

YBS

ORACLE FORMS

APPLICATION STRATEGY

IN A SOA WORLD

Created by: Graham Brown, Application Architecture Manager

Public



AGENDA

- Background to Yorkshire Building Society
- History of YBS Oracle Forms Application
- Strategic Roadmap
- Analysing the application using PITSS
- The journey to SOA supported by PITSS
- Next steps for YBS



BACKGROUND TO YORKSHIRE BUILDING SOCIETY



YORKSHIRE BUILDING SOCIETY

- Financial Services organisation - Mortgages and Savings our key business
- Strong Mutuality agenda - everything we do is for the benefit of our members
- Between 2008 and 2012, we went through significant Merger and Acquisition activity
- YBS Group now contains multiple brands..
- 3.5m Customers
- £37.6bn Assets
- Over 4,500 Employees





HISTORY OF YBS ORACLE FORMS APPLICATION



FORMS APPLICATION - HISTORY

- Initially developed during the late 90's - driven by the need to move from a mainframe to overcome Y2K issues
- Developed using Oracle Forms 4.5 Client/Server - initial module definitions generated from Oracle Designer 2000
- Migrated through versions - currently 10gR2 - plans for 11gR2 this year
- Mid-tier forms - adopted for business rather than technology reasons
- Evolved into a large application - 000's of modules
- Large supporting database - again, 000's objects
- Application is not just Oracle Forms - Also utilise lots of COBOL (overnight processing), Java (Web channel), MS VB/.Net (Branch channel)

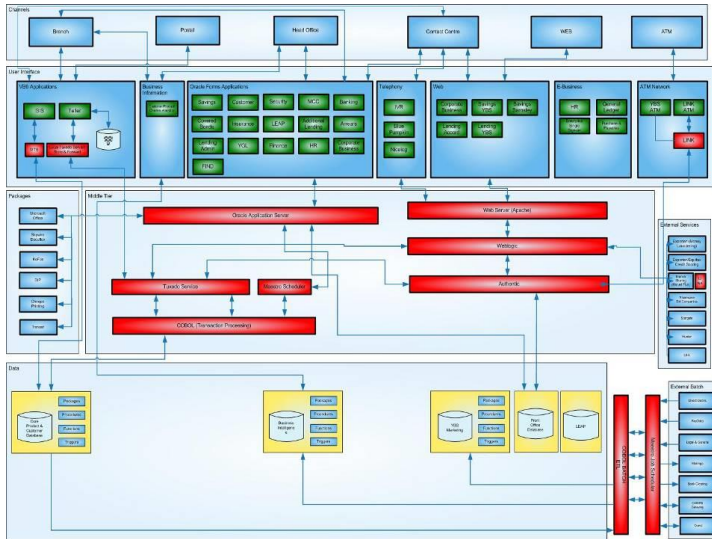


STRATEGIC ROADMAP

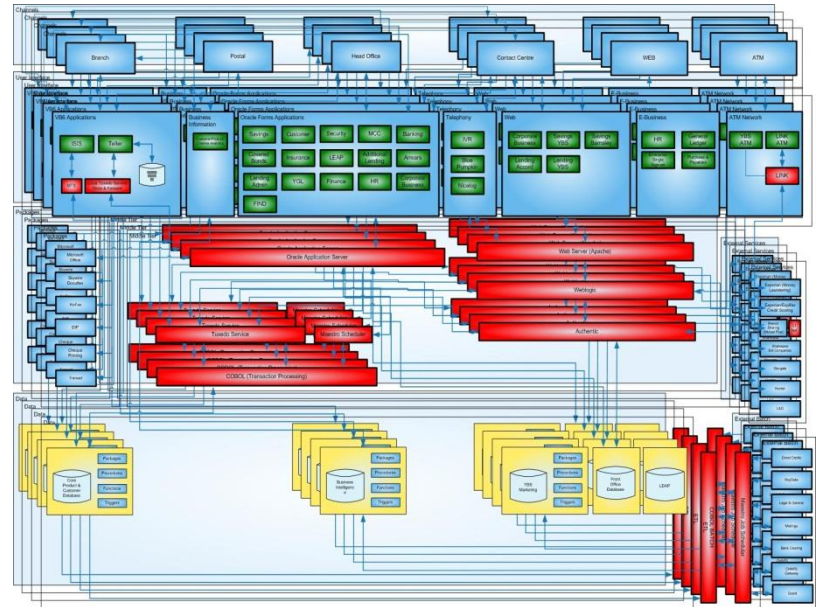


BRAND CLONING

- Single Brand architecture



- Multi - Brand cloned architecture

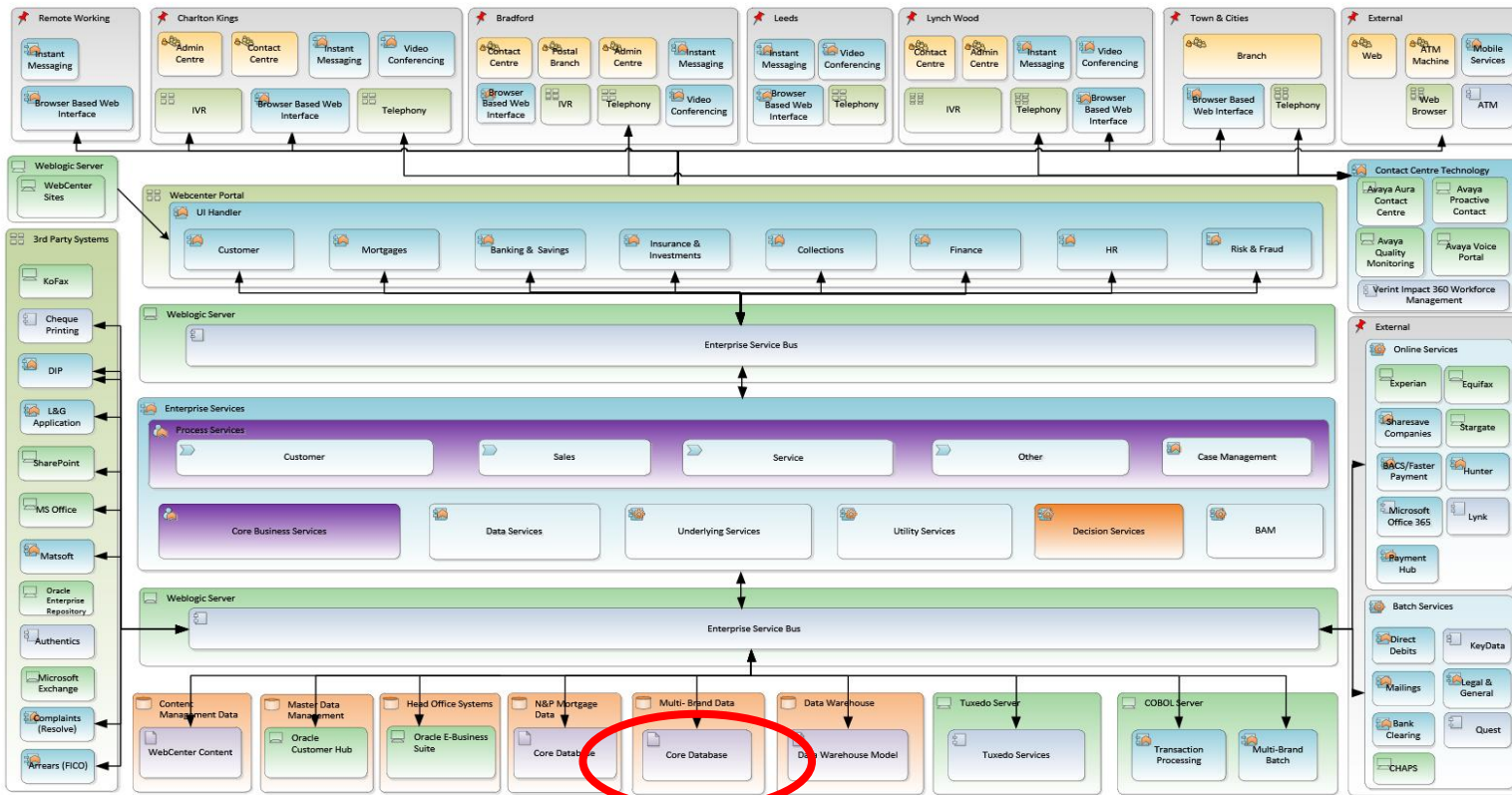


- Our architecture was starting to constrain our business agility!



STRATEGIC ROADMAP

- Organisational growth aspirations - £50bn assets 5 million customers within 5 years
- Re-engineering not just technology but Business and associated processes
- Future state architecture - SOA and BPM. Improve IT agility to satisfy business requirements



- Centred around a single, brand aware database
- No appetite to do a wholesale conversion of our Forms to ADF (yesterdays application in todays technology)



WHY PITSS?

