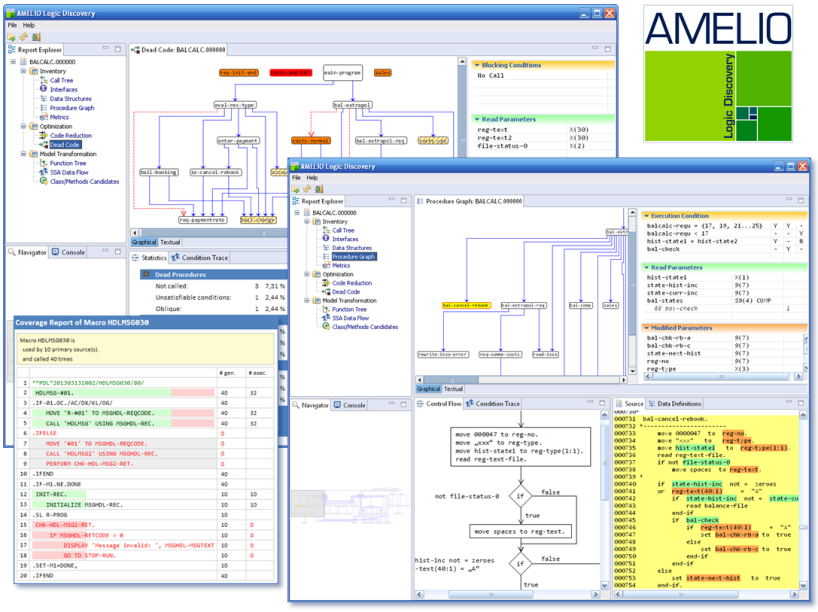
DELTA software technology

AMELIO LOGIC DISCOVERY

Spring-Clean Your COBOL- and PL/I-Applications

Spring is coming and the sun brings light into flats and houses. Dust and dirt being collected unrecognized during the dark season now gets visible. Thus in many households the spring-cleaning is started. Why don't you extend your spring-cleaning also to your COBOL- and PL/I-applications? Such applications were developed over many years or even decades. Within this time they were alive – they got maintained, extended, adapted to new requirements and therefore they grew. At many places dispensable ballast has been collected, program structures got changed, documentation is not available any more or not up to date. Maintaining and comprehending the application gets more and more complex.



AMELIO Logic Discovery-CleanUp

AMELIO Logic Discovery extracts the implemented application logic from COBOL and PL/I applications and thereby helps to comprehend them. The discovery of the application logic is performed in three steps – inventory, code optimization and logic analysis. The inventory determines which elements – i.e. programs, interfaces and data structures as well as dependencies between them – build the application. Elements relevant to be maintained or needed to be comprehend get determined during the optimisa-

tion phase, e.g. dead code is detected, documented if applicable and removed. The information provided by the preceding steps is used by the logic analysis to gain models which are independent from programming languages as well as from programming paradigms. Thereby the application logic is extracted and comprehensible represented.

We combined the inventory and the dead code analysis and reduction in the CleanUp package of AMELIO Logic Discovery. This package is intended to re-document existing applications and particularly to recognize, document and remove dispensable ballast and thus to efficiently, dependably and automatically perform the spring-cleaning.

Inventory – More Than an Appraisal

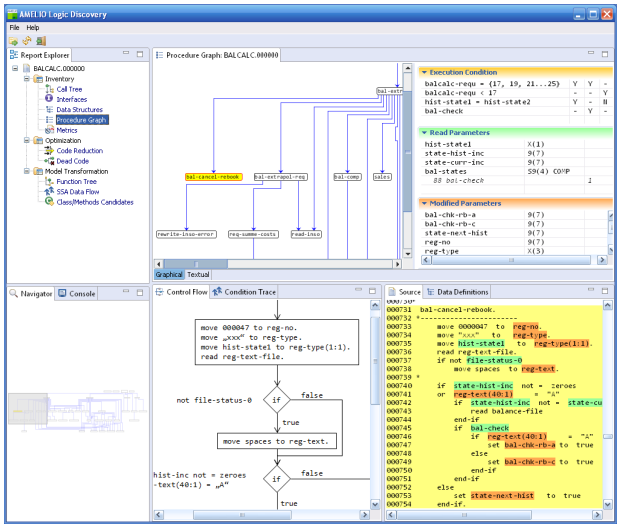
The first step of the spring-cleaning as well as of the comprehension of an application is to determine what has been developed in the past time. Therefore the inventory discovers, for example, which programs build the application and which interfaces, data structures and dependencies do exist in these programs. Instead of a pure appraisal AMELIO Logic Discovery already provides analyses:

- Conventional metrics, such as Halstead or McCabe
- Listing of existing data definitions
- Listing of all data base and file accesses
- Listing of all interface definitions and possible conflicts
- Representation of program and sub-program calls
- Representation of procedures, their call hierarchy, conditions and interfaces.

