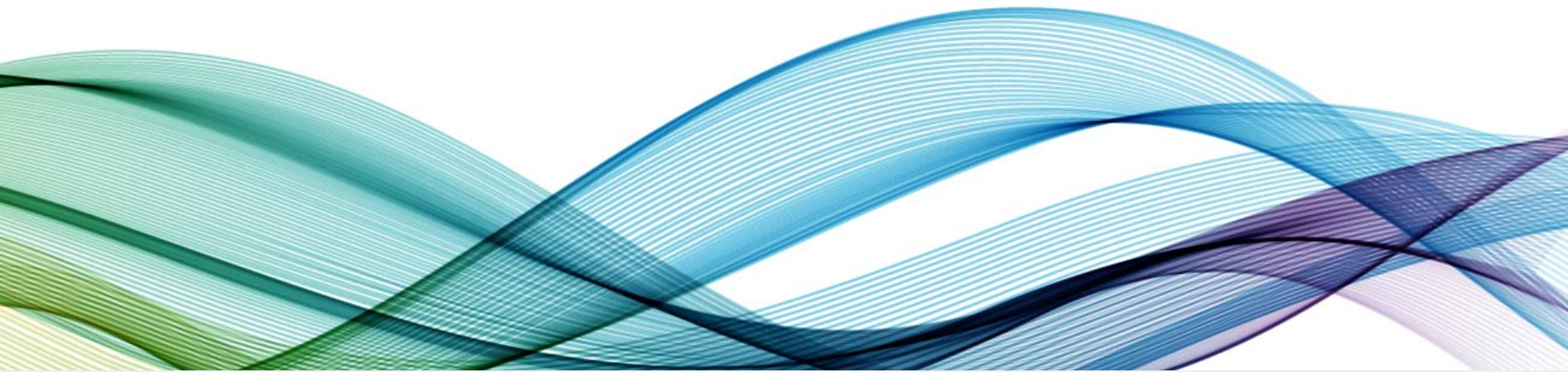




QuintilesIMS™

CompareDirectory – Macro bits and pieces in action

Lawrence Heaton-Wright, QuintilesIMS



CompareDirectory macro

- The Problem
- The Solution in Detail
- Questions

The Problem

- Sometimes we need to re-run a whole batch of outputs
- We want to be able to ascertain what outputs have changed from the previous delivery
- Sometimes these are expected changes
- We can use file comparison software (UltraCompare, ExamDiff, etc.) to check the contents of 2 directories
- Problem solved ... not quite!
- Our outputs usually have a date/time of production included
- Obviously a new run has a different date/time of production
- But the data in the table has not changed
- These are not actual differences
- And we want to focus on what outputs have actually changed

The Problem in Detail

Folder Compare - UltraCompare Professional

File Edit View Session Mode Options Merge Window Help

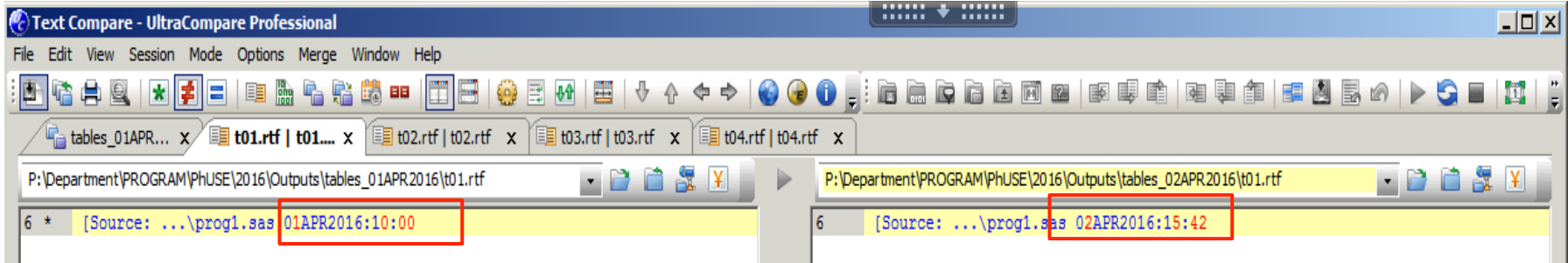
tables_01AP... x t01.rtf | t01.rtf x t02.rtf | t02.rtf x t03.rtf | t03.rtf x t04.rtf | t04.rtf x