We faced a number of challenges to be able to deliver our strategy:

- Declining pot of Subject Matter Experts - both Technical & Business who understand the existing application functionality
- Minimal system documentation for the existing applications - it has evolved over the last 18 years
- Needed a mechanism for identifying business logic and enabling informed decisions to be made to help move us in the strategic direction
- Choice of doing this manually - slow, resource heavy, and possibly inaccurate

or we look for tooling to assist

- A small number of tools out there, but PITSS was the only tool we identified which worked at the code level we required
- We ran a proof of Concept activity to evaluate PITSS in late 2012. It worked for Forms, Reports and Database
- COBOL could be loaded in and allowed simple textual search but didn't provide the full impact analysis we required
- With an assurance a COBOL parser could be provided, we purchased a number of PITSS modules in Mid 2013
- Modules purchased - Technology Base, Maintenance/Development, Application Analysis, Application Engineering and Source Code Analytics



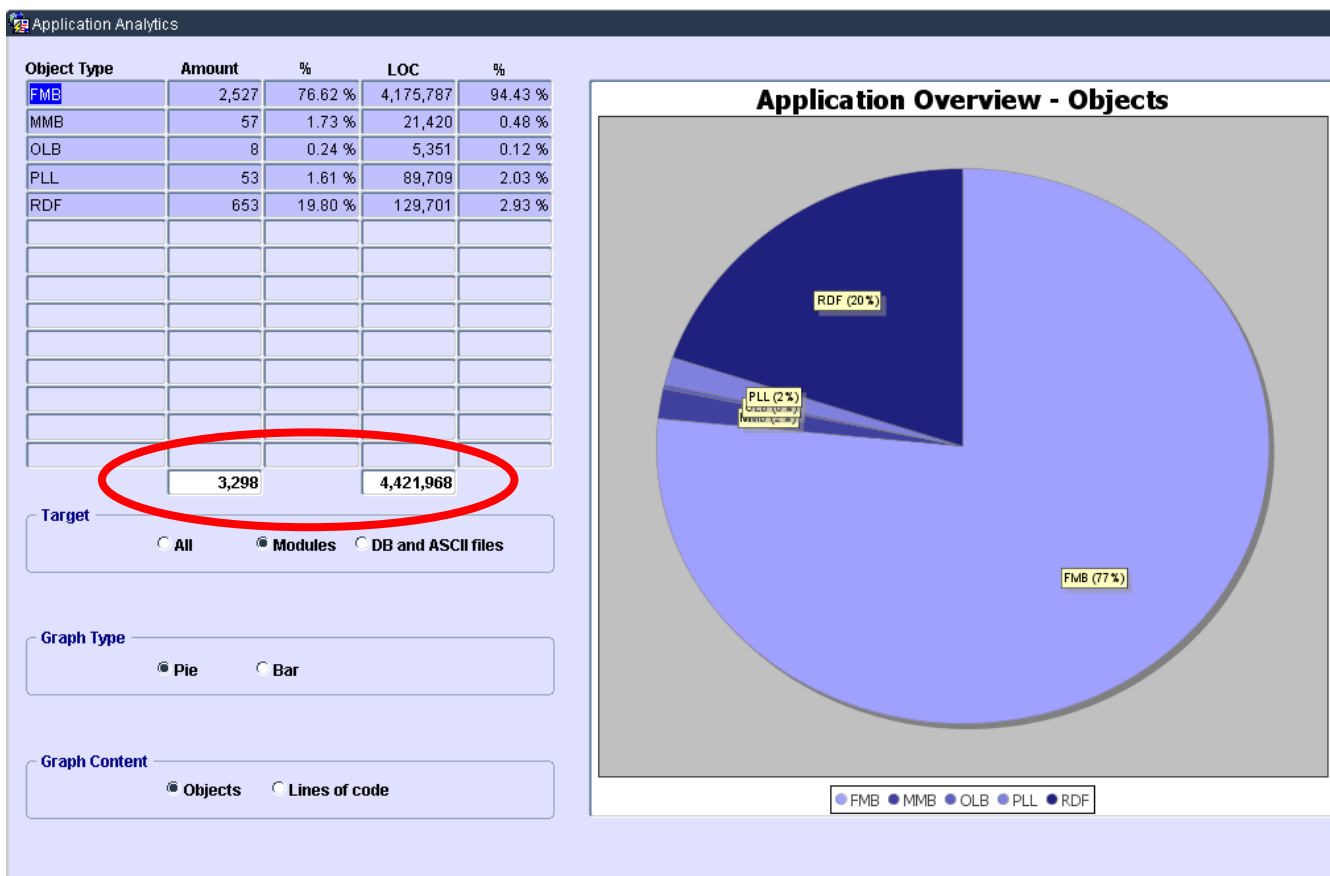
ANALYSING THE APPLICATION

PITSS SOURCE CODE ANALYTICS



WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

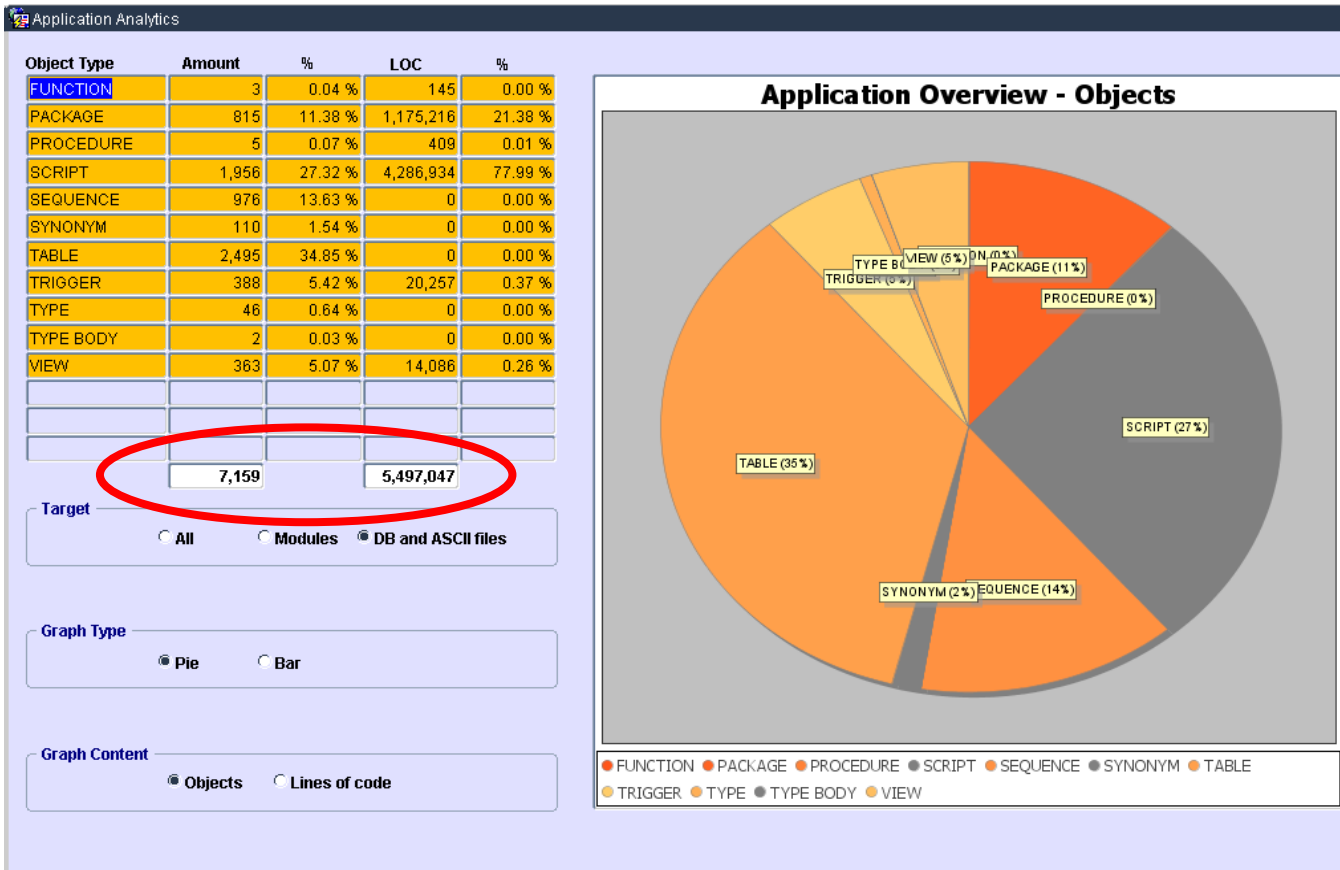
- We know we have a large application, but now we can quantify it...