The results of the various analyses are displayed in AMELIO Logic Discovery's analysis workbench. A special highlight of this inventory is shown below, as an example for an analysis and the result representation: It represents procedures together with their call hierarchy, the interfaces, used parameters and particularly the conditions defining when the corresponding procedure will be performed. Due to the fact that there are no explicit procedures given in COBOL each program of the application is analysed. On the basis of the discovered call structures several parts of a program get combined and thereby build the procedures. In PL/I there are explicit procedures. These can be used as a basis for further analysis however many procedures are hidden by entry-variables and generics and have to be identified first. The discovered procedures and their call hierarchy are graphically represented. The result of the analysis is exemplarily shown in the following picture:

- Procedure graph: Displays the discovered procedures and their call hierarchy as well as calls of sub-programs and "wild" branches produced by GOTO.
- Conditions: During the inventory a condition analysis is already performed. This analysis discovers which conditions have to be fulfilled so that a procedure can actually be performed. The result is displayed as condition table.
- Data structures (COBOL): For each procedure it is displayed which data structures are read respectively modified by the procedure.
- Data structures (PL/I): For each procedure it is displayed which global data structures are read respectively modified by the procedure.
- Interfaces: For PL/I application the interface and, if existing, the return value for each procedure is additionally represented
- Control flow: For each procedure its control flow is displayed.

- Code parts: Those code parts which build a certain procedure are shown. Within these parts read respectively modified data structures are marked by colours.



Dead Code – Eliminate Dispensable Ballast

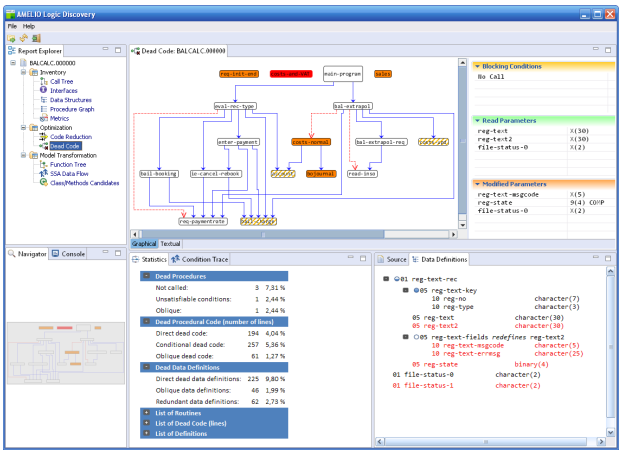
The longer an application exists the more often it got extended and adapted. Therefore it can be assumed that by age the amount of dead code within the application is increasing. I.e. code that exists but will never be performed. This code has to be maintained, complicates the comprehension of the application logic and requires storage. Besides the inventory the detection and elimination of dead code is therefore a central element of the spring-cleaning.

The dead code analysis is performed per program. It consists of the following elements which are based on each other:

- Direct dead code: Procedural code which is never performed as it is never called
- Conditional dead code: Procedural code which is only performed under certain conditions (e.g. within an if-condition) but these conditions can never be fulfilled
- Oblique dead code: Procedural code which is only performed by dead code
- Direct dead data definitions: Data definitions which are never used
- Oblique dead data definitions: data definitions which are used only in dead code
- Redundant data definitions: Literally these are no dead data definitions, instead they got produced by copy and paste. Anyway, they can be detected during the dead code analysis.

The result of the dead code analysis is shown in the following picture:

- Reduced procedure graph: This graph contains all dead procedures (coloured in orange, the selected procedure is coloured red) and all procedures which contain dead code (shaded orange). If a procedure is never called because the according condition cannot be fulfilled the call is marked orange. In addition all procedures are displayed which are above a dead procedure within the call hierarchy or which build alternative paths to a procedure called by a dead procedure.



- Blocking condition: Condition blocking the execution of the procedure.
- Data definitions: Represents the data definitions. Dead data definitions are marked red.

- Statistics: Dead code statistics for the entire program. It notes how many dead routines, lines of dead procedural code and data definitions were detected. Additionally all dead elements are listed.

