



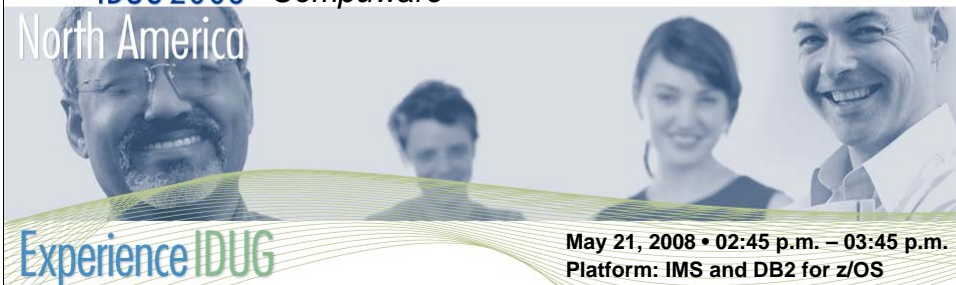
Session: J11

Understanding Your Applications: Now you can have your IMS and DB2

IDUG 2008

North America

Randy S. Crossley
Compuware



May 21, 2008 • 02:45 p.m. – 03:45 p.m.
Platform: IMS and DB2 for z/OS

Understanding critical mainframe applications can be difficult. When IMS and DB2 are introduced into the mix, it further complicates this issue. Determining application relationships to and from DB2 and IMS entities becomes particularly difficult. Ever try to trace a DB2 column back to the related program data items? DevEnterprise, along with the recent addition of being able to collect DB2 information, has now introduced improved IMS support. You will now see valuable information on your IMS databases, including PCB and PSB information. For online IMS applications the Transaction to Program relationship will be identified. The Impact analysis function of DevEnterprise can now assist greatly in identifying the application relationships to the DB2 and IMS entities.

Objectives:

- Identifying the inter-relationships of application components
- Producing impact analysis of database fields and columns
- Creating graphical views of the database and application entities
- Base line program complexity
- Identifying code flaws that may potentially affect application quality

Problem:

- Applications are becoming increasingly more difficult to understand and maintain.
 - In particular DB2 and IMS add another layer to the analysis phase
 - As data gets passed from field to field more scans of the application components are required to find the relationships
 - Changes to applications often increase complexity

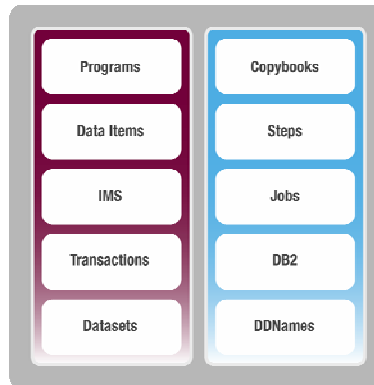
Are you making a simple code change like renaming or expanding a field or changing a DB2 table, and need to know the effect on the other components of your application? Are you building a new application and need to know what data is available? Or do you need to locate potential relationships between data to help with a data migration project? Application relationships are critical to any major maintenance project. Typically identifying these relationships is a tedious and manual effort. Often times relationships go un-identified causing production problems. As more changes take place in an application, complexity increases, making an already difficult task even more trying.

The reason:

- Aging applications
- Complex applications
- Outsourced applications
- Staff turnover / Retirement

There are several reasons contributing to this issue. The combination of aging applications and staff turnover is one major area. Often times those with the most knowledge leave before being able to fully transition that information over.

What makes up your application?



An application is made up of many entities. Is it an online application or batch? Where is the data located? Do I need JCL in addition to the source and copybooks?.

How do we gather this data?



- Scan source libraries.
- Scan JCL and PROCS.



- Review documentation.



- Work with team members to “share” the knowledge.

Who knows where this information is and how do I get to it? These are all questions you need to ask before you even get started with identifying the application relationships.

Then What? How do we . . .

- Identify relationships
- Identify impact analysis
- Understand data flow analysis
- Create graphical views

OK, now that we have the information, how do we do the following?

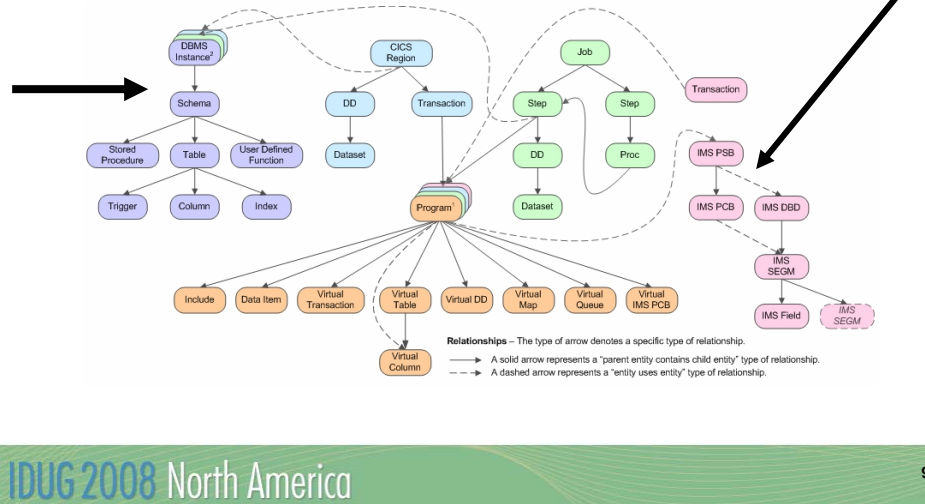
Impact analysis today

- Manual process
- Generally requires multiple scans through source for data item analysis
 - Difficult to find based on name or attribute.
- Copybooks and databases require manual scanning of additional libraries
- Difficult to cross-reference entities related to DB2 and IMS
- Program complexity increases amount of manual effort
- Difficult, if not impossible, to gain a graphical view

Impact analysis generally requires the scanning of source, copybooks, and databases. It can get very difficult when the name doesn't conform to the item. It might be simple to identify date fields, for example, but not all data names will be as easy to locate.