P:\Department\PROGRAM\PhUSE\2016\Outputs\tables_01APR2016\ P:\Department\PROGRAM\PhUSE\2016\Outputs\tables_02APR2016\

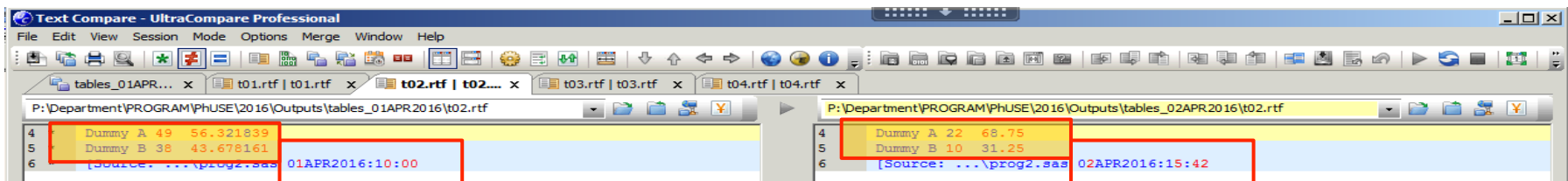
Name	Size	Date	Name	Size	Date
tables_01APR2016	15,332	27/07/2016 13:57:17	tables_02APR2016	15,324	27/07/2016 13:19:18
! t01.rtf	3,833	27/07/2016 14:00:56	! t01.rtf	3,833	27/07/2016 14:00:56
! t02.rtf	3,833	27/07/2016 14:00:56	! t02.rtf	3,825	27/07/2016 14:00:56
! t03.rtf	3,833	27/07/2016 14:00:57	! t03.rtf	3,833	27/07/2016 14:00:57
! t04.rtf	3,833	27/07/2016 14:00:57	! t04.rtf	3,833	27/07/2016 14:00:57

- As can be seen from this example we have re-run all of our outputs
- And according to the file comparison software they are all different
- So how do I identify which output has the actual data change(s) in it?
- I have to open each comparison manually to identify whether the summary data has changed or if it's just the date/time of production

The Problem in Detail

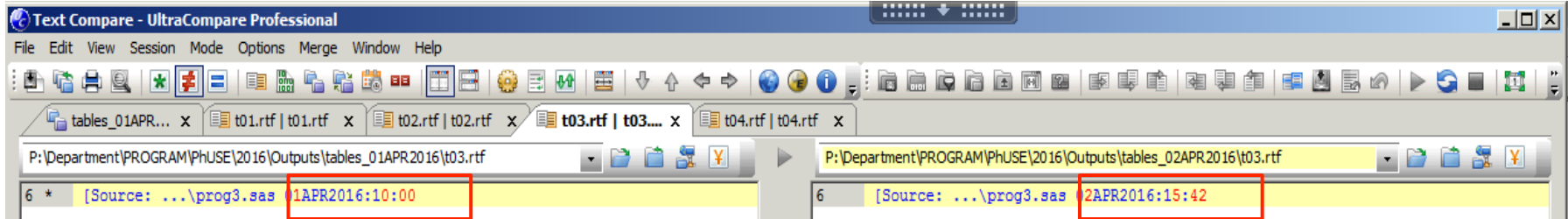


- The first compare shows that only the date/time of production has changed

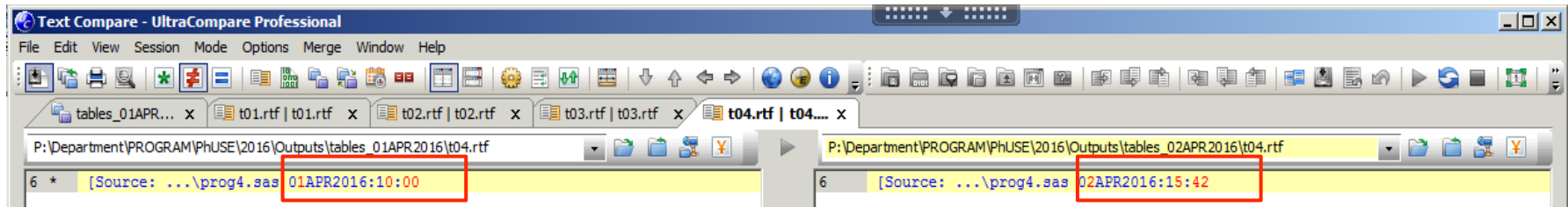


- The second compare shows some actual differences in the data
- I'm not interested in the actual differences at the moment
- I make a note that this output has data differences in it and I can pass this information onto the relevant programmers