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

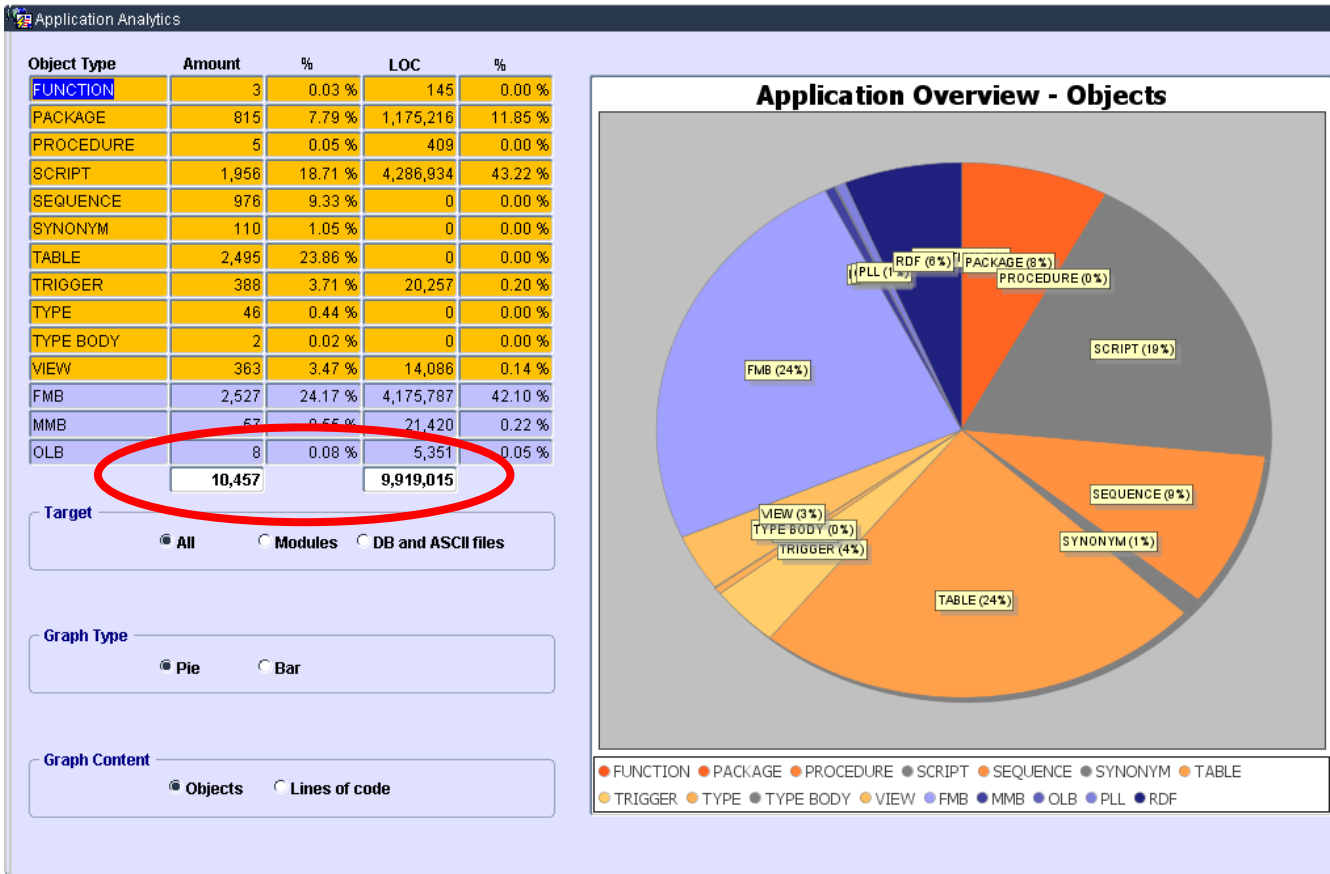
- We know we have a large application, but now we can quantify it...





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

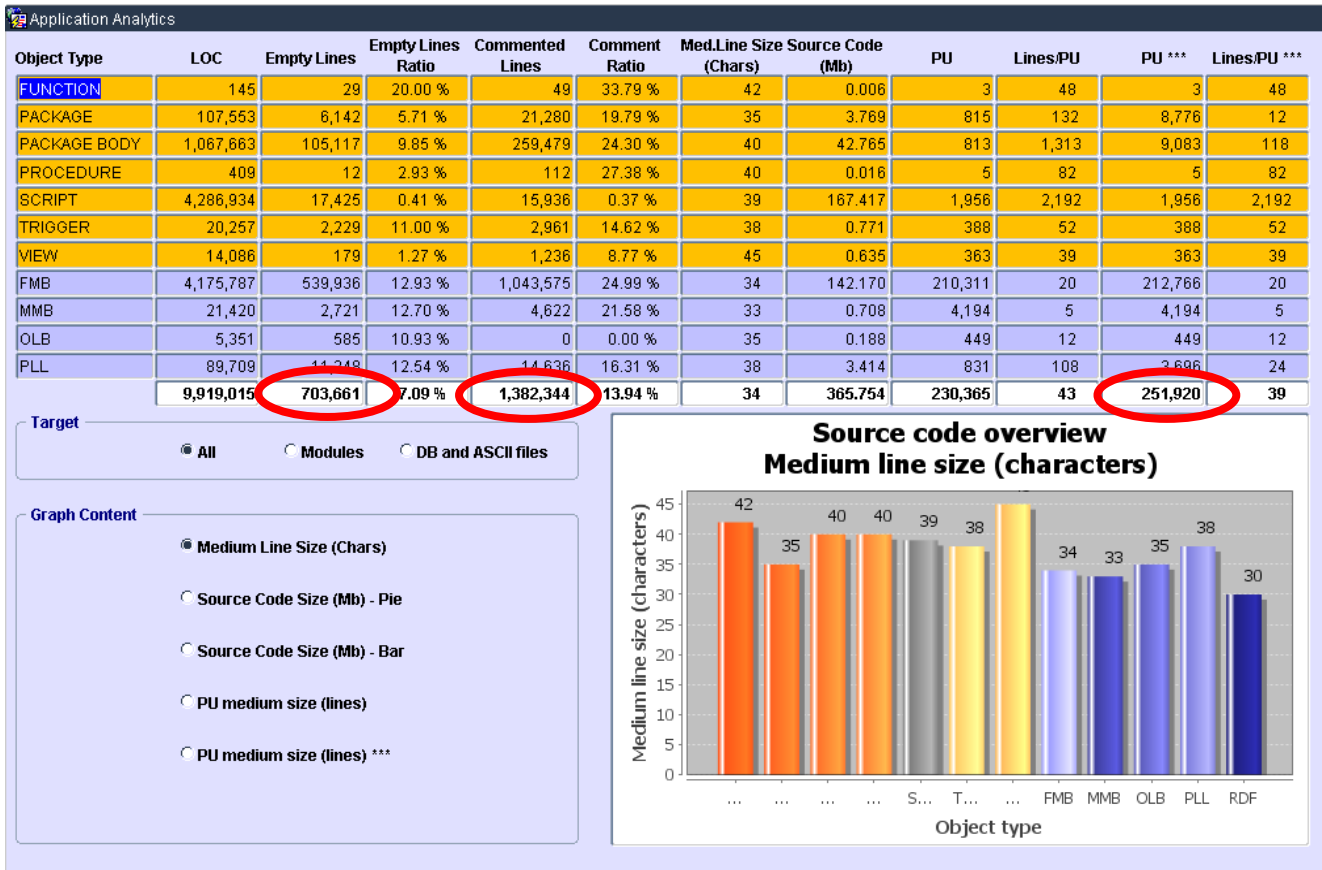
- We know we have a large application, but now we can quantify it...





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

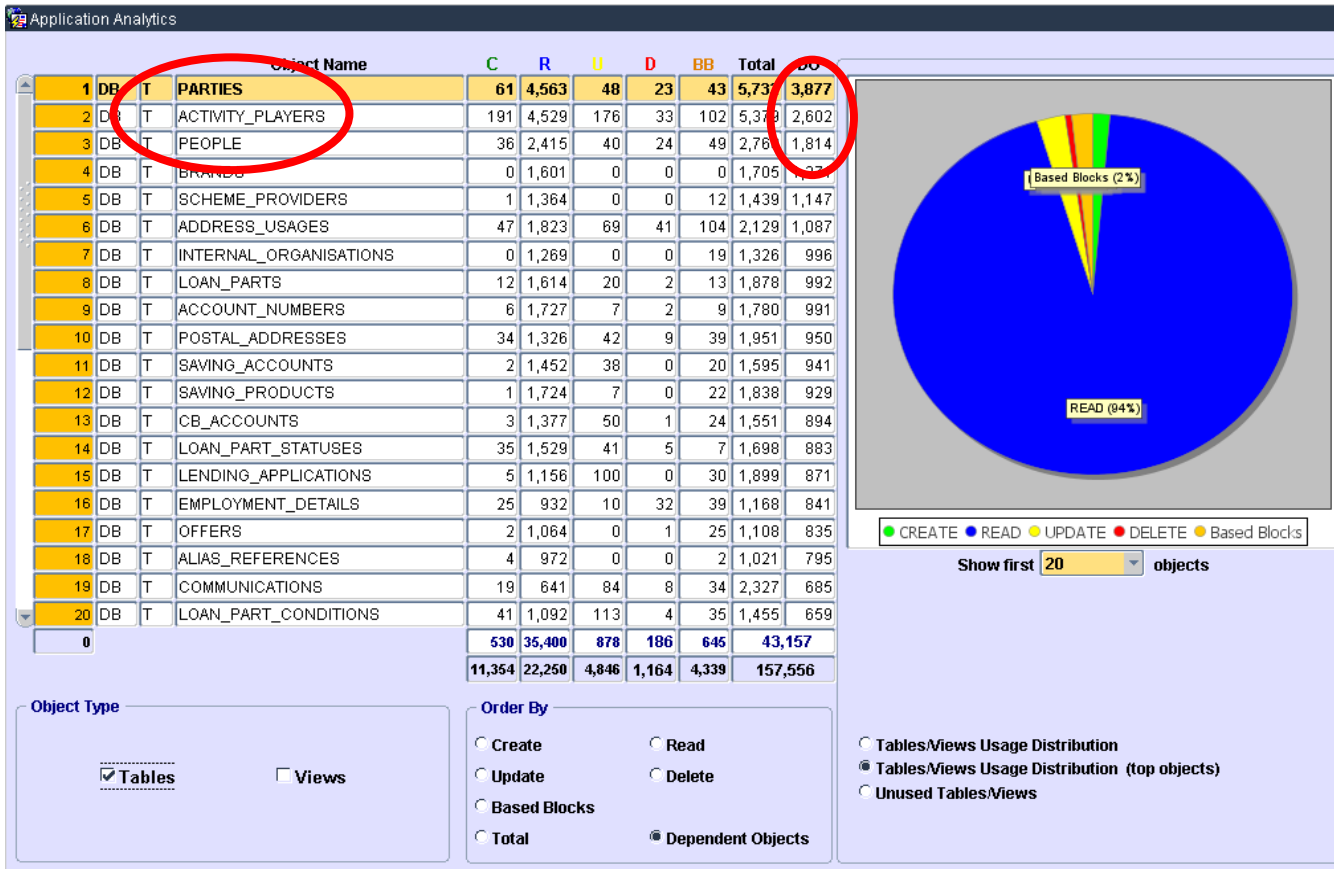
- We know we have a large application, but now we can quantify it...