Because the information is scattered throughout various files (source, database, JCL) the effort to cross reference all information can become very time consuming.

The big picture:

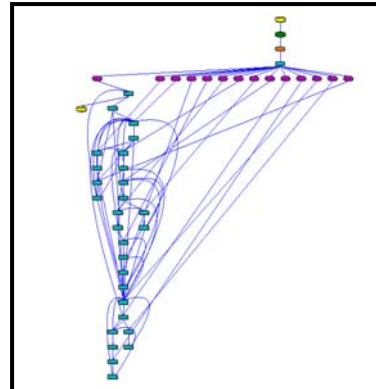
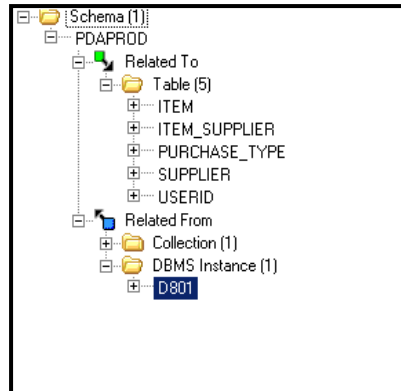


This diagram represents all the possible entities associated with an application. A solid area in this diagram represents a “parent entity contains a child entity” type of relationship. A dashed arrow represents a “entity uses entity” type of relationship.

With the Metadata Analyzer, you can collect metadata from all of your batch, CICS, DB2, COBOL, PL/I, IMS, and Assembler applications—automatically, behind the scenes.

Multiple users can simultaneously access the information they each need to determine the impact of changing includes, jobs, Procs, datasets, DB2 columns, fields, etc. . . even if they cross applications!

The solution: DevEnterprise



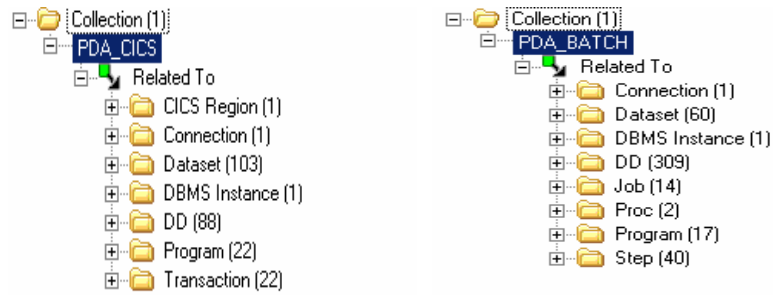
XPEDITER/DevEnterprise

- All data located in a single repository
- Data easily kept in sync with mainframe
- Simple search process for any entity in the repository
 - Search by partial name
 - Search by size
- Relationships can be discovered beginning with any level, not just “top down”
- Easy to perform and graph impact analysis
- Graphical views of application and programs

Administrators set up collections to gather metadata, and guest users simply search the repository for metadata that matches what they are looking for. Collections can be automated and setup to only collect data that has been updated since the last collection. This allows to easily keep the repository in sync with the data on the mainframe.

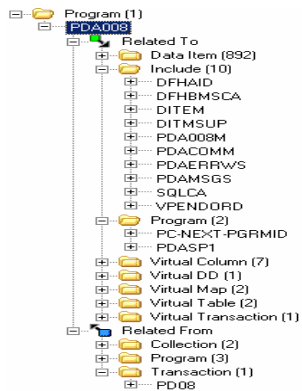
Search's can be done on partial names or even by the size of the data item.

DevEnterprise: Tree View - Application



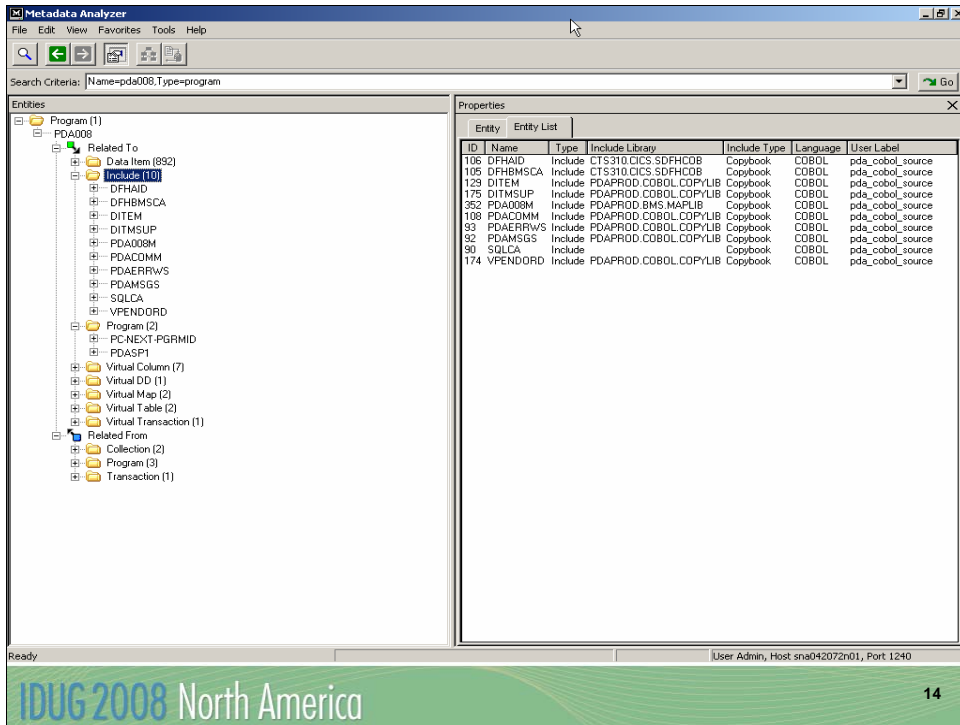
The tree view of an application lets you quickly see how many programs, copybooks, and datasets exist within the application. This information is produced from a “one click” search of the repository.

