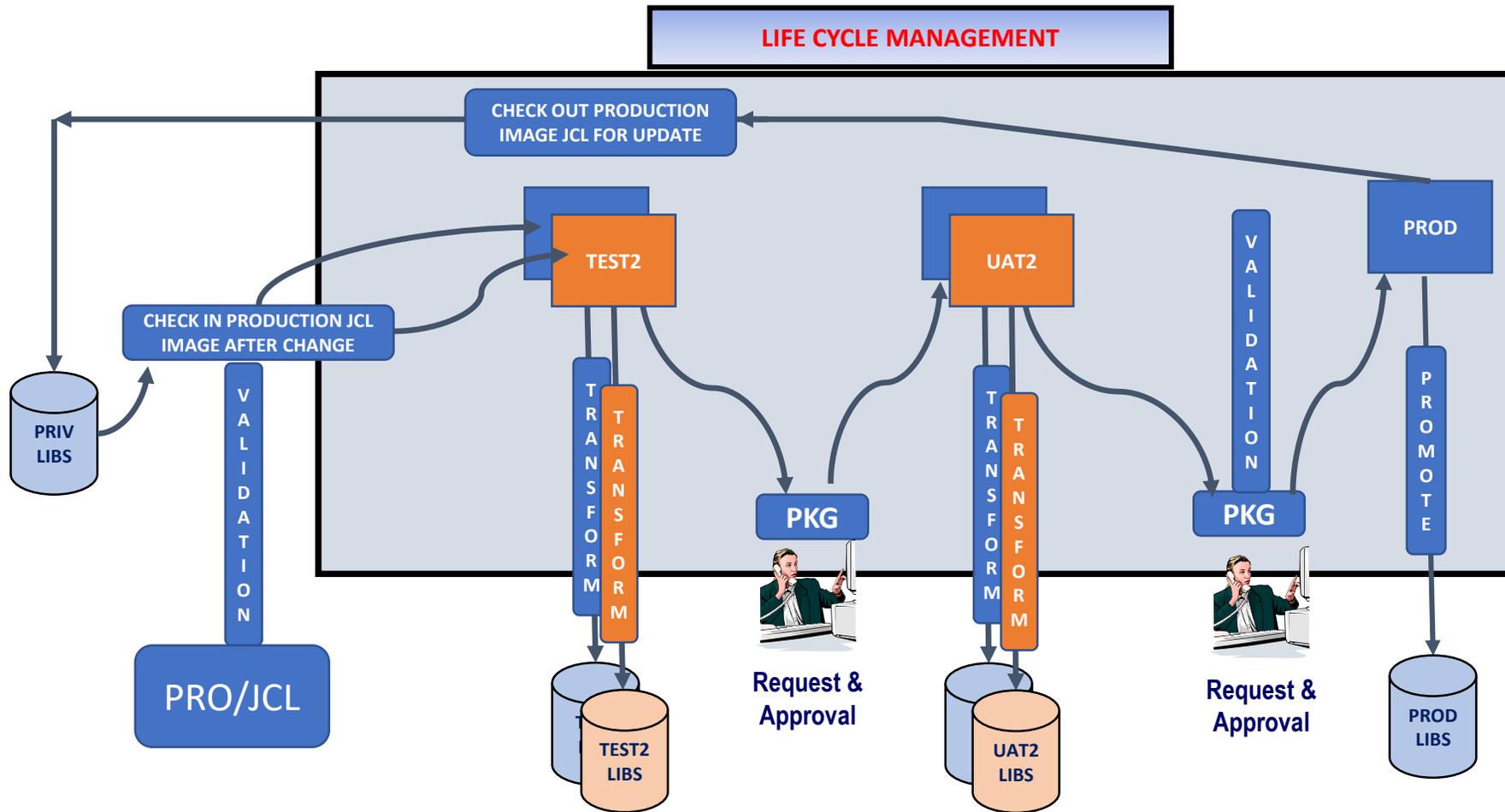
```
//CIBK0010 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(0,NE)
//STEPLIB DD DSN=DB2PRD.SDSNLOAD,DISP=SHR
// DD DSN=MNI.BCA.P.DB2LOAD,DISP=SHR
// DD DSN=MNI.BC.P.LOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//BULKIN DD DSN=LKS.GS.A0.CIBK0010.BULKIN,DISP=SHR
//BANKDTE DD DSN=IDS.QS.A0.GLDATE,DISP=SHR
//BULKCTLO DD DSN=IDS.QS.A0.CIBK0010.BULKCNTL,
// DISP=(,CATLG,DELETE),UNIT=SYSDA,
// SPACE=(CYL,(20,10),RLSE),
// DCB=(RECFM=FB,LRECL=1000,BLKSIZE=0)
//CI001AGS DD DSN=IDS.GS.A0.CIBK0010.CI001AGS,
// DISP=(,CATLG,DELETE),UNIT=SYSDA,
// SPACE=(CYL,(20,10),RLSE),
// DCB=(RECFM=FB,LRECL=1542,BLKSIZE=0)
//SYSTSIN DD *
DSN SYSTEM(DSN) RETRY(0) TEST(0)
RUN PROGRAM(CIBK0010) PLAN(DPBTCHPL) -
  LIB('MNI.BC.P.LOAD')
/*
//*-----*
//* EOF THE CIS BATCH CUSTOMER OPEN REPORT GSAM FILES
```



```
//CIBK0010 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(0,NE)
//STEPLIB DD DSN=DB2PRD.SDSNLOAD,DISP=SHR
// DD DSN=MNI.BCA.P.DB2LOAD,DISP=SHR
// DD DSN=LCM.ATM.UAT1.LOADLIB,DISP=SHR
// DD DSN=MNI.BC.PCOPY.LOADLIB,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//BULKIN DD DSN=LKSX.GST.A0.CIBK0010.BULKIN,DISP=SHR
//BANKDTE DD DSN=IDSX.QS.A0.GLDATE,DISP=SHR
//BULKCTLO DD DSN=IDSX.QS.A0.CIBK0010.BULKCNTL,
// DISP=(,CATLG,DELETE),UNIT=SYSDA,
// SPACE=(CYL,(20,10),RLSE),
// DCB=(RECFM=FB,LRECL=1000,BLKSIZE=0)
//CI001AGS DD DSN=IDSX.GS.A0.CIBK0010.CI001AGS,
// DISP=(,CATLG,DELETE),UNIT=SYSDA,
// SPACE=(CYL,(20,10),RLSE),
// DCB=(RECFM=FB,LRECL=1542,BLKSIZE=0)
//SYSTSIN DD *
DSN SYSTEM(DSN2) RETRY(0) TEST(0)
RUN PROGRAM(CIBK0010) PLAN(DPBTCHPL) -
  LIB('LCM.ATM.UAT1.LOADLIB','MNI.BC.PCOPY.LOADLIB')
/*
//*-----*
//* EOF THE CIS BATCH CUSTOMER OPEN REPORT GSAM FILES
```