The Problem in Detail



- The third compare shows that only the date/time of production has changed



- The fourth compare shows that only the date/time of production has changed

The Problem in Detail

- This process only took about 10 minutes as there were only 4 outputs
- This includes:
 - › Running the compare software for folder comparisons
 - › Opening each individual comparison for differences
 - › Informing the programming and validation team of which outputs to check
 - › The validation process involving the programming and validation programmers
- This doesn't seem too bad in terms of time and it didn't take very long ...

The Problem in Detail

- However, scale this up to 200 outputs based on about 30 seconds per compare
- $200 \times 30 \text{ seconds} = 100 \text{ minutes}$
- Clearly this is not an optimum use of time
- It's also a tedious, manual process



- There has to be a better way of identifying the actual data differences

The Solution

- We needed a method of identifying outputs that have true data changes in them
- As I'm a SAS® programmer I decided to use SAS
- After all the outputs (even RTF and DOC files) are essentially text files
- These files can be read into SAS and compared to see if there's any data changes
- And, crucially, I can use SAS functions and statements to exclude a specified string from consideration
 - › i.e. masking the date/time of production from the comparison process
- I had been investigating SAS external file functions (PhUSE 2013 CS02) and my mind turned to utilising these functions in some way
- The tedious output comparison section can be handled by the program and the user can add the value by just checking the actual data differences

The Solution in Detail

Identify files in common

- 2 folder locations supplied by user
- Identify files in each folder
- Compare these lists of files
- Those in common will be compared
- Those not in common reported

Identifying files in common

- The user inputs 2 macro variables: DIR1 and DIR2
- These are the location of the folders for comparison
- External file functions are used to identify the number of objects within each directory (MEMCOUNT macro variable)

```
%DO i=1 %TO 2;  
  %LET filrf=MYDIR;  
  %LET rc=%SYSFUNC(FILENAME(filrf,"&&Dir&i.."));  
  %LET did=%SYSFUNC(DOPEN(&filrf));  
  %LET memcount=%SYSFUNC(DNUM(&did));  
  %LET rc=%SYSFUNC(DCLOSE(&did));
```

Scan through each directory to determine the list of files with the specified extension

```
%END;
```

- When the FILENAME function is used within a %SYSFUNC then the file reference name is the name of a macro variable without the &.
 - › In this case FILRF