DevEnterprise: Tree View - Program



- All relationships displayed in the tree for the program
 - 892 data items
 - 10 copybooks
- Can easily see what copybooks program uses
 - What programs are called from the program
 - What transactions call the program

Expanding the tree lets us see what entities are directly related within the application. For example, I see how many data items and copybooks are related to the program I have expanded. Again, one click, and I now know exactly what copybooks are used by the program.



Selecting an item from the tree produces the “entity list” on the right. I not only what copybooks are used by the program, but I can also see the library the are being called from.

DB2 Collections:

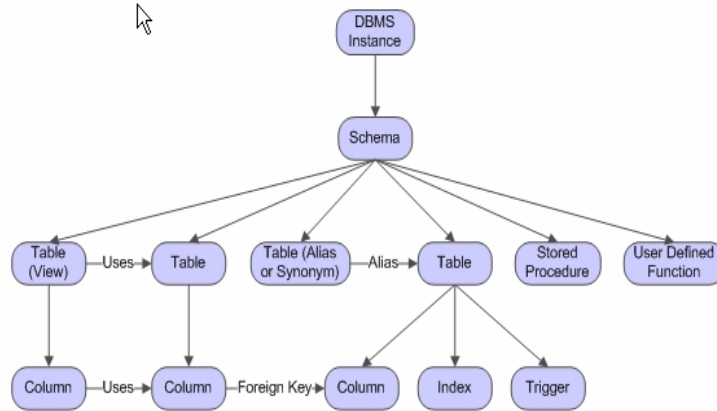
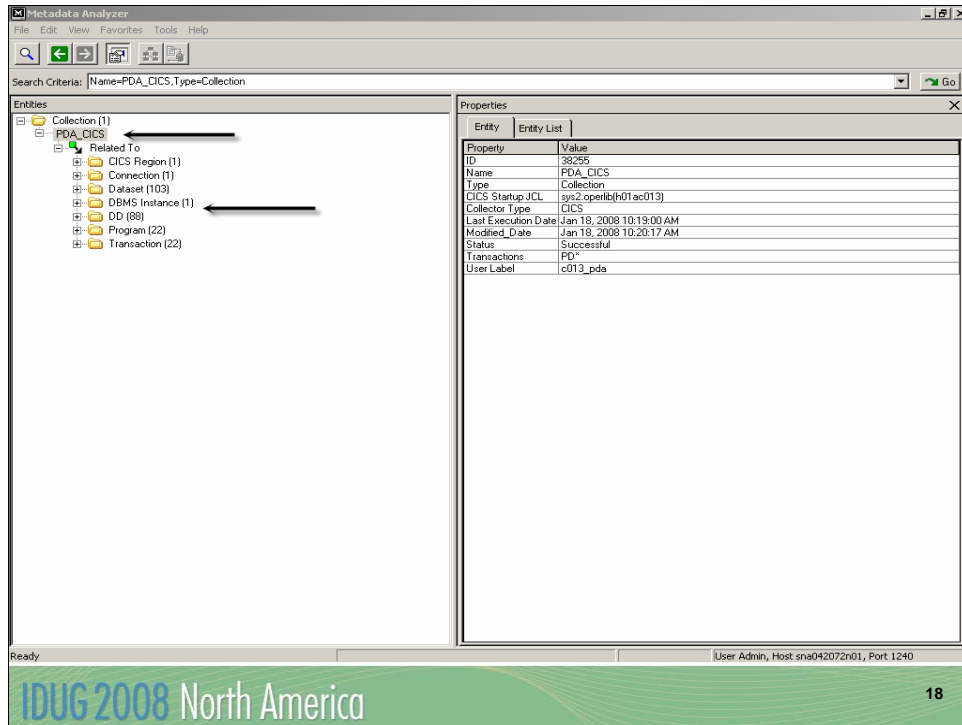
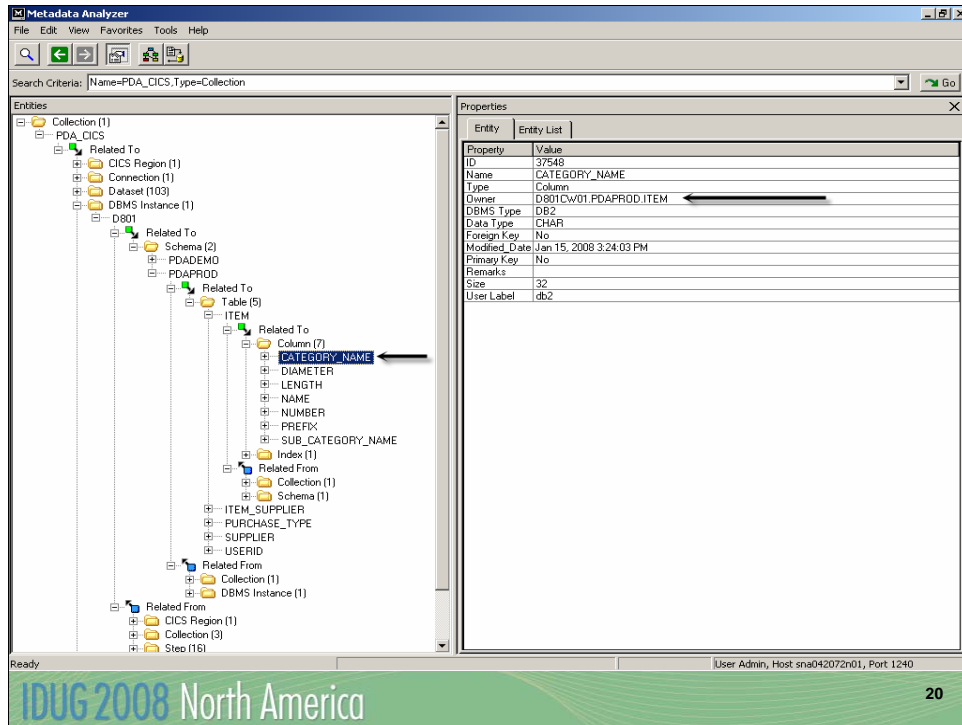