Integration of JCL Management Supporting Agile development



Bring JCL Changes in from the cold!

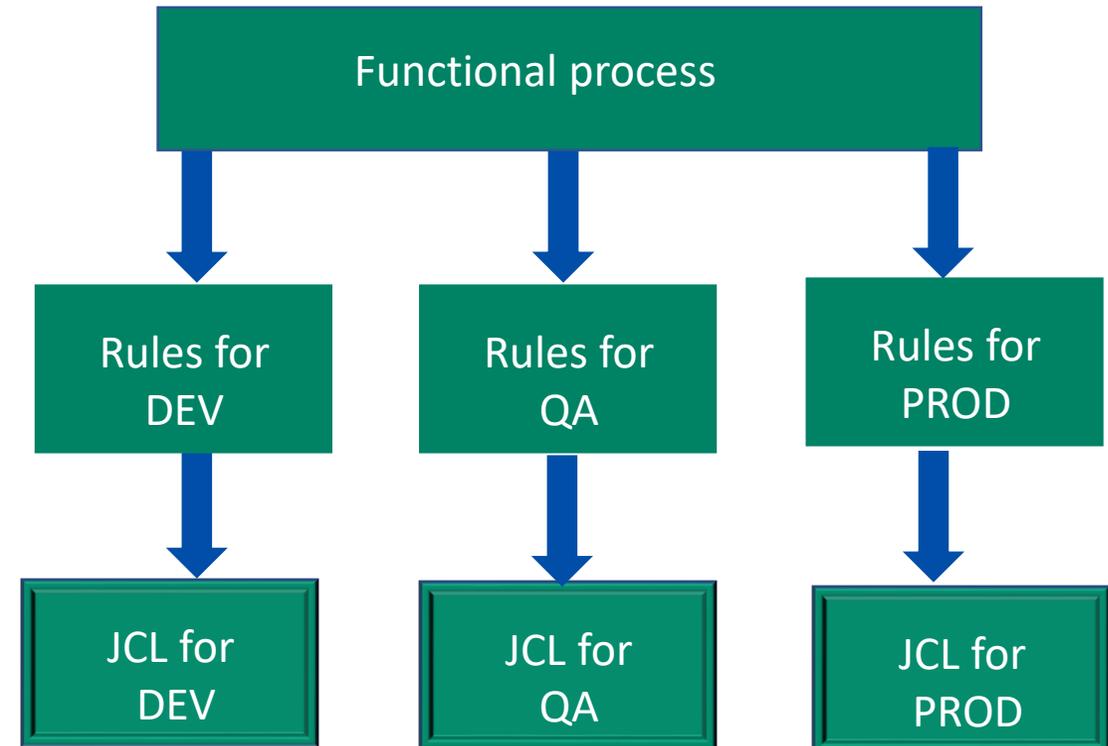
- Easier to use. Accessible from Eclipse based development workbenches in common with many other development tools.
- Automation of changes for each stage of the life cycle
 - You are sure the constraints of each stage (Dev, QA, Preprod, Prod) are **strictly** followed
 - Everybody **works the same way**, avoiding loss of time when an other person has to check or maintain a JCL
 - You gain **productivity**, because adding technical steps or transforming a dataset name is not productive and prone of errors
- Easier to work with parallel development environments, frees the developer from needing to know all the technical constraints of each development environment.
- Supports development on IBM Z Development and Test Environment