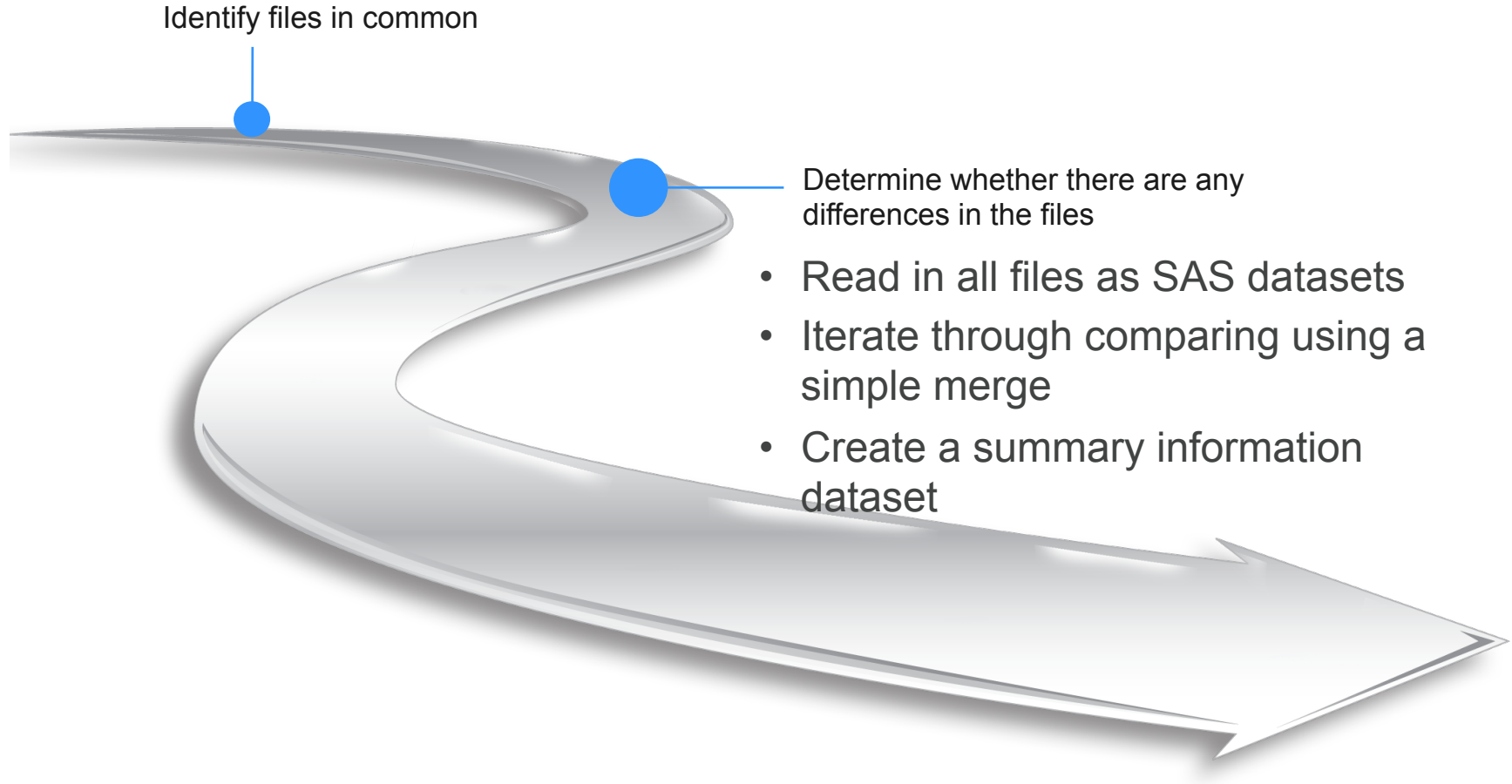
Identifying files in common

- Using MEMCOUNT iterate through every object in the folder, read in the filename, check whether the file has the required extension and send this to a dataset containing the files in folder 1 or 2

```
DATA _SASFiles_&i. (DROP=_:);
  ATTRIB
    __FileExt    LENGTH=$15.
    SASFile      LABEL="SAS File" LENGTH=$100.
    Dir&i.       LABEL="Directory &i." LENGTH=$200.
;
*** Open directory ***;
__rc=filename("mydir", "&&Dir&i..");
__did=dopen("mydir");
DO __i=1 TO &memcount;
  SASFile=DREAD(__did, __i);
  Dir&i.="&&Dir&i..";
  __FileExt=UPCASE(SCAN(SASFile, -1, '.'));
  IF __FileExt EQ "&FileType." THEN OUTPUT;
END; *** __I ***;
RUN;
```

- This type of construct (with some modification) can be used in many different macros to ascertain what files are in folders

The Solution in Detail



Determining Differences

- The SAS_FILES dataset contains the directory and file information of the files in common
- Iterate through all of these files creating Local macro variables for each pair of files

```
%LET StatusDSet=;
%IF &N_File. GT 0 %THEN %DO FileNum=1 %TO &N_File.;
  DATA _NULL_;
    SET SAS_Files (WHERE=(_file_id EQ &FileNum.));
    %DO i=1 %TO 2;
      CALL SYMPUTX ("File&i.",CATX("&dlm.",dir&i.,SASFile),'L');
    %END;
  RUN;
```

Read in each pair of files to 2 SAS datasets

Determining differences between these pairs of files

```
%LET StatusDSet=&StatusDSet. _Status_&FileNum.;
%END; %*** FileNum ***;
```