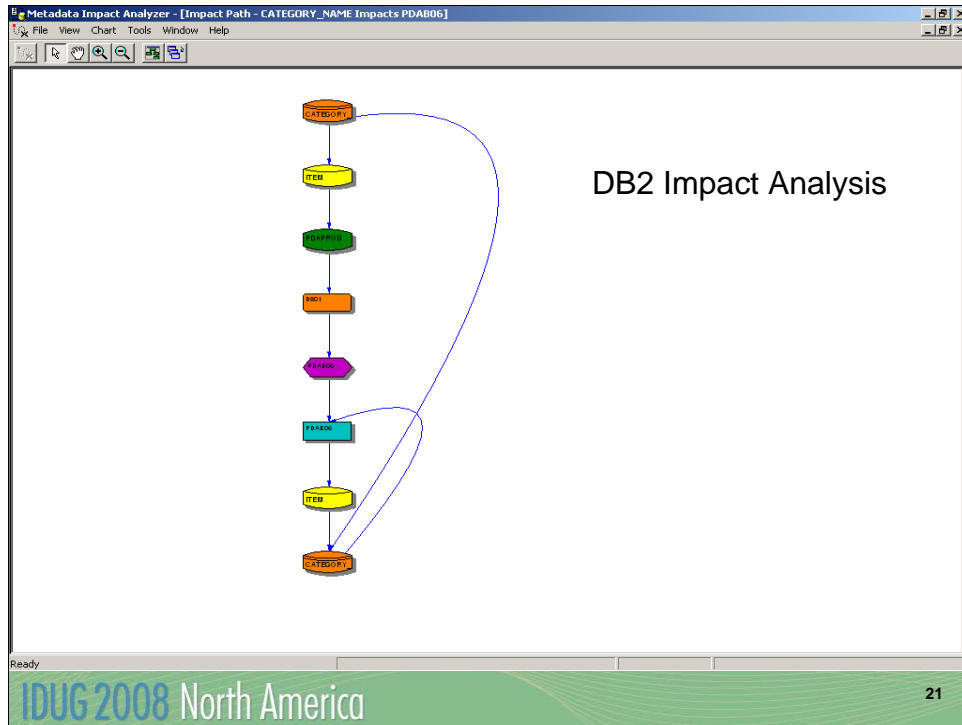
Diagram of DB2 relationships. The instance represents the DB2 subsystem to be collected.



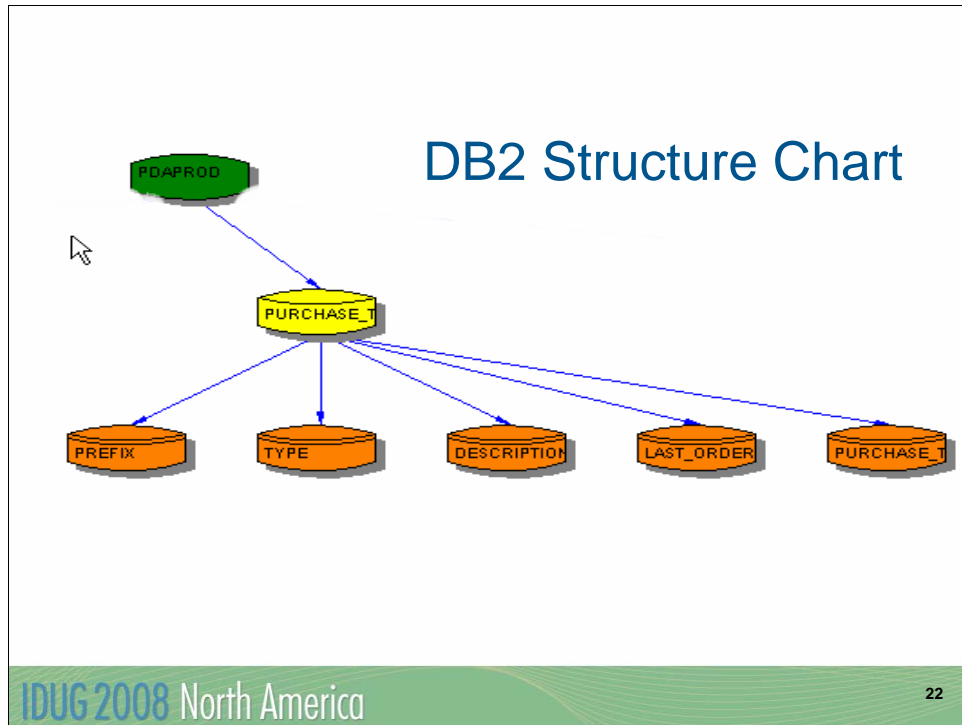
Another one click search on the DB2 collection produces the tree view of the DB2 entities.