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

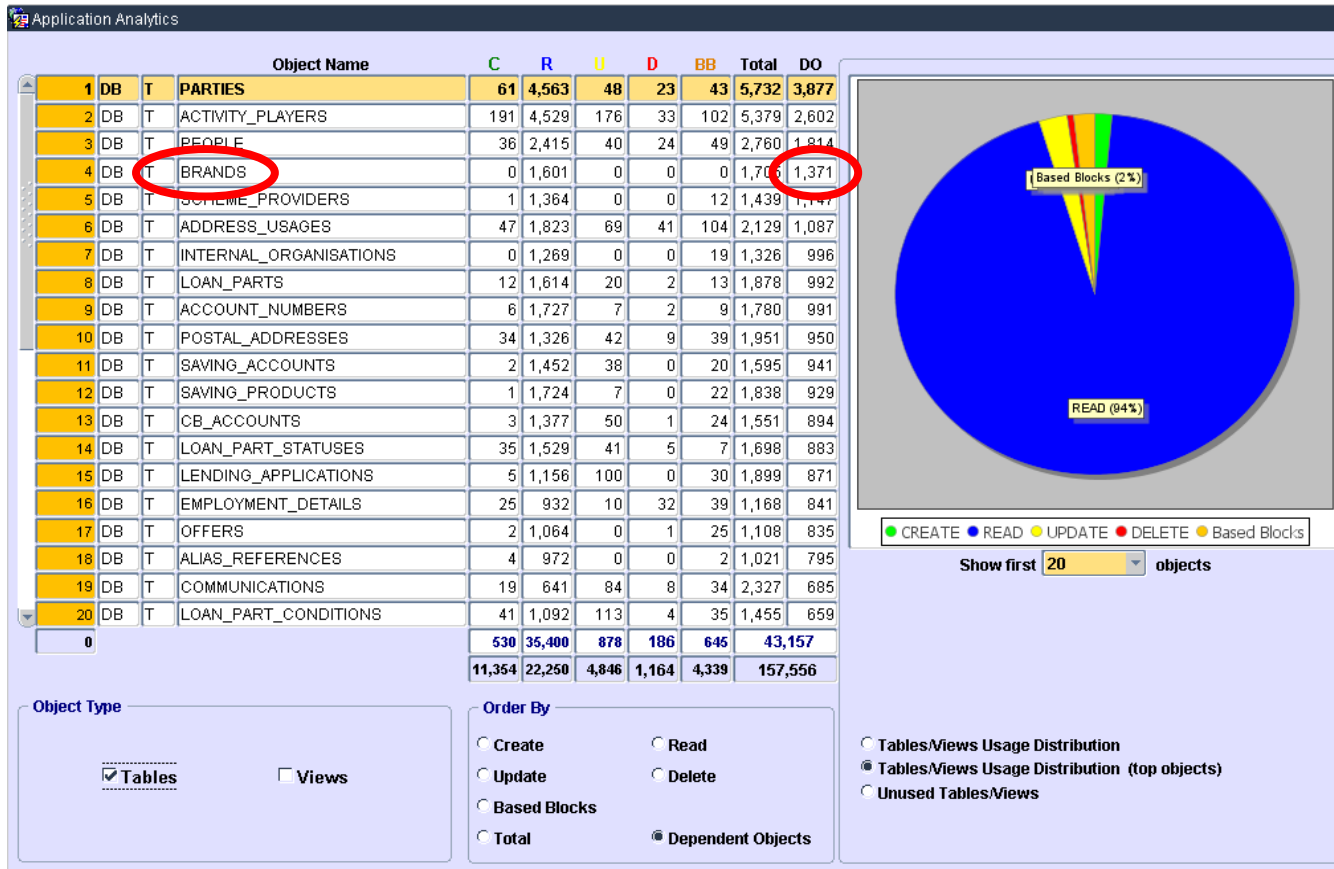
- So, we have a complex application. But how complex is it?





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- We initially thought we would only need to brand a handful of tables!





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- We were surprised to find we have a number of unused tables!

Application Analytics

	Object Name	C	R	U	D	BB	Total	DO
1	DB T NJ2	0	0	0	0	0	0	0
2	DB T NEIL2	0	0	0	0	0	0	0
3	DB T NEIL	0	0	0	0	0	0	0
4	DB T BANK_TRANS_SPLITS	0	0	0	0	0	0	0
5	DB T CST_ISC_WORK_APPLIC	0	0	0	0	0	0	0
6	DB T CST_ISC_TABLE	0	0	0	0	0	0	0
7	DB T CST_PROFNAME	0	0	0	0	0	0	0
8	DB T CST_NEW_BUSINESS	0	0	0	0	0	0	0
9	DB T HIST_MAINTENANCE_TRANSACTIONS	0	0	0	0	0	0	0
10	DB T HIST_ICBS_TXN_CODES	0	0	0	0	0	0	0
11	DB T GL_SETS_OF_BOOKS_COPY	0	0	0	0	0	0	0
12	DB T GL_POSTING_HISTORY	0	0	0	0	0	0	0
13	DB T GG_EVENT	0	0	0	0	0	0	0
14	DB T GGTEST	0	0	0	0	0	0	0
15	DB T GENTIA_SQL	0	0	0	0	0	0	0
16	DB T GAE_LIVE_ACCOUNTS	0	0	0	0	0	0	0
17	DB T GAE_CUS0936B	0	0	0	0	0	0	0
18	DB T GAE_ACCOUNTS	0	0	0	0	0	0	0
19	DB T FTP_PARAMETER	0	0	0	0	0	0	0
20	DB T F8CS_SAVING_PRODUCT_PROVISIOI	0	0	0	0	0	0	0
323		0	0	0	0	0	0	0

Object Type: Tables Views

Order By: Create Read Update Delete Based Blocks Total Dependent Objects

Tables/Views Usage Distribution: Tables/Views Usage Distribution Tables/Views Usage Distribution (top objects) Unused Tables/Views

Unused Tables/Views

Legend: ● Used objects ● Unused objects

Show first 20 objects



WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- Source Code Metrics - the number of Statements within a module

Application Analytics

	Object Name	Statem. Complexity	Cyclomatic Complexity	Halstead Volume	Maintain. Index
1	FMB P PR_SET_PARM_OPTS_BNK	778	24	32,532	-49
8	FMB T KEY-COMMIT	363	178	7,749	-69
9	FMB P CGRI\$CHK_ACCOUNT_NUMBERS	357	132	2,090	14
10	FMB P PR_REALLOCATION_OF_PAYMENT	355	151	11,742	-65
11	FMB P PROCESS_NEXT_BLOCK	353	71	7,592	-46
12	FMB P CGRI\$CHK_ACCOUNT_NUMBERS	335	124	1,943	-1
13	PLL P COMM_PACK.PR_BATCH_LET	333	96	9,179	-49
14	FMB P LEA3550S.PR_POST_FORM_COMMIT	324	99	5,737	43
15	FMB P PR_EXECUTE	323	102	7,734	-9
16	FMB P PR_POSTING_DATA	309	70	5,293	11
17	FMB T WHEN-NEW-RECORD-INSTANCE	309	74	9,879	15
18	FMB T WHEN-NEW-RECORD-INSTANCE	308	74	9,755	26
19	FMB T KEY-COMMIT	305	89	8,337	-7
20	PLL P SQA_EVT_HANDLER	304	103	4,585	38

Source Code Metrics (Statements)

Number of Statements

- 0 - 100 Small: Further functional decomposition probably unnecessary, cohesion probably not an issue
- 101 - 250 Medium: Probable candidate for functional decomposition, some cohesion improvements may exist
- > 250 Large: Likely candidate for functional decomposition, numerous cohesion improvements probably exist

Legend: Medium (92%), Large (8%)

Show first 500 objects

Source Code Metrics (top objects)

Statements: 0, 402, 0, 38

Location: DB FMB PLL OLB MMB RDF

Object Type: Package (object) Package (details) Procedure Function Trigger Menu Item Built-in

Order By: Statements Cyclomatic Complexity Halstead Volume Maintainability Index Object Usage LOC



WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- Cyclomatic Complexity - How easy is it to test?