Create an overall status dataset detailing whether there are any differences between pairs of files for reporting

Determining Differences – reading in each pair of files

- Iterate through each pair of files
- In the previous step we have created FILE1 and FILE macro variables
- These will be used in the FILENAME statement as the input file

```
%DO i=1 %TO 2;  
  FILENAME SASFile "&&File&i..";  
  DATA _File_&i.;  
    INFILE SASFile LRECL=200 TRUNCOVER;  
    INPUT @1 line&i. $CHAR200.;  
  RUN;
```

Exclude text from file

Special case for RTF

Define merging variable for next section

```
filename SASFile;  
%END; *** I for selecting Files ***;
```

Determining Differences – excluding text from file

- Following reading the file into a dataset we now need to remove the text string that identifies the date/time of production
- For RTF files remove the line which contains the file metadata which is based on the line with Creation date/time in it

```
DATA _File_&i.;  
  SET _File_&i.;  
  IF index(UPCASE(line&i.),"%upcase(&TextExcl.)") gt 0 THEN DELETE;  
RUN;
```

```
%IF &FileType. in (RTF) %THEN %DO;  
  DATA _File_&i.;  
  SET _File_&i.;  
  IF index(UPCASE(line&i.),"CREATIM") gt 0 THEN DELETE;  
  RUN;  
%END;
```

Determining Differences – define a merging variable

- The next task is to define a merging variable (OBSNO)
- The theory is that if the files are identical (after removing the exclusion text) then the observation number will contain the same information for each file
- If not this indicates that there is a difference in the files

```
DATA _File_&i.;  
  SET _File_&i.;  
  ATTRIB  
    ObsNo LABEL="SAS File Line Number"  
  ;  
  ObsNo=_n_;  
RUN;
```

Determining Differences – between each pair of files

- Following this the 2 datasets containing the output (as a dataset) are then merged together
- Using a simple variable comparison any differences in the outputs are detected and a single observation is sent to a findings dataset
 - › COMPBL is used to ignore differences in white space as we are interested only in differences to the actual text and values

```
DATA _CheckDiff;  
  MERGE _File_1 _File_2;  
  BY ObsNo;  
  _file_id=&FileNum.;  
  IF COMPBL(line1) NE COMPBL(line2) THEN _difference=1;  
RUN;
```

```
DATA _Status_&FileNum. (KEEP=_file_id AnyDiff);  
  SET _CheckDiff END=eof;  
  ATTRIB  
    AnyDiff LABEL="Any Differences" LENGTH=$1.  
  ;  
  RETAIN AnyDiff 'N';  
  IF _difference EQ 1 THEN AnyDiff='Y';  
  IF eof THEN OUTPUT;  
RUN;
```