Further expanding down to the COLUMN level. The entity list allows us to see owning or “parent” entity.



Taking it a step further. Producing impact analysis on the column lets us see a graphical flow from COLUMN to the PROGRAM.



Example of DB2 structure chart on a SCHEMA name showing TABLE and COLUMNS.

IMS Collections:

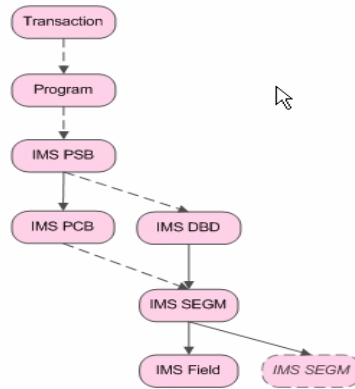
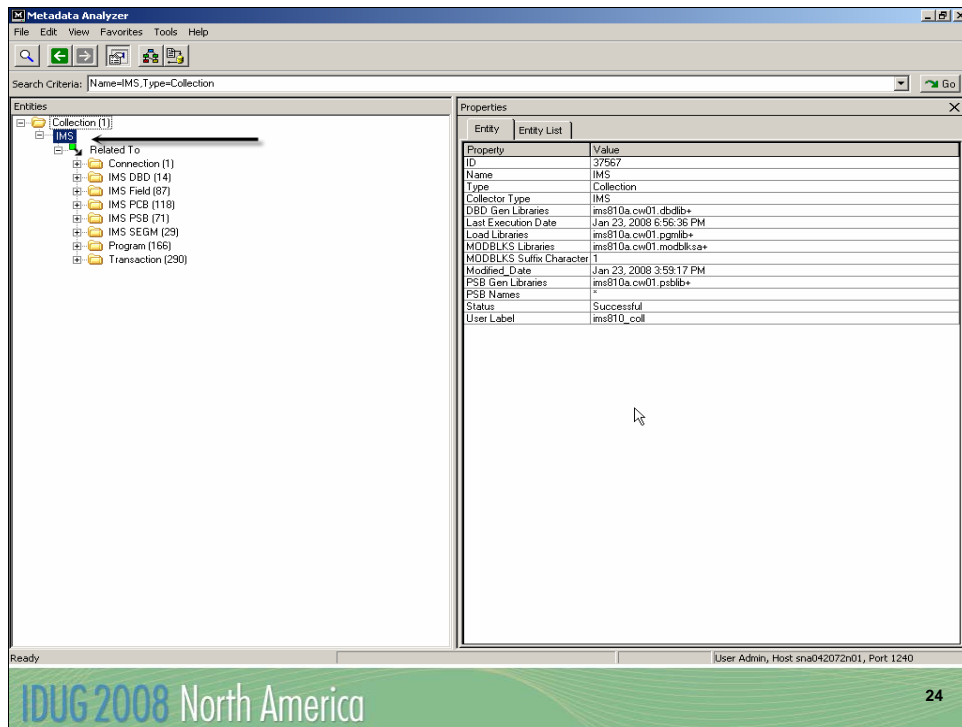


Diagram of IMS relationships.