Application Analytics

	Object Name	Statem.	Cyclomatic Complexity	Halstead Volume	Maintain. Index
1	FMB P PR_OLD_VALIDATE	436	218	7,117	-79
2	FMB T KEY-COMMIT				
3	FMB P PR_REALLOCATION_OF_PA				
4	FMB T WHEN-MOUSE-DOUBLECLI				
5	FMB P PR_CHECKING_EVENTS				
6	FMB P PR_CHECKING_EVENTS				
7	FMB P CGRISCHK_ACCOUNT_NUM				
8	FMB P PR_CHECKING_EVENTS				
9	FMB P PR_CHECKING_EVENTS	235	128	5,418	-36
10	FMB P PR_ENABLE_DISABLE_PROC	377	127	9,412	43
11	FMB P PR_CHECKING_EVENTS	230	124	5,222	-18
12	FMB P CGRISCHK_ACCOUNT_NUMBERS	335	124	1,043	-1
13	FMB T WHEN-BUTTON-PRESSED	263	118	3,504	-40
14	FMB F FN_DEFAULT_WHERE_OPT7A	446	117	1,487	27
15	FMB F FN_DEFAULT_WHERE_OPT7	446	117	1,487	27
16	FMB F FN_CALC_POINTS	228	116	5,798	50
17	FMB F FN_CALC_POINTS	232	116	5,842	-26
18	FMB P PR_REFERRALS	296	115	6,411	-11
19	FMB P PR_CHECK_CONTINUE_THIRDPART	172	114	3,149	-3
20	FMB P PR_VALIDATE_BNK	171	113	2,220	-32

Cyclomatic Complexity

- 1 - 10 A simple program, without much risk
- 11 - 20 A more complex program, with moderate risk
- 21 - 30 A complex, high risk program
- > 50 An un-testable program with very high risk

Source Code Metrics (Cyclomatic Complexity)

Un-testable program - very high risk (40%)
Very complex program - high risk (51%)

● Very complex program - high risk
● Un-testable program - very high risk

Show first 500 objects

Source Code Metrics
 Source Code Metrics(top objects)

Statements
 Cyclomatic Complexity 0 0 253 247
 Halstead Volume
 Maintainability Index

Location

DB FMB PLL OLB MMB RDF

Object Type

Package (object) Package (details)
 Procedure Function Trigger
 Menu Item Built-in

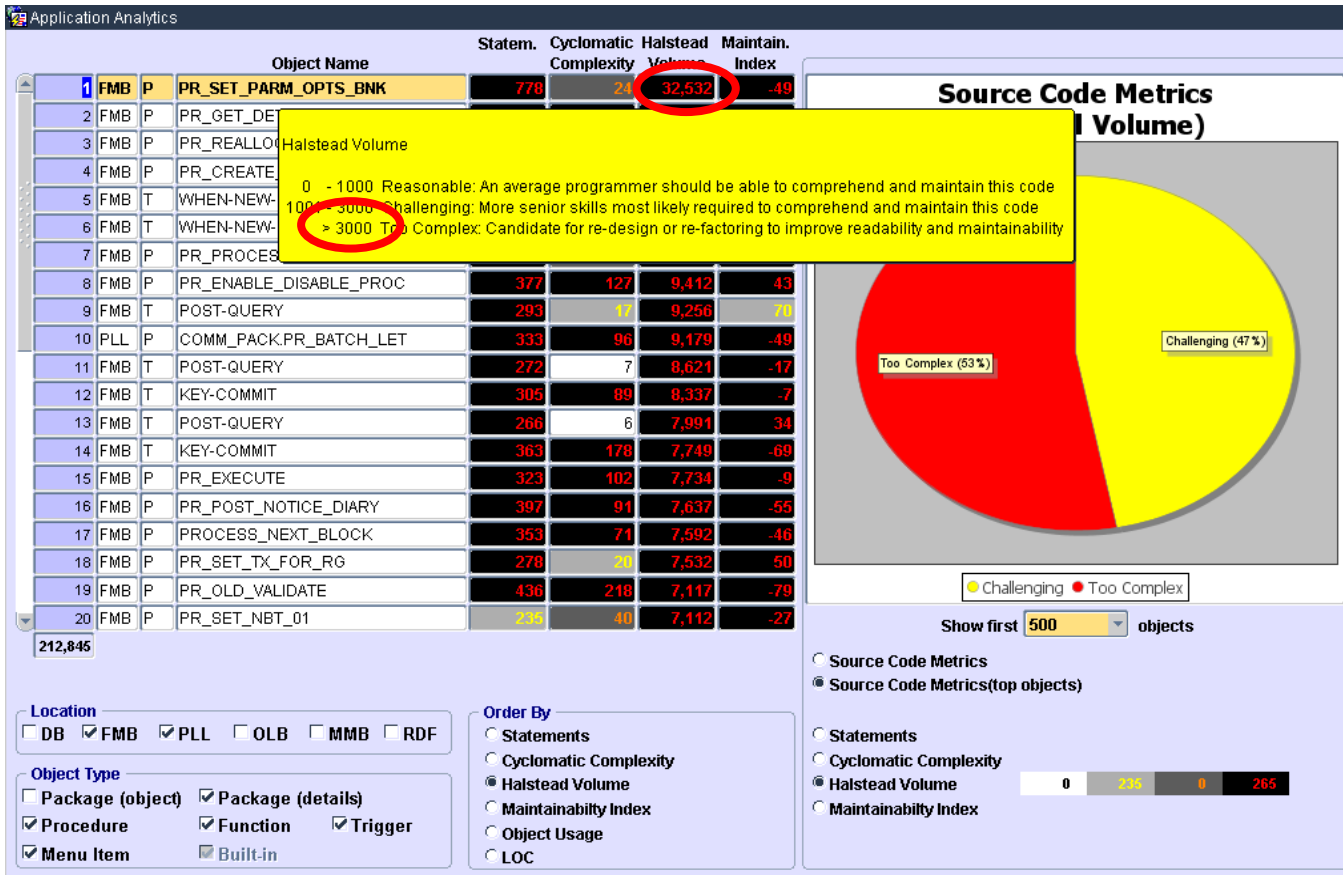
Order By

Statements
 Cyclomatic Complexity
 Halstead Volume
 Maintainability Index
 Object Usage
 LOC



WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- Halstead Volumetrics - How easy is the code to comprehend and maintain for developer





WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- Maintainability - How easy is the code to maintain?

Application Analytics

		Object Name	Statem.	Cyclomatic	Halstead	Maintain.	
			Complexity	Volume	Index		
1	FMB	P	PR_OLD_VALIDATE	436	218	7,117	-79
2	FMB	T	KEY-COMMIT	363	178		
3	FMB	P	PR_REALLOCATION_OF_PAYMENT	355	151		
4	FMB	T	WHEN-MOUSE-DOUBLECLICK	206	144		
5	FMB	P	PR_POST_NOTICE_DIARY	397	96		
6	FMB	P	PR_CHECKING_EVENTS	250	136		
7	FMB	P	PR_CHECKING_EVENTS	250	136		
8	PLL	P	COMM_PACK.PR_BATCH_LET	333	96	9,179	-49
9	FMB	P	PR_SET_PARM_OPTS_BNK	778	24	32,532	-49
10	FMB	P	PROCESS_NEXT_BLOCK	353	71	7,592	-46
11	FMB	T	WHEN-BUTTON-PRESSED	256	86	5,006	-44
12	FMB	P	PR_INSERT	231	57	2,339	-43
13	FMB	T	KEY-LISTVAL	125	104	3,407	-40
14	FMB	T	WHEN-BUTTON-PRESSED	263	118	3,504	-40
15	PLL	P	WRK0895K.WRK0120D	177	62	4,658	-40
16	FMB	T	KEY-LISTVAL	125	104	3,407	-40
17	FMB	T	POST-FORMS-COMMIT	172	78	4,142	-37
18	FMB	P	PR_FL_PRECOMMIT	173	78	2,100	-36
19	FMB	P	PR_CHECKING_EVENTS	235	128	5,418	-36
20	FMB	T	WHEN-VALIDATE-ITEM	194	53	4,057	-35

212,845

Location: DB FMB PLL OLB MMB RDF

Object Type: Package (object) Package (details) Procedure Function Trigger Menu Item Built-in

Order By: Statements Cyclomatic Complexity Halstead Volume Maintainability Index Object Usage LOC

Source Code Metrics (Maintainability)

0 - 64: Difficult to maintain
65 - 84: Moderate Maintainability
> 85: Highly Maintainable

Difficult to maintain (100%)