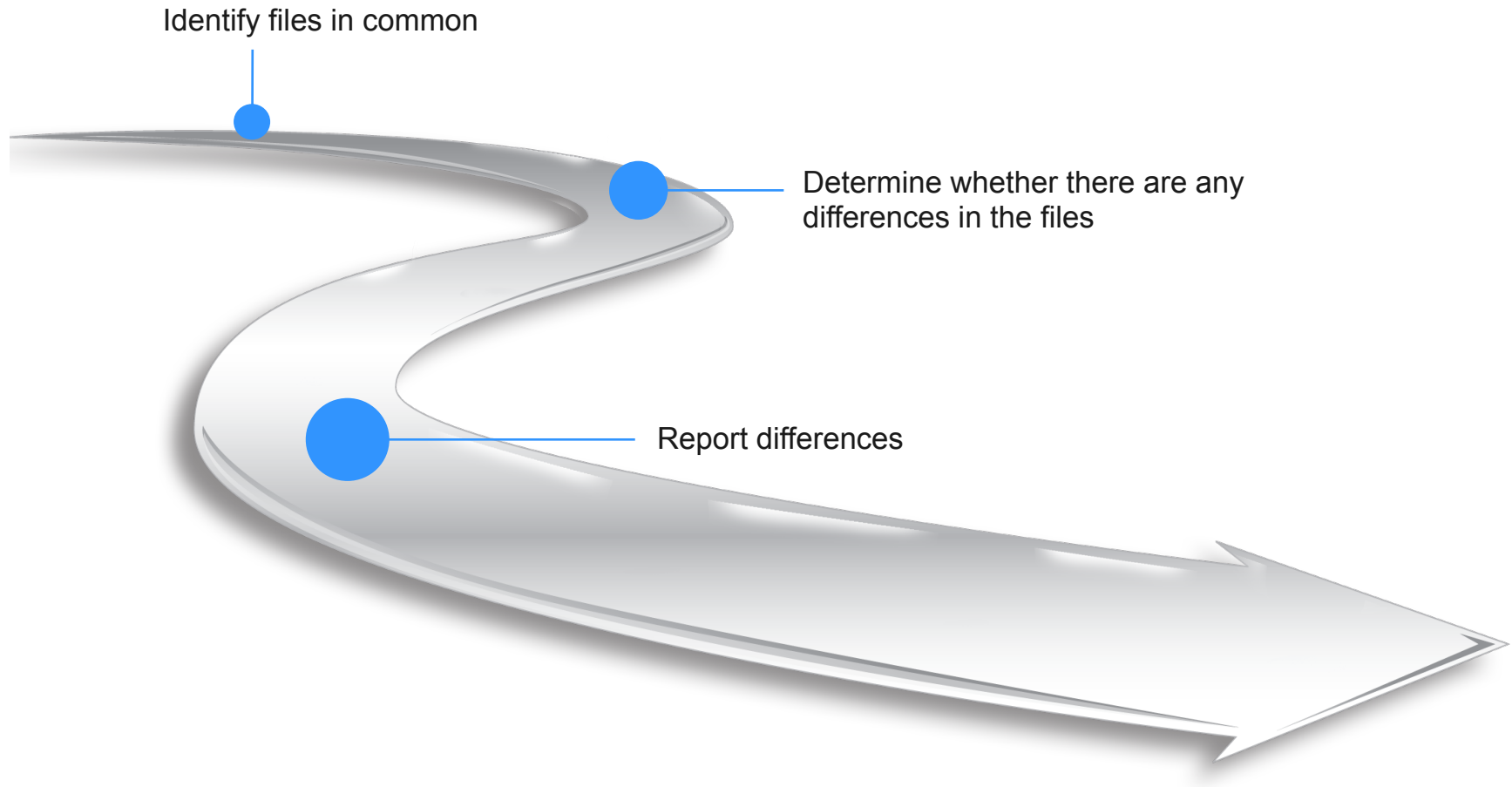
Determining Differences – between each pair of files

- The STATUSDSET macro variable is initialised prior to iterating through the pairs of files
- This will then have a list of dataset names which is then used to create the overall status dataset of the findings of the pairs of files using a SET statement
 - › Used in place of a PROC APPEND
 - › The dataset created will not need to be removed from the WORK library if the macro is called multiple times within the same program
 - › PROC APPEND will continually add to an existing dataset unless the WORK library is cleaned up

```
DATA Status;  
    SET &StatusDSet.;  
    BY _file_id;  
RUN;
```