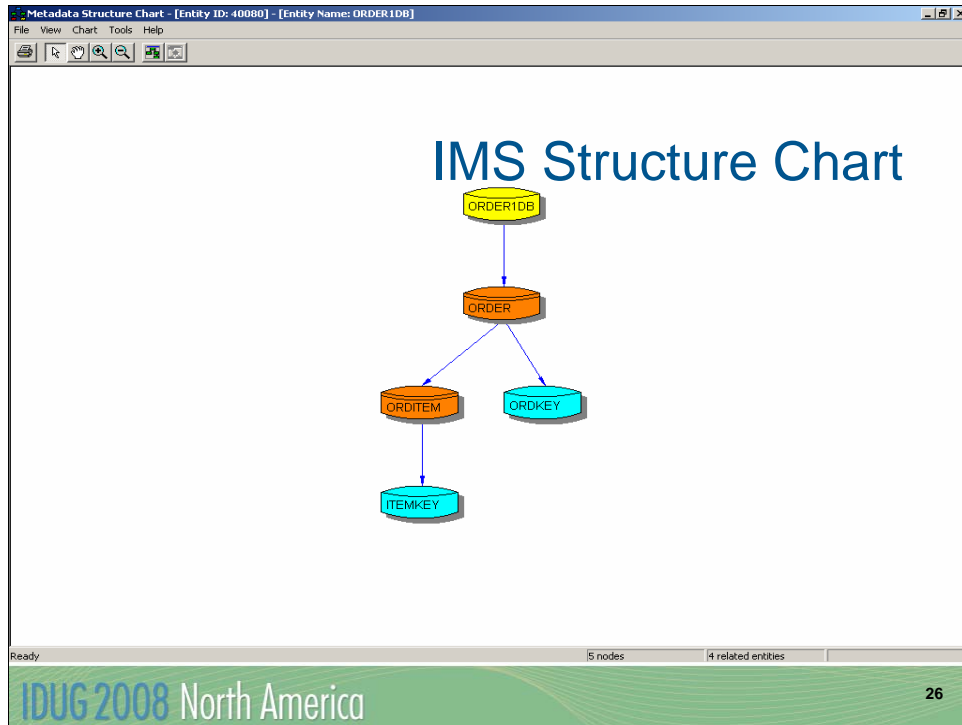


Another one click search on the IMS collection produces the tree view of the IMS entities.

The screenshot shows the Metadata Analyzer interface. On the left, the 'Entities' pane shows a tree view with 'Collection (1)' expanded to show 'IMS', which is further expanded to show 'Related To' entities including 'Connection (1)', 'IMS DBD (14)', 'IMS Field (87)', 'IMS PCB (118)', 'IMS PSB (71)', 'IMS SEGMENT (20)', 'Program (156)', and 'Transaction (290)'. The 'IMS SEGMENT (20)' entity is selected. On the right, the 'Properties' pane displays an 'Entity List' table with columns: ID, Name, Type, Length, Level, Modified_Date, and Owner. The table lists 20 entries, all of which are 'SEGMENT' entities owned by various DBDs.

ID	Name	Type	Length	Level	Modified_Date	Owner
40055	A1	IMS SEGMENT	10	1	Jan 23, 2008 3:59:04 PM	IVFDB11
39981	A1111111	IMS SEGMENT	40	1	Jan 23, 2008 3:59:03 PM	IVFDB1
39986	A1111111	IMS SEGMENT	40	1	Jan 23, 2008 3:59:03 PM	IVFDB2
39991	A1111111	IMS SEGMENT	42	1	Jan 23, 2008 3:59:03 PM	IVFDB3
40019	A1111111	IMS SEGMENT	40	1	Jan 23, 2008 3:59:03 PM	IVFDB4
39964	BACKDRDR	IMS SEGMENT	75	3	Jan 23, 2008 3:59:01 PM	DI21PART.PARTROOT.STOKST
39871	CHKSEGM	IMS SEGMENT	52	2	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT
39872	CHKTRSG	IMS SEGMENT	36	3	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT.CHCKS
39884	CURRSEGM	IMS SEGMENT	52	2	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT
39885	CURRTRSG	IMS SEGMENT	36	3	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT.CURRS
39851	CUSDEPSG	IMS SEGMENT	36	2	Jan 23, 2008 3:58:54 PM	DBFSAMD3.CUSTROOT
39897	CUSTADDR	IMS SEGMENT	50	2	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT
39850	CUSTROOT	IMS SEGMENT	49	1	Jan 23, 2008 3:58:54 PM	DBFSAMD3
39907	CUSTROOT	IMS SEGMENT	100	1	Jan 23, 2008 3:58:57 PM	DBFSAMD4
39962	CYCCOUNT	IMS SEGMENT	25	3	Jan 23, 2008 3:59:01 PM	DI21PART.PARTROOT.STOKST
39992	DSEGM2	IMS SEGMENT	42	2	Jan 23, 2008 3:59:03 PM	IVFDB3.A1111111
39928	ENLGR	IMS SEGMENT	40	1	Jan 23, 2008 3:58:57 PM	DBFSAMD1
39908	LOANSEGM	IMS SEGMENT	94	2	Jan 23, 2008 3:58:57 PM	DBFSAMD4.CUSTROOT
40081	ORDER	IMS SEGMENT	123	1	Jan 23, 2008 3:59:06 PM	ORDER1DB
40123	ORDER	IMS SEGMENT	123	1	Jan 23, 2008 3:59:08 PM	ORDER2DB
40082	ORDITEM	IMS SEGMENT	95	2	Jan 23, 2008 3:59:06 PM	ORDER1DB.ORDER
40124	ORDITEM	IMS SEGMENT	95	2	Jan 23, 2008 3:59:08 PM	ORDER2DB.ORDER
39959	PARTROOT	IMS SEGMENT	50	1	Jan 23, 2008 3:59:01 PM	DI21PART
40120	PENDORD	IMS SEGMENT	89	1	Jan 23, 2008 3:59:07 PM	PEND01DB
39858	SAVESEGM	IMS SEGMENT	52	2	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT
39859	SAVETRSG	IMS SEGMENT	36	3	Jan 23, 2008 3:58:55 PM	DBFSAMD3.CUSTROOT
39959	STANINFO	IMS SEGMENT	85	2	Jan 23, 2008 3:59:01 PM	DI21PART.PARTROOT
39961	STOKSTAT	IMS SEGMENT	160	2	Jan 23, 2008 3:59:01 PM	DI21PART.PARTROOT
39934	TELLSEG	IMS SEGMENT	60	1	Jan 23, 2008 3:58:57 PM	DBFSAMD2

Selecting the SEGMENT entity produces the entity list on the right. Easy to see each SEGMENT name on the owning IMS DBD.