● Difficult to maintain

Show first 500 objects

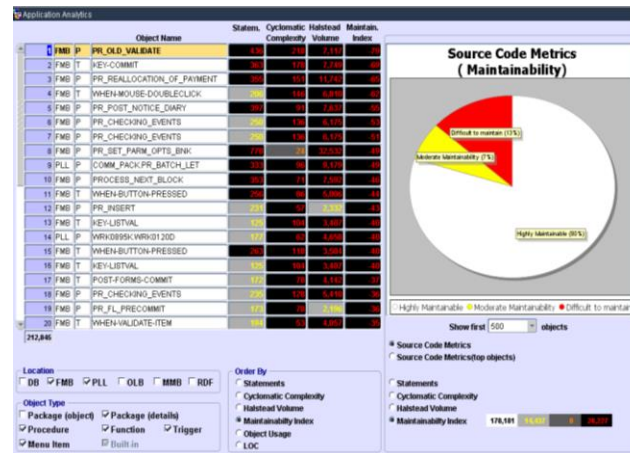
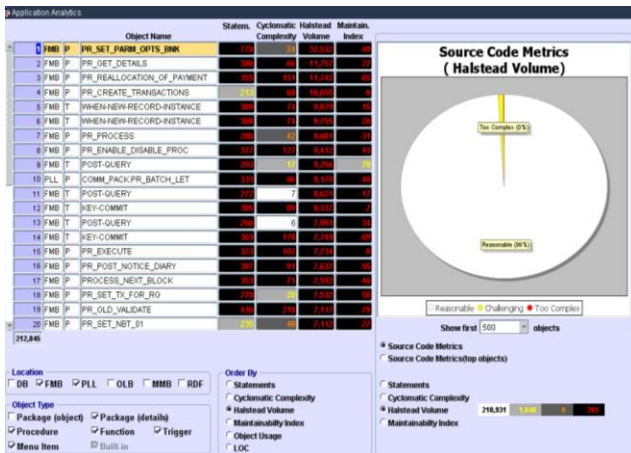
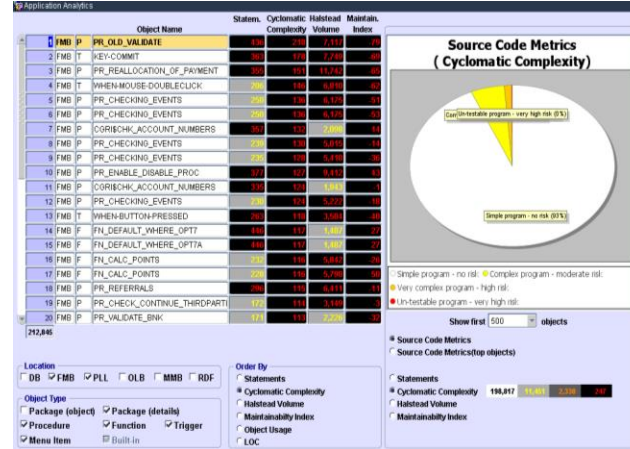
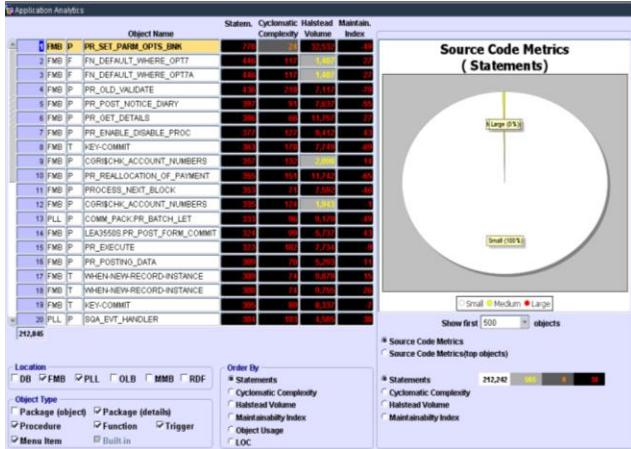
Source Code Metrics (top objects): Maintainability Index

Statements: 0
Cyclomatic Complexity: 0
Halstead Volume: 0
Maintainability Index: 500



WHAT HAVE WE LEARNED ABOUT OUR APPLICATION?

- So the application really is complex. Right?





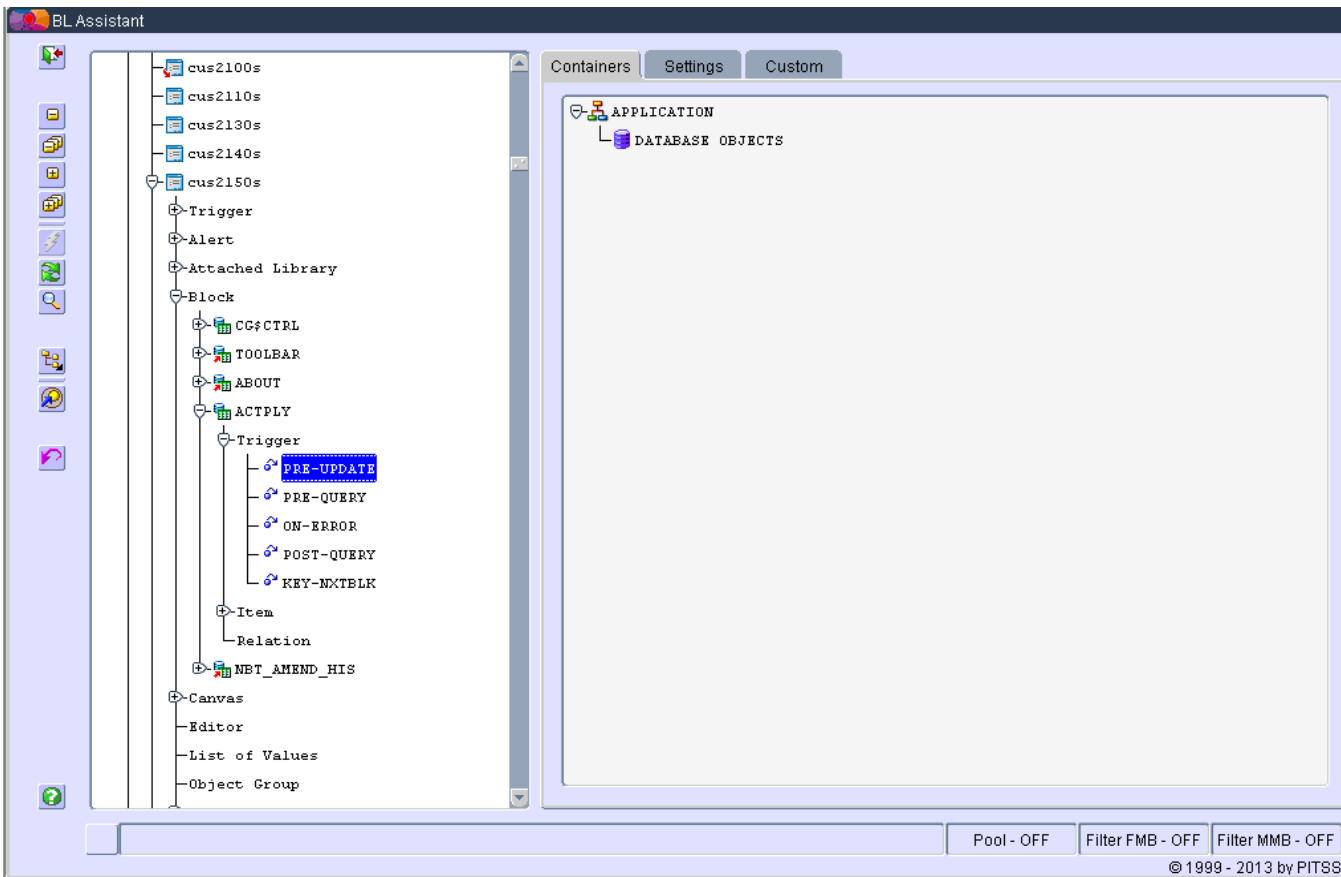
THE JOURNEY TO SOA

PITSS APPLICATION ENGINEERING



WHAT CAN WE START TO REUSE IN A SOA WORLD?