- This dataset is then used for reporting purposes

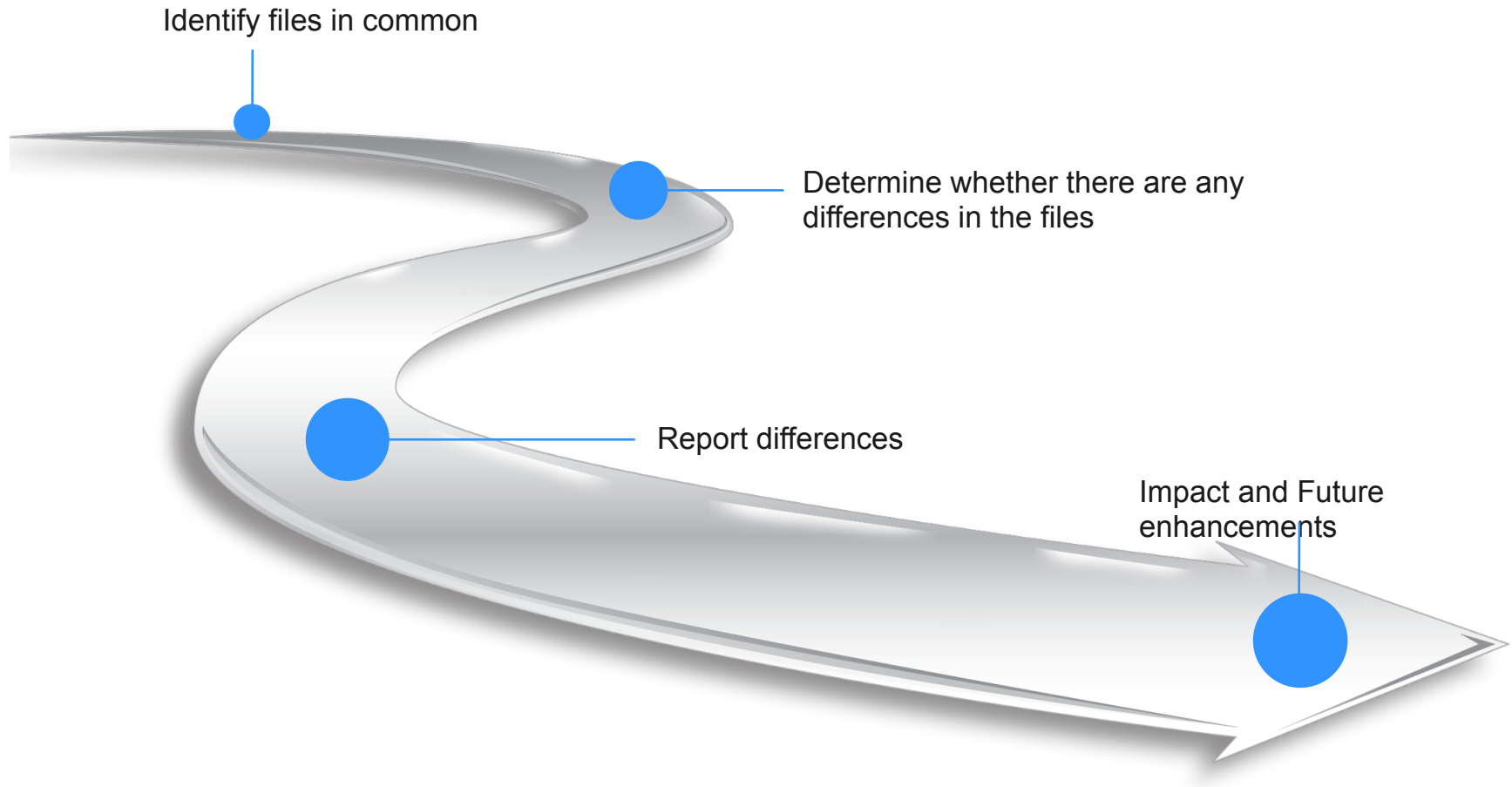
The Solution in Detail



Reporting Differences

- Define a macro variable if there are any differences
- This way %IF statements can be used to selectively print reports
- Reports are generated showing
 - › Files where differences occur
 - › Files with no differences
- Information reported includes
 - › how many files have differences
 - › how many files were compared
 - › where these files are located
 - › what the text excluded is
- If there are any files not in common this information is also reported.
- The reports are simple PROC PRINTs
- An enhancement from the original program was sending the output to RTF

The Solution in Detail



Future enhancements

- A clean-up macro could be added to clear out WORK datasets and global macro variables that are produced during the execution of the macro
- Add an option to send the reports to an Excel workbook in place of the RTF file/output window.
- Add an option to ignore case for text differences
- Add an option to ignore white space for text differences
 - › This is the default behaviour of the macro but could be switched off

Impact

- The use of this macro has saved us a considerable amount of time as it allows us to concentrate on the true differences in outputs
- Using SAS to perform the tedious, repetitive work allows us as programmers to concentrate on adding value by only checking what has truly changed
- The use of external file functions should allow this macro to work on any operating system
- It was also fun to produce 😊

Your comments and questions are valued and encouraged.

Contact the author at:

Lawrence Heaton-Wright

QuintilesIMS

500 Brook Drive,

Green Park,

Reading,

Berkshire.

RG2 6UU

Phone: +44 118 4508320

Email: lawrence.heaton-wright@quintilesims.com

Web: www.quintilesims.com