Example of IMS structure chart.

Metrics – Tracking Complexity

- Changes to programs often increase complexity
- The more complex a program becomes, the more prone to code flaws and errors
- Tracking changes from outsourcers
 - Baseline metrics
 - Are changes increasing complexity
 - Dead code?

Advantages of tracking metrics.

DevEnterprise: Metrics - McCabe

- Base line current complexity of individual program
- Track changes to code for increased complexity
- If outsourced, are you getting optimal coding for you cost?
- Quality gates: Are they being met?

1	Program	Procedure Name	McCabe	GOTOs	Violations	Returns			
2	PDA001	P00050-INITIALIZE THRU P00050-INITIALIZE-EXIT	3	0	No	Maybe			
3	PDA001	P99500-PDA-ERROR THRU P99500-PDA-ERROR-EXIT	5	0	No	No			
4	PDA001	P00100-MAIN-PROCESS THRU P00100-MAIN-PROCESS-EXIT	3	0	No	Maybe			
5	PDA001	P00500-CHK-TRANS-INTENT THRU P00500-CHK-TRANS-INTENT-EXIT	4	1	No	Yes			
6	PDA001	P01000-MENU-PROCESS THRU P01000-MENU-PROCESS-EXIT	1	0	No	Yes			
7	PDA001	P80000-SEND-FULL-MAP THRU P80000-SEND-FULL-MAP-EXIT	3	0	No	Yes			
8	PDA001	P03000-EDIT-PROCESS THRU P03000-EDIT-PROCESS-EXIT	3	1	No	Maybe			
9	PDA001	P80200-RECEIVE-MAP THRU P80200-RECEIVE-MAP-EXIT	4	0	No	Yes			
10	PDA001	P03100-EDIT-SCREEN THRU P03100-EDIT-SCREEN-EXIT	8	3	No	Maybe			
11	PDA001	P03200-EDIT-PFKEY THRU P03200-EDIT-PFKEY-EXIT	12	2	No	Maybe			
12	PDA001	P80400-SEND-MESSAGE THRU P80400-SEND-MESSAGE-EXIT	3	0	No	No			
13	PDA001	P70000-ERROR-ROUTINE THRU P70000-ERROR-ROUTINE-EXIT	3	0	No	Yes			
14	PDA001	P03300-EDIT-SELECTION THRU P03300-EDIT-SELECTION-EXIT	3	1	No	Yes			
15	PDA001	P04000-VERIFY-USERID THRU P04000-VERIFY-USERID-EXIT	7	0	No	Yes			
16	PDA001	P04100-UPDATE-USERID THRU P04100-UPDATE-USERID-EXIT	3	0	No	Yes			
17	PDA001	P04200-ADD-USERID THRU P04200-ADD-USERID-EXIT	11	1	No	Yes			
18	PDA001	P04300-LOAD-USERID-DATA THRU P04300-LOAD-USERID-DATA-EXIT	3	0	No	Yes			
19	PDA001	P80100-SEND-MAP-DATAONLY THRU P80100-SEND-MAP-DATAONLY-EXIT	3	0	No	Yes			
20	PDA001	P80300-XFER-CONTROL THRU P80300-XFER-CONTROL-EXIT	1	0	No	Maybe			
21	PDA001	P00200-CICS-RETURN THRU P00200-CICS-RETURN-EXIT	1	0	No	Maybe			
22	PDA001	ENTRY	1	0	No	Yes			
23	PDA002	P80400-SEND-MESSAGE THRU P80400-SEND-MESSAGE-EXIT	5	0	No	No			
24	PDA002	P99500-PDA-ERROR THRU P99500-PDA-ERROR-EXIT	5	0	No	No			
25	PDA002	P00050-INITIALIZE THRU P00050-INITIALIZE-EXIT	7	0	No	Yes			
26	PDA002	P00100-MAIN-PROCESS THRU P00100-MAIN-PROCESS-EXIT	3	0	No	Maybe			
27	PDA002	P00500-CHK-TRANS-INTENT THRU P00500-CHK-TRANS-INTENT-EXIT	3	0	No	Yes			
28	PDA002	P01000-MENU-PROCESS THRU P01000-MENU-PROCESS-EXIT	3	0	No	Yes			
29	PDA002	P80000-SEND-FULL-MAP THRU P80000-SEND-FULL-MAP-EXIT	3	0	No	Yes			
30	PDA002	P03000-EDIT-PROCESS THRU P03000-EDIT-PROCESS-EXIT	3	0	No	Maybe			
31	PDA002	P80200-RECEIVE-MAP THRU P80200-RECEIVE-MAP-EXIT	4	0	No	Yes			
32	PDA002	P03100-EDIT-SCREEN THRU P03100-EDIT-SCREEN-EXIT	7	2	No	Maybe			
33	PDA002	P03200-EDIT-PFKEY THRU P03200-EDIT-PFKEY-EXIT	9	2	No	Maybe			
34	PDA002	P70000-ERROR-ROUTINE THRU P70000-ERROR-ROUTINE-EXIT	3	0	No	Yes			
35	PDA002	P80300-XFER-CONTROL THRU P80300-XFER-CONTROL-EXIT	1	0	No	Maybe			