A Radical Approach – No JCL!

A Radical Approach – No JCL!

- The [ASG-Cortex PDB](#) approach is that JCL is the transposition of a functional process into a technical instance in a designated technical environment
- The idea is that the path from functional description of the process to the technical JCL stream can be automated by applying some customized rules
- So, by applying different rule sets, you can also get different JCL streams: one for DEV, one for QA, one for PROD





A Functional Language (PCL) - to describe the process

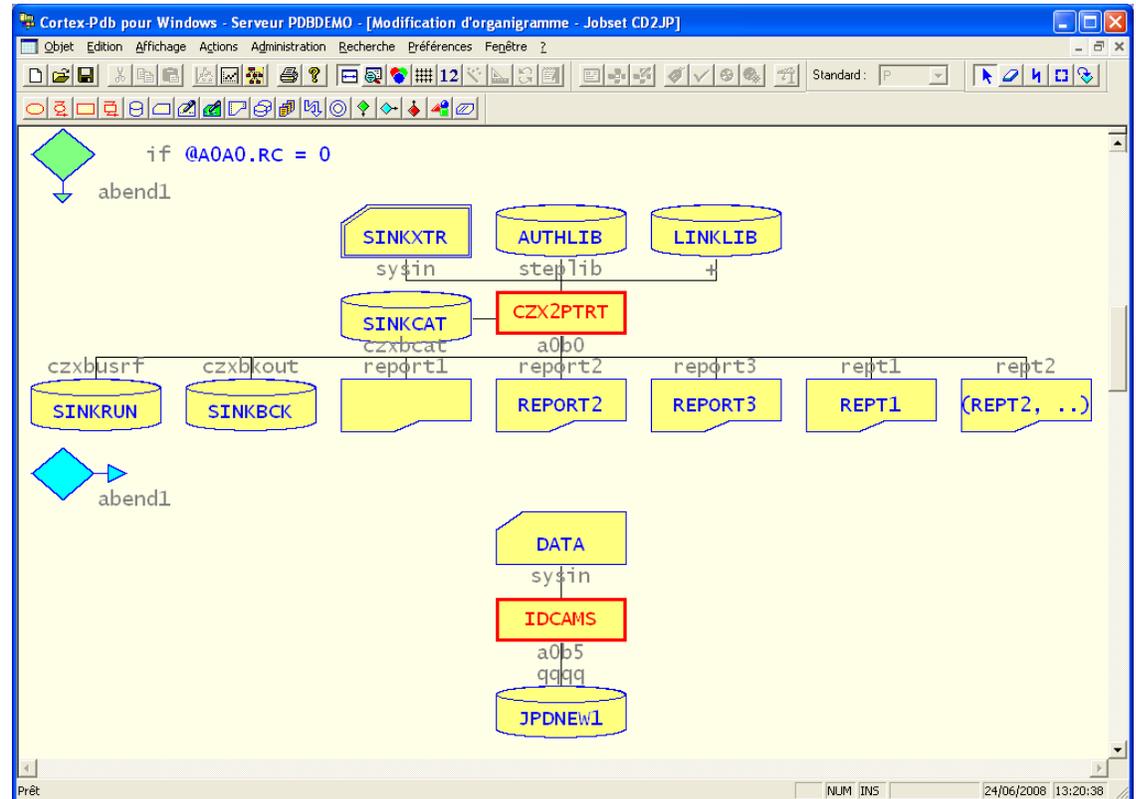
- To allow functional description, **ASG-Cortex PDB** embeds a functional and concise language called PCL ('Production Control Language')
- In the example on right, you can see that the access mode to a file is more precise than the DISP parameter in the JCL
- From that language easy to learn, **ASG-Cortex PDB** is able to generate the different 'flavours' of JCL

```
STEP PGM=LKRB0007,LANG=CBL
*
LKL6600 FILE MODE=I,NAME=LKL6600
LKR0801 FILE MODE=I,NAME=LKL0801
LKM8301 FILE MODE=O,NAME=LKL83AA
LKM8302 FILE MODE=O,NAME=LKL83AB
LOM7131 FILE MODE=O,NAME=LKL7131
LKC1001 FILE MODE=U,NAME=LKL1046
LOP0301 FILE MODE=O,NAME=LKL03PW
LKR0008L REPORT NAME=LBI0015R
COMREG DATA *
//*** MESSAGE      STE-COD 0128 --> REMA.
        DATAEND
LOP0101 DATA *
STE-COD 0128
        DATAEND
```