Extended Dead Code-Analysis

But the dead code analysis of AMELIO Logic Discovery takes one step further. Developing COBOL as well as PL/I applications copy books respectively includes or - in case of ADS generated applications - macros are used. In these cases it is not sufficient to analyse single programs as the dead code can result from the use of copy books, includes or macros. Additionally they themselves can contain code which is never compiled respectively generated. This code is also detected by AMELIO Logic Discovery. To allow the analysis of copy books and includes AMELIO Logic Discovery emulates the appropriate compile function. The result of this coverage analysis contains the following information:

Coverage Report of Macro HDLMSG030

Macro HDLMSG030 is
used by 10 primary source(s).
and called 40 times

		# gen.	# exec.
1	**PDL*201303131002/HDLMSG030/80/		
2	HDLMSG-#01.	40	32
3	.IF-01.OC./AC/DX/KL/OG/	40	
4	MOVE 'R-#01' TO MSGHDL-REQCODE.	40	32
5	CALL 'HDLMSG' USING MSGHDL-REC.	40	32
6	.IFELSE	0	
7	MOVE '#01' TO MSGHDL-REQCODE.	0	
8	CALL 'HDLMSG2' USING MSGHDL-REC.	0	
9	PERFORM CHK-HDL-MSG2-RET.	0	
10	.IFEND	40	
11	.IF-M1.NE.DONE	40	
12	INIT-REC.	10	10
13	INITIALIZE MSGHDL-REC.	10	10
14	.SL R-PROG	10	
15	CHK-HDL-MSG2-RET.	10	0
16	IF MSGHDL-RETCODE > 0	10	0
17	DISPLAY 'Message invalid: ', MSGHDL-MSGTEXT	10	0
18	GO TO STOP-RUN.	10	0
19	.SET-M1=DONE,	10	
20	.IFEND	40	

- How many primary sources use a certain copy book, include or macro

- How often a certain copy book, include or macro is called during compilation respectively generation

- How often certain parts of a copy book or include are used during the compilation respectively how often certain parts of a macro are used during generation

- How often the created code is actually executable

Code Clean-Up

Now that the dead code, in all its facets, is detected and documented it can be removed. If the clean-up is performed manually there is the risk that not all lines or too many lines are removed and thereby the program logic is modified. Thus it is safer and faster to use the transformation function of AMELIO. It removes the entire dead code, procedural as well as data definitions.

Conclusion

COBOL and PL/I application grew over several years or even decades. In addition to the actually necessary functionality also dispensable ballast, complicating the maintenance and the comprehension of the application, has been accumulated during this time. With its inventory functions AMELIO Logic Discovery helps to gain an overview over the parts which define the application and which relations exist. The dead code analysis detects dead code, procedural as well as data definitions, and can also remove it automatically. This results in meaningful documentations as well as in cleaned up applications which are easier to maintain and to comprehend. Thus AMELIO Logic Discovery CleanUp supports maintenance and quality assurance.

Start spring-cleaning your COBOL and PL/I applications now. The CleanUp package of AMELIO Logic Discovery lights also the most hidden corners of application. It shows the dust and can remove it - fast, reliably and efficiently.

Predicate BEST OF 2015 for AMELIO Logic Discovery

The "Initiative Mittelstand" (Initiative for Small and Medium-sized Businesses) awarded AMELIO Logic Discovery with the predicate BEST OF 2015.

AMELIO Logic Discovery - No “one size fits all” Solution

AMELIO Logic Discovery helps to understand the existing COBOL- and PL/I-applications and thus reduces the costs for re-implementation of the existing functions and for the modernization of the applications.

Further information can be found under: **www.delta-software.com/amld**

Delta Software Technology

Delta Software Technology is a specialist for generative development tools that automate the modernisation, integration, development and maintenance of individual IT applications.

Our solutions help you to quickly and safely adapt your applications to new business requirements, architectures, technologies and technical infrastructures.

AMELIO® Modernization Platform™
The tailor-made factory for the modernisation of large IT applications:
100% automatically and that's why it is safe, reliable and error-free.

AMELIO® Logic Discovery
Comprehending COBOL- and PL/I-Applications:
Cut costs and risks for maintenance, modernization and re-implementation.

HyperSenses®
Integrated system for model driven development of DSLs and software generators.

SCORE® Adaptive Bridges™
Intelligent service enablement for the reuse of proven applications with modern technologies: flexible, profitable and non-invasive.

SCORE® Data Architecture Integration™
Data as real business services: fast, easy and independent of data architectures and management systems.

SCOUT²™ Development Platform
Optimized and integrated development processes across all software components, tools and platforms: Stop the “fight against the infrastructure”.

ADS™ Application Development for COBOL and PL/I
Platform independent development for future-proof back-end applications.

Delta has a more than 35-year track record of successfully delivering advanced software technology to Europe’s leading organisations, including AMB Generali, ArcelorMittal, Deutsche Telekom, Hüttenwerke Krupp Mannesmann, Gothaer Versicherungen, La Poste, RDW, Suva and UBS.

Delta Software Technology GmbH
Eichenweg 16
57392 Schmallenberg, Germany

phone +49 2972 9719-0
fax +49 2972 9719-60
e-mail info@delta-software.com

www.delta-software.com

Order number: MA 24’017.02 – March 2015