Code Flaws: Potential “gotcha”

- Identifying potential code flaws to help prevent future issues
- What may seem “ok” today, could be a problem farther down the road.
 - Perform range violations
 - Dead Code
 - Unexecutable Code

	B	F	G	H	I
733	PDAB17	I006	Unentered procedure: P99999-ERROR		
734	PDAB17	I006	Unentered procedure: P99999-ERROR-EXIT		
735	PDAS02	I006	Unentered procedure: P00000-MAINLINE-EXIT		
736	PDASP1	I006	Unentered procedure: 9999-SQLERROR		
737	PDA050	I008	PRV : violates range LISTENER-STARTED-TASK THRU LISTENER-STARTED-TASK-EXIT: LISTENER-STARTED-TASK		
738	PDA050	I008	PRV : violates range LISTENER-STARTED-TASK THRU LISTENER-STARTED-TASK-EXIT: LISTENER-STARTED-TASK		
739	PDA050	I008	PRV : violates range LISTENER-STARTED-TASK THRU LISTENER-STARTED-TASK-EXIT: LISTENER-STARTED-TASK		
740	PDA050	I008	PRV : violates range INIT-SOCKET THRU INIT-SOCKET-EXIT: INIT-SOCKET		
741	PDA050	I008	PRV : violates range SCKET-BIND-LSTN THRU SCKET-BIND-LSTN-EXIT: SCKET-BIND-LSTN		
742	PDA050	I008	PRV : violates range SCKET-BIND-LSTN THRU SCKET-BIND-LSTN-EXIT: SCKET-BIND-LSTN		
743	PDA050	I008	PRV : violates range SCKET-BIND-LSTN THRU SCKET-BIND-LSTN-EXIT: SCKET-BIND-LSTN		
744	PDA050	I008	PRV : violates range ACCEPT-CLIENT-REQ THRU ACCEPT-CLIENT-REQ-EXIT: ACCEPT-CLIENT-REQ		
745	PDA050	I008	PRV : violates range ACCEPT-CLIENT-REQ THRU ACCEPT-CLIENT-REQ-EXIT: ACCEPT-CLIENT-REQ		
746	PDA050	I008	PRV : violates range GET-NAME-INFO THRU GET-NAME-INFO-EXIT: GET-NAME-INFO		
747	PDA051	I008	PRV : violates range TAKESOCKET-SEC THRU TAKESOCKET-SEC-EXIT: TAKESOCKET-SEC		
748	PDA051	I008	PRV : violates range TAKESOCKET-SEC THRU TAKESOCKET-SEC-EXIT: TAKESOCKET-SEC		
749	PDA051	I008	PRV : violates range GET-PEER-NAME THRU GET-PEER-NAME-EXIT: GET-PEER-NAME		
750	PDA051	I008	PRV : violates range GET-NAME-INFO THRU GET-NAME-INFO-EXIT: GET-NAME-INFO		
751	PDA051	I008	PRV : violates range CLIENT-TASK THRU CLIENT-TASK-EXIT: CLIENT-TASK		
752	PDA051	I008	PRV : violates range CLIENT-TALK-END THRU CLIENT-TALK-END-EXIT: CLIENT-TALK-END		
753	PDA051	I008	PRV : violates range CLIENT-TASK THRU CLIENT-TASK-EXIT: CLIENT-TASK		
754	PDA003	I010	SRV : violates scoped IF: P03100-EDIT-SCREEN		
755	PDA007	I010	SRV : violates scoped IF: P05200-SCROLL-FORWARD		
756	PDA007	I010	SRV : violates scoped IF: P05200-SCROLL-FORWARD		
757	PDA007	I010	SRV : violates scoped IF: P05220-FORMAT-FORWARD-LINE		
758	PDA007	I010	SRV : violates scoped IF: P05220-FORMAT-FORWARD-LINE		
759	PDA007	I010	SRV : violates scoped IF: P03120-EDIT-PFKEY		
760	PDA007	I010	SRV : violates scoped IF: P03120-EDIT-PFKEY		
761	PDA007	I010	SRV : violates scoped IF: P06200-SCROLL-BACKWARD		
762	PDA007	I010	SRV : violates scoped IF: P06200-SCROLL-BACKWARD		
763	PDA007	I010	SRV : violates scoped IF: P06220-FORMAT-BACKWARD-LINE		
764	PDA007	I010	SRV : violates scoped IF: P06220-FORMAT-BACKWARD-LINE		
765	PDA007	I010	SRV : violates scoped IF: P03100-EDIT-SCREEN		
766	PDA007	I010	SRV : violates scoped IF: P03100-EDIT-SCREEN		
767	PDA007	I010	SRV : violates scoped IF: P03130-EDIT-QUANTITY		
768	PDA007	I010	SRV : violates scoped IF: P03130-EDIT-QUANTITY		

Case Study

- Major Financial Institute
 - Billing application sporadically failing in production
 - Many developers spent several months trying to correct
 - DevEnterprise diagnostics report created
 - Recursive performs identified and corrected
- Problem Solved!

Would like to close with a case study regarding the value of metrics.

Questions???

Productivity Gains

- Reduced manual effort
- Repository dynamically synchronized with mainframe
- Relationships are maintained in repository
- Programmers work independently
- Reduced cost / mips
- Hard dollar savings

Session J11



Understanding Your Applications: Now you can have your IMS and DB2

Randy Crossley

Compuware

Randy.Crossley@compuware.com