Not Excited to Learn a New Language?

- New people in the team have not a lot of knowledge in mainframe and JCL
- Hope they will like this kind of tool to describe the functional process.
- From a Windows station connected to the mainframe server, using Drag and Drop facilities, they can draw the process as they would do on a paper.
- When they save the flowchart, PCL stream is created in the mainframe.





Get a JCL Stream Through a Single Click !

- From the same tool; a user can also:

- Push a button to start the JCL generation (on mainframe) and get the result in the Windows tool
- Launch requests like “where is my file used ?”
- Print the flowchart

PROD

The screenshot shows a mainframe JCL editor window with the following JCL code:

```

*****
*** STEP DEMO0A0 PGM=IKJEFT1B ** ID=AOA0
*****
//DEMO0A0 EXEC PGM=IKJEFT1B,PARM=('030'),COND=((1,EQ,4EXAOA0),(50,LT,
//          FROM), (50,GT,TO)),DYNAMNBR=99
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=X
//DD1 DD DSN=CORTEX.DEMO.F.CORT01(0),DISP=SHR
//DD3 DD DSN=CORTEX.DEMO.F.CORT02(0),DISP=SHR
//SYSIN DD DSN=CORTEX.DEMO.P.MTXTPFM1(DEMO0A0),DISP=SHR
/*-M-CORTD2 - IS UPDATED IN PLACE WITH MODE=U