- We have logic within our existing Forms functionality which we may want to reuse rather than having to re-write within a new technology

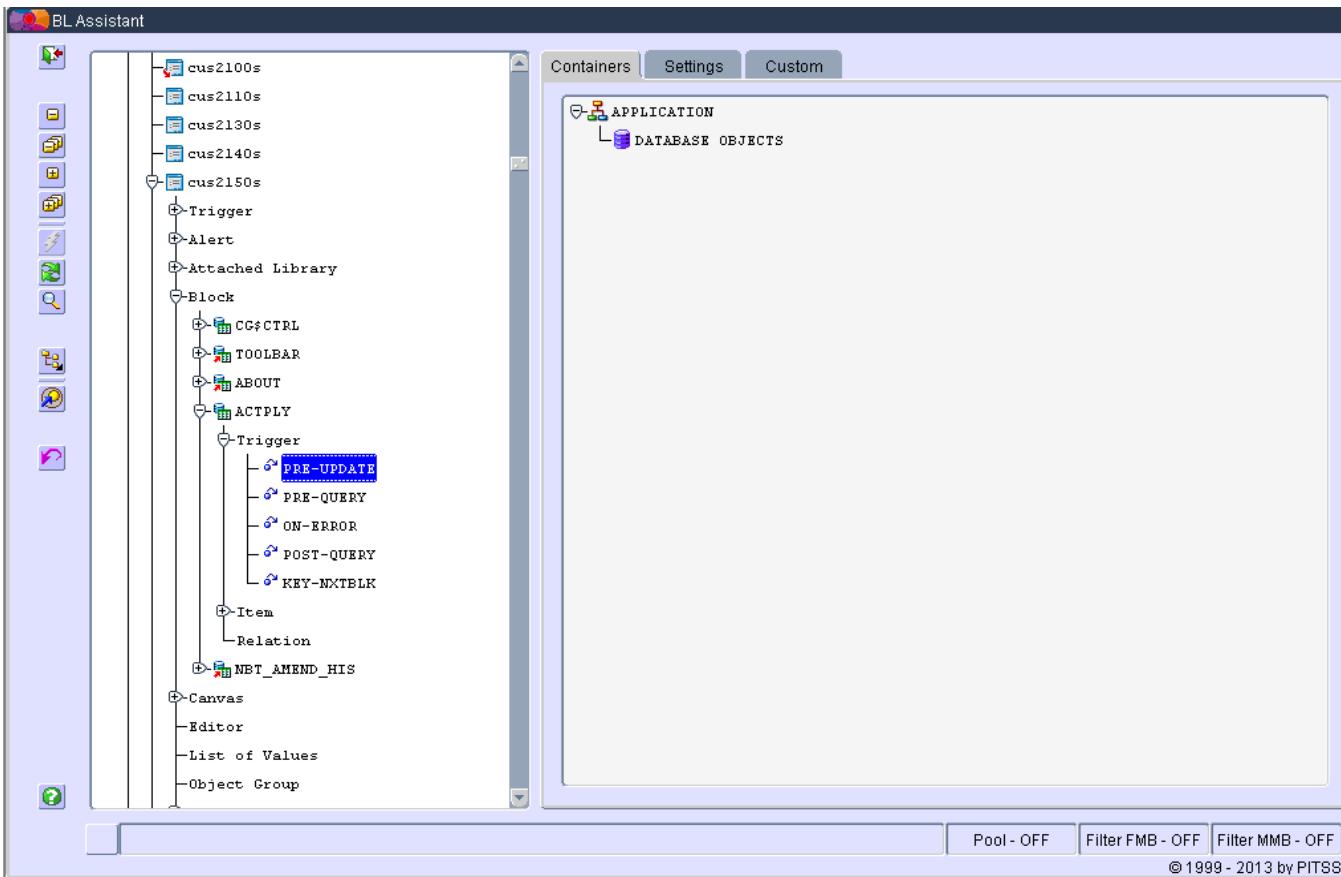


- Using the Application Engineering functionality within PITSS, where it is appropriate we can look to push this logic from Oracle Forms into the database



WHAT CAN WE START TO REUSE IN A SOA WORLD?

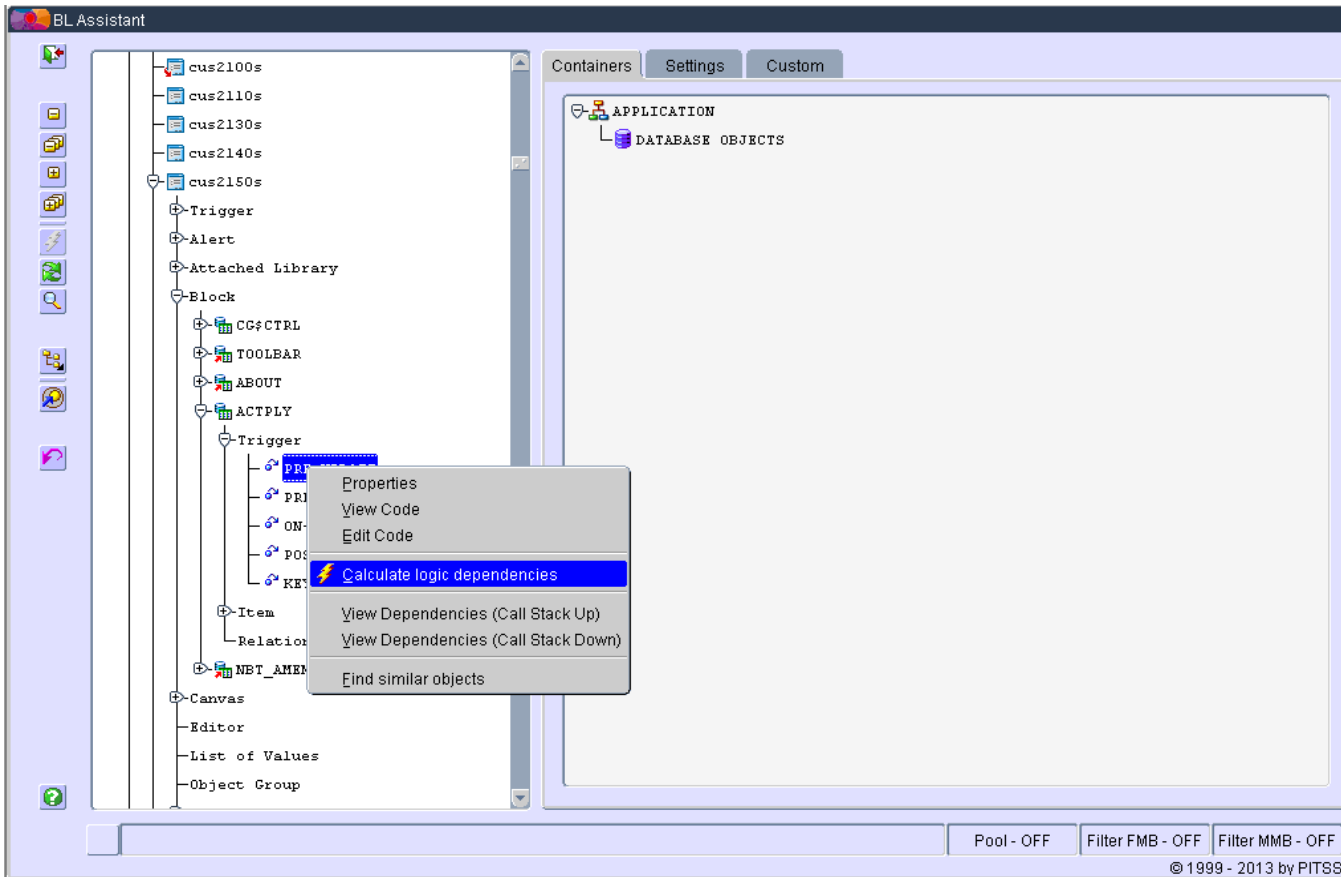
- The PRE-UPDATE trigger contains logic which allows the business user to link customer records and associated customer related data together





WHAT CAN WE START TO REUSE IN A SOA WORLD?

- PITSS calculates all the dependencies within the trigger





WHAT CAN WE START TO REUSE IN A SOA WORLD?

- And displays them for review...

The screenshot shows the 'EL Assistant' application window. On the left is a tree view of database objects, with 'PRE-UPDATE' selected. The main area displays a table of object statistics and a 'Details' panel for the selected object.

Object Name	Type	SQL	Call Intern	BI	Used	Activity
18 PR_UPDATE_CB_ACCOUNTS	PROCEDURE	1	1	0	2	2
19 PR_UPDATE_APP_PARTIES	PROCEDURE	1	1	0	2	1
20 PR_UPDATE_AFFORDABILITY	PROCEDURE	3	1	0	2	3
21 PR_SALES_APP_PARTIES	PROCEDURE	1	1	0	2	1
22 PR_SALES_EXIST_MORTGAGE	PROCEDURE	1	1	0	2	1
23 PR_SALES_MPI_DETAILS	PROCEDURE	2	1	0	2	2
24 PR_CHECK_PERSONIDS	PROCEDURE	6	3	0	1	21
25 PR_CHECK_ADDRIDS	PROCEDURE	8	3	0	1	23
26 PRE-UPDATE	TRIGGER	13	15	1	0	34

Transfer Objects Flow to DB

Details | **SQL Details**

New Object Name: T_PRE_UPDATE ✓

Type	Object Name (8)	Data Type	New Object name
GLOBAL	:GLOBAL.GS_PC_ID	IN VARCHAR2	G_GS_PC_ID
ITEM	:ACTPLY.DRV_SYSID	IN NUMBER	I_DRV_SYSID
ITEM	:ACTPLY.ENDED_AT	IN OUT VARCHAR2	I_ENDED_AT
ITEM	:ACTPLY.ENDED_BY	IN OUT VARCHAR2	I_ENDED_BY
ITEM	:ACTPLY.ENDED_DATE	IN OUT DATE	I_ENDED_DATE

Type	Location	Object name	Reference
PROCEDURE	DB	PR_ADDUSE_PARTY_DEDUPE	Keep
PROCEDURE	DB	PR_CHECK_PERMRK	Keep
BUILT_IN	BUILT_IN	MESSAGE	Remove
FUNCTION	MODULE	FN_CHECK_MAIN_HOLDER	Remove

Pool - OFF | Filter FMB - OFF | Filter MMB - OFF

© 1999 - 2013 by PITSS



WHAT CAN WE START TO REUSE IN A SOA WORLD?