//DD2 DD DSN=CORTEX.DEMO.F.CORTD2,DISP=SHR
/*-M-CORTD1 - IS UPDATED IN PLACE WITH MODE=U
//DDU DD DSN=CORTEX.DEMO.F.CORTD1,DISP=SHR
//DD7 DD DSN=CORTEX.DEMO.F.CORT04(+1),DISP=(NEW,CATLG,DELETE),
//          RECFM=FB,LRECL=120,SPACE=(4096,(59,12),RLSE)
//FIC01 DD DSN=CORTEX.DEMO.F.CORT01(+1),DISP=(NEW,CATLG,DELETE),
//          RECFM=FB,LRECL=120,SPACE=(4096,(3,1),RLSE)
//PRINT1 DD SYSOUT=A
//          IF $RA0A0.NE 0 THEN
/*
*****
*** STEP DEMO0A5 PGM=CZSP2WT ** ID=AOA5
*****
//DEMO0A5 EXEC PGM=CZSP2WT,PARM=('020'),COND=((1,EQ,4EXAOA5),(51,LT,
//          FROM), (51,GT,TO))
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=X
//FIC03 DD DSN=CORTEX.DEMO.F.CORT03(+1),DISP=(NEW,CATLG,DELETE),
//          RECFM=FB,LRECL=120,SPACE=(4096,(3,1),RLSE)
//          RESOURCE NAME=DBASEXX
//FIC01 DD DSN=CORTEX.DEMO.F.CORT01(4081),DISP=
//          RECFM=FB,LRECL=120,SPACE=(4096,(3,1),RLSE)
/*-M-CORTD1 - IS UPDATED IN PLACE WITH MODE=U
//FICD1 DD DSN=CORTEX.DEMO.F.CORTD1,DISP=SHR
/*
*****
*** STEP DEMO0B0 PGM=CZSP2WT ** ID=AOB0
*****

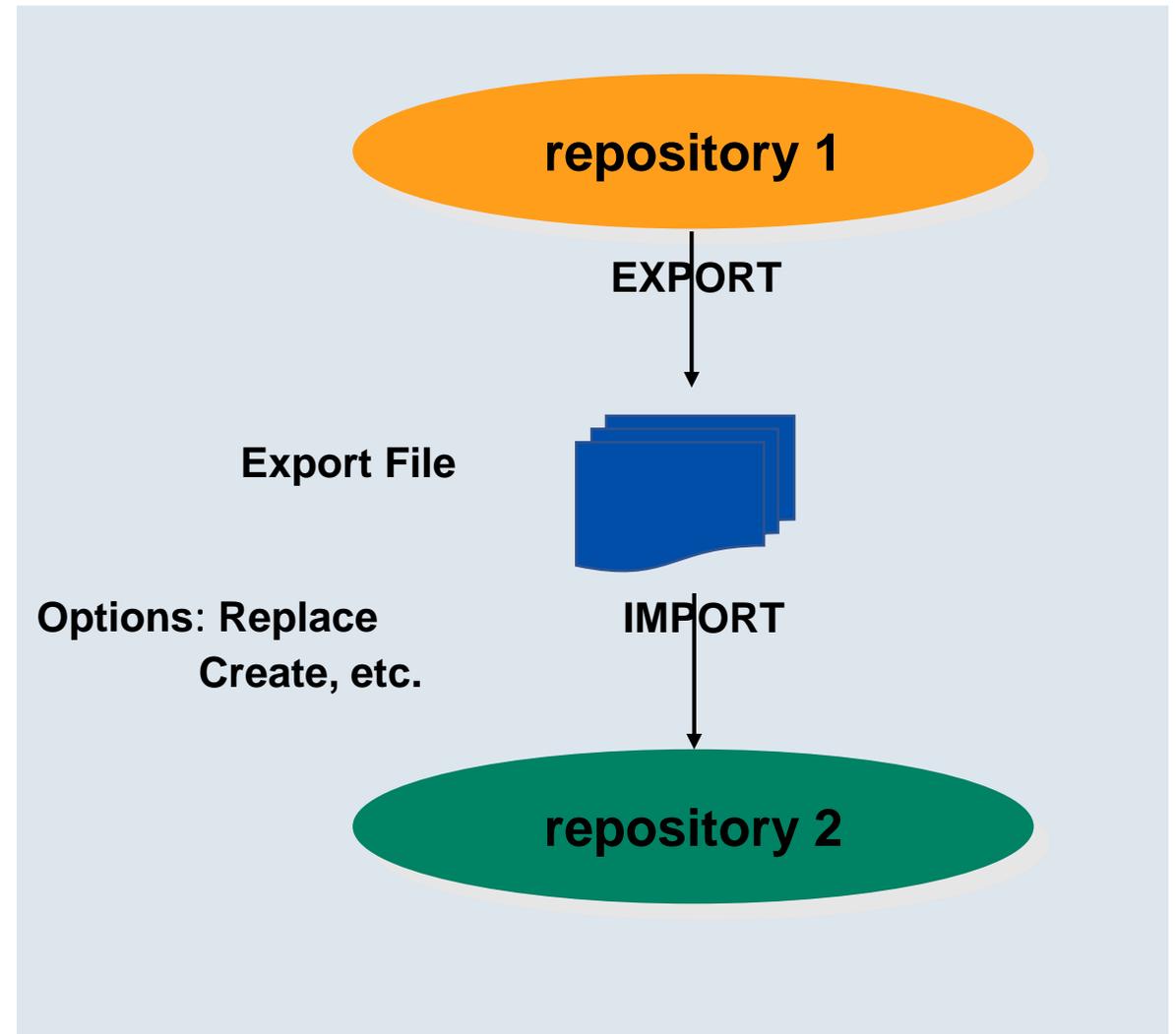
```

The 'File properties' dialog box for file CORT01 shows the following table:

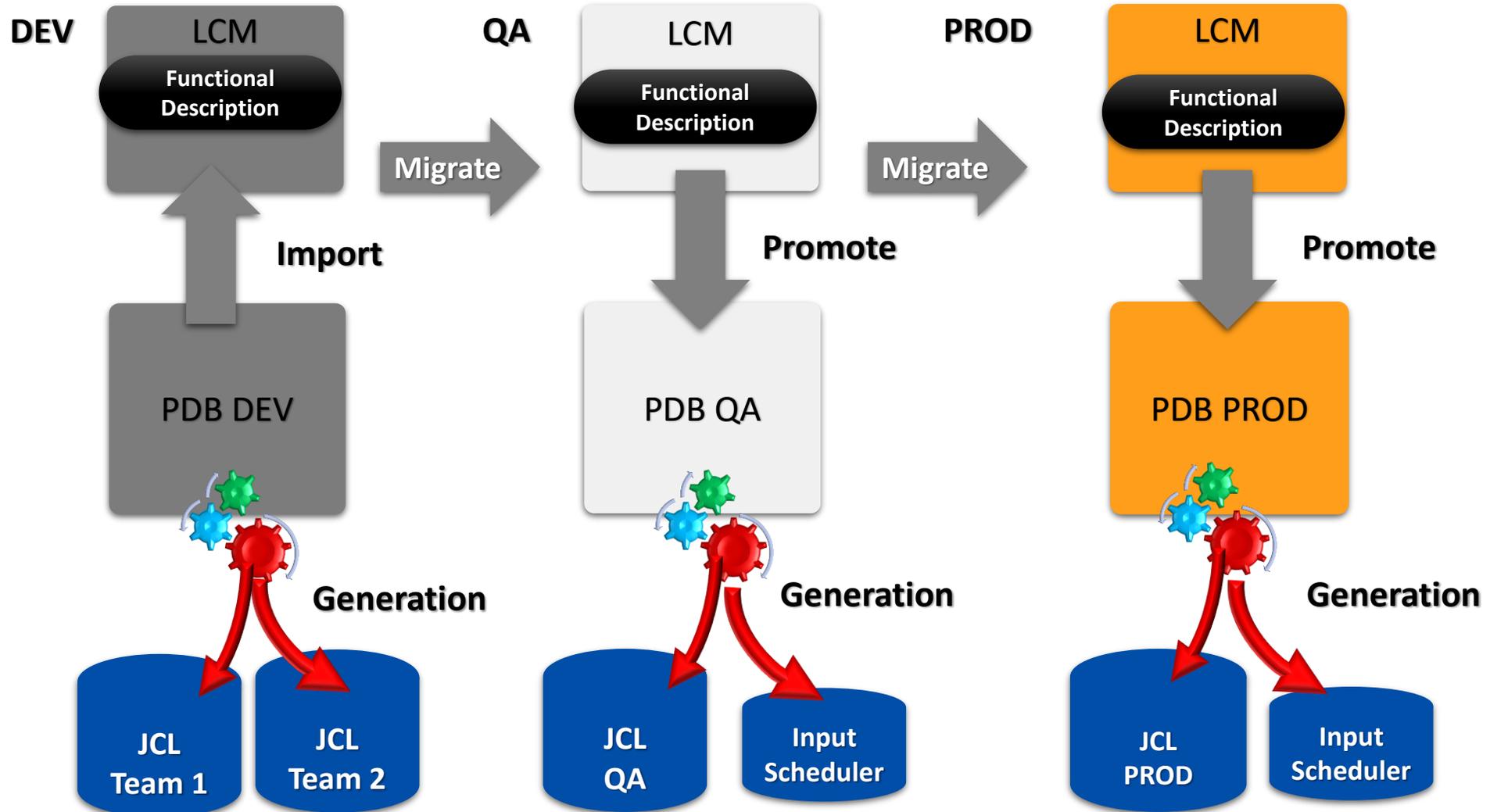
Functional Attributes		Technical Attributes		MVS Attributes				
General		Cross-refs		Used by...				
Jobset	Freq	Job-id	Step-id	Program	ddname	Mode	Shr	Pass
CRJ1	D	A0	A0A0	CZSP2WT	FIC01	o↑		NO
CRJ2	D	A0	A0A0	CZSP2WT	FIC01	i↓		NO
CRJ2	D	A0	A0A5	SORT	SORTIN	i↓		NO
CRJ5	D	A0	A0A0	CZSP2WT	FIC01	o↑		NO
CRJ5	D	A0	A0A5	CZSP2WT	FIC01	i↓		NO
CRM1	M	A0	A0A0	CZSP2WT	FIC01	o↑		NO
CRF0	R0	A0	A0A0	IEFB14	CORT01	o↑		NO
CRW1	W	A0	A0A0	CZSP2WT	FIC01	i↓		NO
DEMO	D	A0	A0A0	IKJEFT01	DD1	i↓		NO
DEMO	D	A0	A0A0	IKJEFT01	FIC01	o↑		NO
DEMO	D	A0	A0A5	CZSP2WT	FIC01	o↑		NO
DEMO	D	A0	A0B0	CZSP2WT	FIC01	o↑		NO
DEMO	D	A0	A0C5	IEBGENER	SYSUT1	i↓		NO

Architecture Flexibility

- ASG-Cortex PDB allows different architectures:
 - Several set of rules in one unique PDB repository (so called 'standards')
 - Several PDB repositories exchanging functional descriptions (PCL packages)
 - Mix of multiple repositories and multiple set of rules
 - One repository can support 1 to 26 set of rules
- Tools Import and Export allows to exchange PCL from one PDB to another PDB
- Exchanges can be controlled through a Life Cycle Manager (LCM, ENDEVOR, CHANGEMAN, ...)

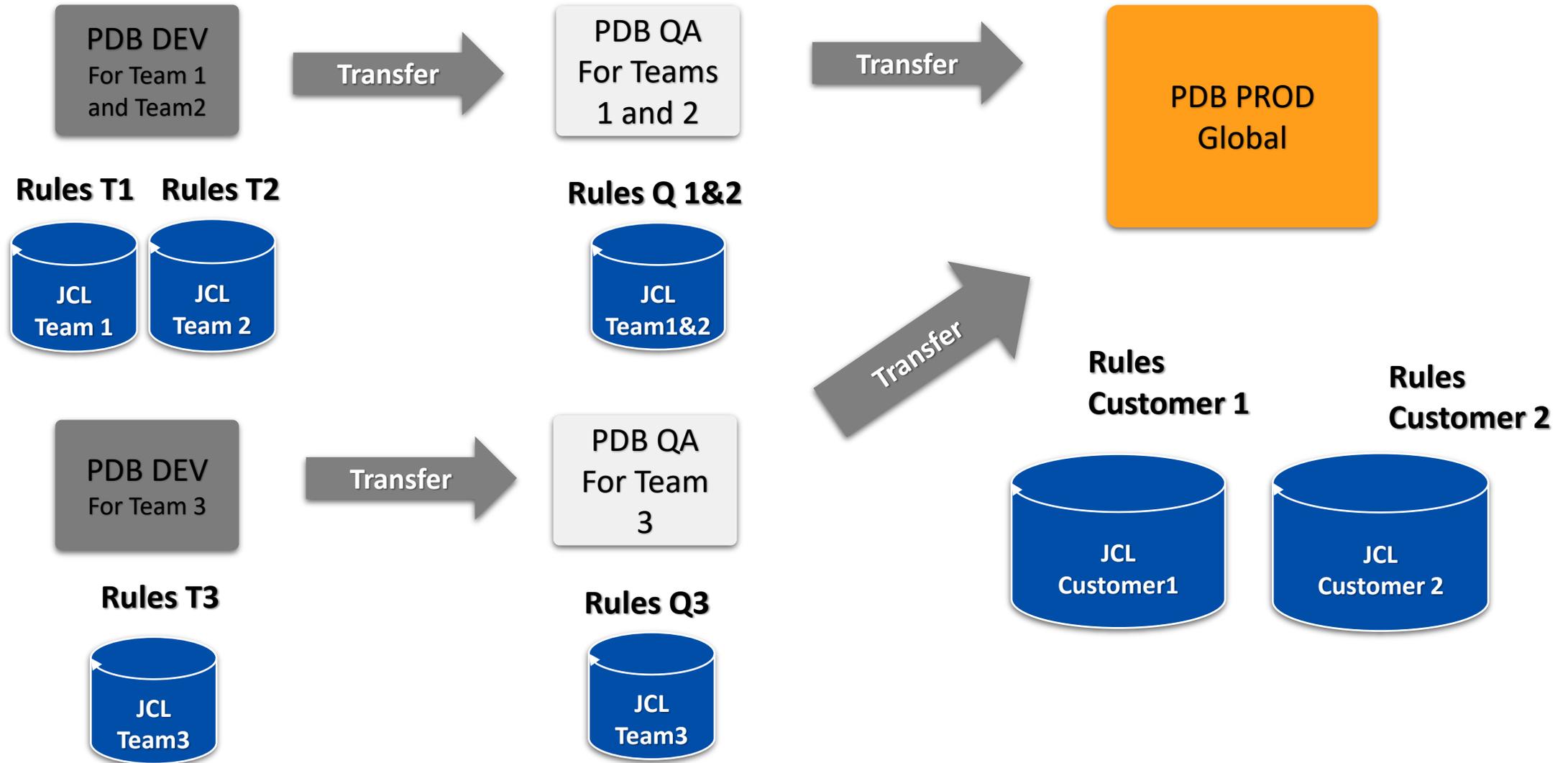


Organize the DEVOPS lifecycle





Facilitate Agile working



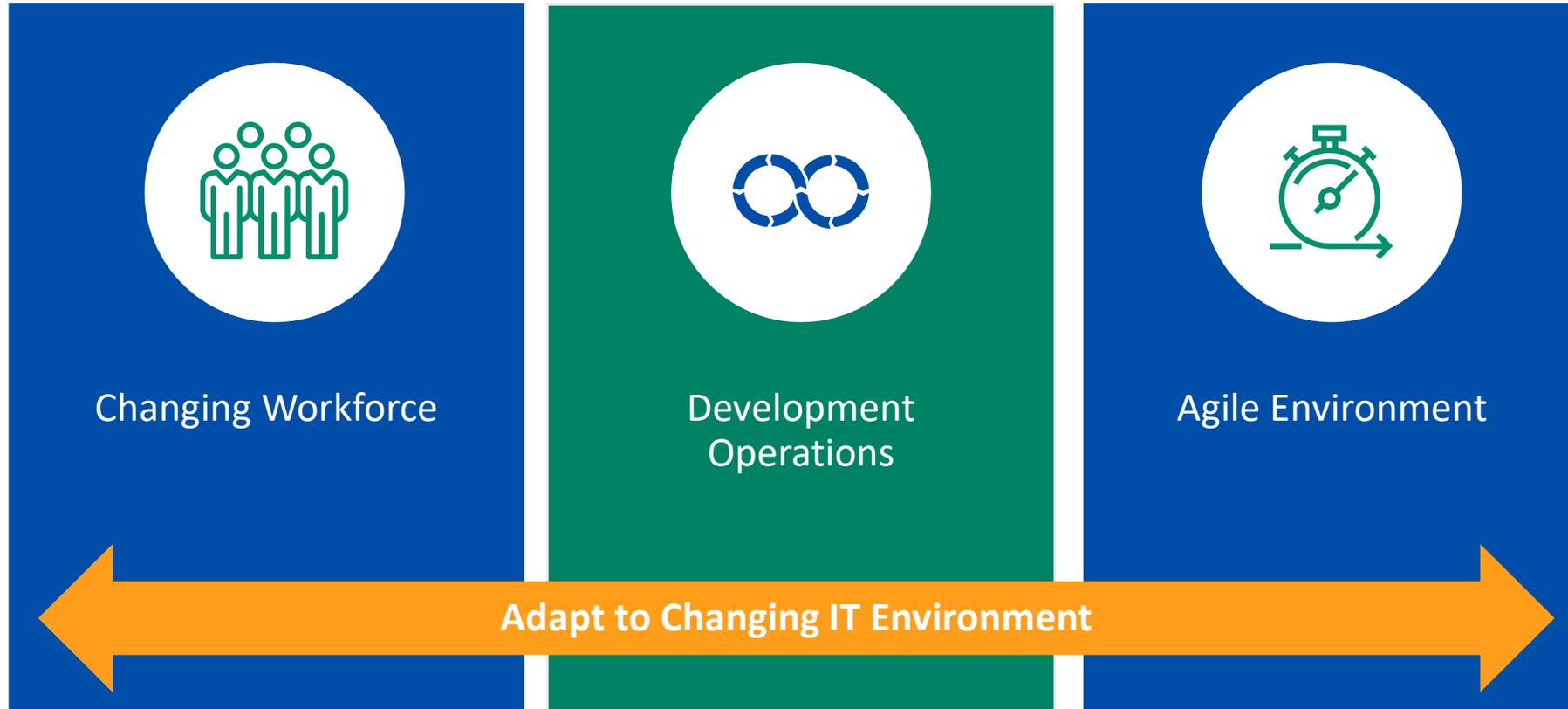
Why generation of JCL is a modern approach

- Your staff do not require JCL coding skills
- You get an easier and faster way to go from Dev to Ops
- You are sure the constraints of each stage (Dev, Qualification, Preprod, Prod) are **strictly** followed
- Everybody **works the same way**, avoiding loss of time when an other person has to check or maintain a JCL
- You gain **productivity**, because adding technical steps or transforming a dataset name is not productive and prone of errors
- If you work for different customers with same application, you can generate the JCL ad hoc for each one of them



And in Conclusion.....

Initiatives in a Mainframe World



THANK YOU!

Discussion & Questions?

Please submit your session feedback!

- Do it online at <http://conferences.gse.org.uk/2019/feedback/ML>
- This session is **ML**

1. What is your conference registration number?

 This is the three digit number on the bottom of your delegate badge

2. Was the length of this presentation correct?

 1 to 4 = "Too Short" 5 = "OK" 6-9 = "Too Long"

1 2 3 4 5 6 7 8 9

3. Did this presentation meet your requirements?

 1 to 4 = "No" 5 = "OK" 6-9 = "Yes"

1 2 3 4 5 6 7 8 9

4. Was the session content what you expected?

 1 to 4 = "No" 5 = "OK" 6-9 = "Yes"

1 2 3 4 5 6 7 8 9

Place your custom session QR code here. Please remove the border and text beforehand.