- The objects can then be transferred to a database package

Transfer Objects Flow to DB

Create new container
 Add to an existing container

Pitsscon Owner
PITSS1

DB Owner
COREOWN

Container
CUS2150S_PKG

Calculate Flow

Object Name	Type	SQL	Call Intern	BI	Used	Activity
18 PR_SALES_APP_PARTIES	PROCEDURE	1	1	0	2	1
19 PR_SALES_MPI_DETAILS	PROCEDURE	2	1	0	2	2
20 PR_UPDATE_CARD_PIN_DETAILS	PROCEDURE	6	1	0	2	17
21 PR_UPDATE_AFFORDABILITY	PROCEDURE	3	1	0	2	3
22 PR_UPDATE_R85_TAX_EXEMPT_TABLI	PROCEDURE	3	1	0	2	5
23 PR_CHECK_PERSONIDS	PROCEDURE	6	3	0	1	21
24 FN_CHECK_MAIN_HOLDER	FUNCTION	1	1	0	2	4
25 PR_UPDATE_APP_PARTIES	PROCEDURE	1	1	0	2	1
		5	1	0	34	

Type	Location	Object name	Reference
PROCEDURE	DB	PR_ADDUSE_PARTY_DEDUPE	Keep
PROCEDURE	DB	PR_CHECK_PERMRK	Keep
BUILT_IN	BUILT_IN	MESSAGE	Remove
FUNCTION	MODULE	FN_CHECK_MAIN_HOLDER	Remove

© 1999 - 2013 by PITSS



WHAT CAN WE START TO REUSE IN A SOA WORLD?

- Internal logic calls now reference other objects transferred to the package

The screenshot shows the 'BL Assistant' tool with the following data:

Object Name	Type	SQL	Call Intern	BI	Used	Activity
18 PR_SALES_APP_PARTIES	PROCEDURE	1	0	0	2	1
19 PR_SALES_MPI_DETAILS	PROCEDURE	2	0	0	2	2
20 PR_UPDATE_CARD_PIN_DETAILS	PROCEDURE	6	0	0	2	17
21 PR_UPDATE_AFFORDABILITY	PROCEDURE	3	0	0	2	3
22 PR_UPDATE_R85_TAX_EXEMPT_TABL	PROCEDURE	3	0	0	2	5
23 PR_CHECK_PERSONIDS	PROCEDURE	6	0	0	1	21
24 FN_CHECK_MAIN_HOLDER	FUNCTION	1	0	0	2	4
25 PR_UPDATE_APP_PARTIES	PROCEDURE	1	0	0	2	1
26 PRE-UPDATE	TRIGGER	13	0	1	0	34

Transfer Objects Flow to DB

Details SQL Details

New Object Name: T_PRE_UPDATE

Type	Object Name (14)	Data Type	New Object name
GLOBAL	:GLOBAL.GS_PC_ID	IN VARCHAR2	G_GS_PC_ID
ITEM	:ACTPLY.DRV_SYSID	IN NUMBER	I_DRV_SYSID
ITEM	:ACTPLY.ENDED_AT	IN OUT VARCHAR2	I_ENDED_AT
ITEM	:ACTPLY.ENDED_BY	IN OUT VARCHAR2	I_ENDED_BY
ITEM	:ACTPLY.ENDED_DATE	IN OUT DATE	I_ENDED_DATE

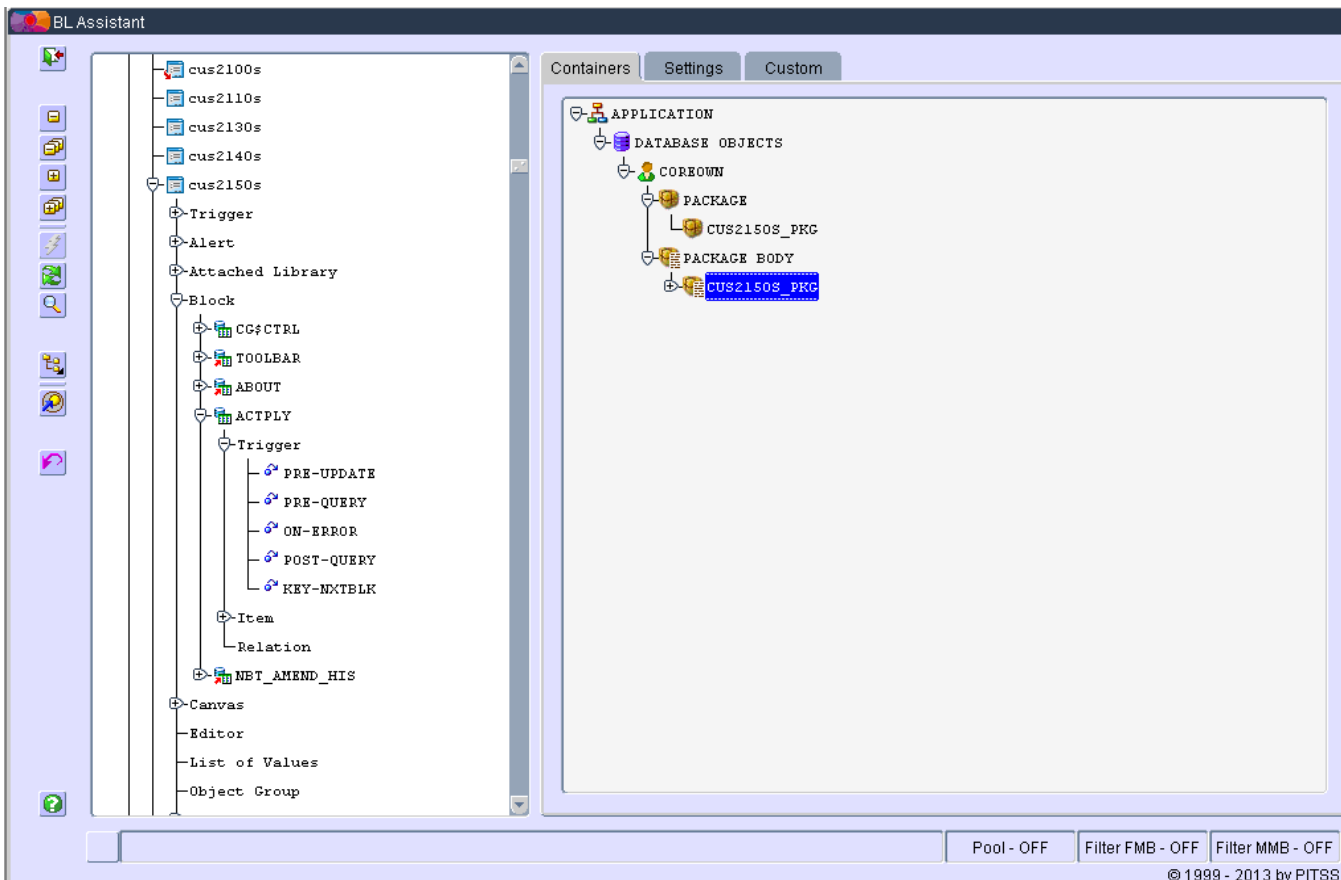
Type	Location	Object name	Reference
FUNCTION	MODULE	FN_CHECK_MAIN_HOLDER	Change CUS2150S_PKG.FN_CHECK_M
PROCEDURE	MODULE	MESSAGE_DISPLAY	Change CUS2150S_PKG.MESSAGE_DI
PROCEDURE	MODULE	PR_CHECK_ADDRIDS	Change CUS2150S_PKG.PR_CHECK /
PROCEDURE	MODULE	PR_CHECK_PEOPLE_MATCH	Change CUS2150S_PKG.PR_CHECK PE

Pool - OFF Filter FMB - OFF Filter MMB - OFF © 1999 - 2013 by PITSS



WHAT CAN WE START TO REUSE IN A SOA WORLD?

- The newly created package can be inspected and edited within PITSS before being applied to the database



- This example was estimated by development to have reduced the amount of developer effort required by between 60-75% over doing it manually



NEXT STEPS FOR YBS



NEXT STEPS

- Strategic Journey continues - currently mapped out to 2019
- SOA & BPM has initially proved to be perhaps more difficult than we originally envisaged
- Importance of our investment in Oracle Forms is back on the agenda (It is running our Business today!). We have an upgrade project in progress - although in reality, it was the constraints of IE8 which has pushed the upgrade to the fore
- Identify further opportunities to exploit using PITSS. We would like to take the opportunity as part of the Forms upgrade to remove some of the considerable quantities of 'dead code' which exists within our application today
 - Would demonstrate we are shrinking our 'legacy' code base as we increase deployment of newer technologies and code, but also
 - Make this legacy code 'easier and simple' to maintain in future - but....
 - Requires a change of mind-set, both Management and Developer - currently no time is allocated to improving or refactoring our code when checked out by projects
- Continue to push the benefits of PITSS to our development community and harden code refresh processes
- Look to exploit new features of PITSS - specifically:
 - Development effort in Source Code Analytics - more accurate estimating?
 - User Journey's - ability to track execution activities within the application which are not directly visible to our end-user

Any regrets?

- That we didn't have the tool 10 or more years ago!



QUESTIONS?

Thank you for listening