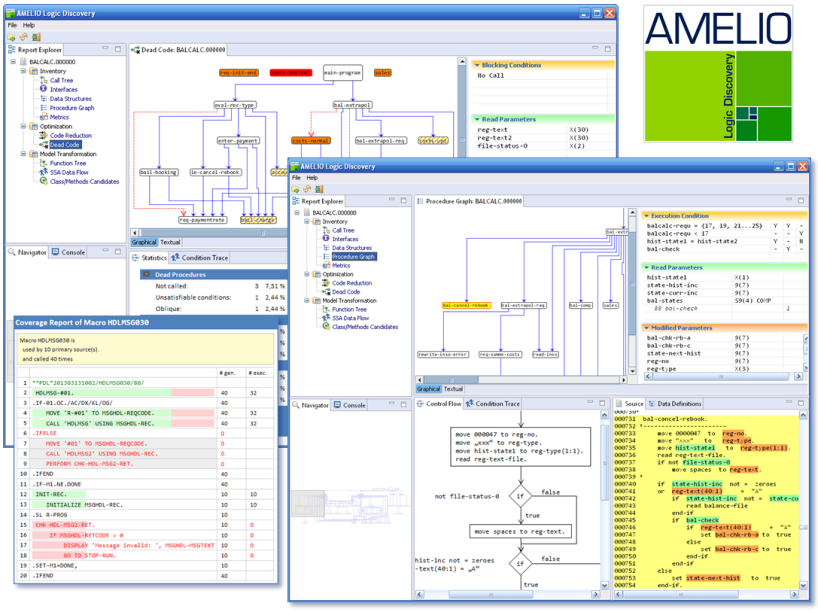
DELTA software technology

AMELIO LOGIC DISCOVERY

Frühjahrsputz für COBOL- und PL/I-Anwendungen

Der Frühling naht und die Sonne bringt Licht in Wohnungen und Häuser. Staub und Schmutz, die sich in der dunklen Jahreszeit ungesehen ansammeln konnten, werden sichtbar, so dass in vielen Haushalten nun ein Frühjahrsputz beginnt. Warum den Frühjahrsputz nicht auch gleich auf Ihre COBOL- und PL/I-Anwendungen ausweiten? Solche Anwendungen sind über viele Jahre oder sogar Jahrzehnte gewachsen. In dieser Zeit haben sie gelebt - sie wurden gewartet, erweitert, an neue Anforderungen angepasst und sind dadurch gewachsen. An vielen Stellen hat sich überflüssiger Ballast angesammelt, Programmstrukturen haben sich geändert, Dokumentation ist nicht mehr vorhanden oder nicht mehr aktuell. Die Wartung und das Verstehen der Anwendung werden immer schwieriger.



AMELIO Logic Discovery-CleanUp

AMELIO Logic Discovery extrahiert die implementierte Anwendungslogik aus COBOL- und PL/I-Programmen und hilft somit die Anwendungen zu verstehen. Die Ermittlung der Anwendungslogik erfolgt in den drei Schritten Inventur, Code Optimierung und Logikanalyse. Die Inventur ermittelt, aus welchen Bestandteilen – also Programmen, Schnittstellen und Datenstrukturen, sowie den entsprechenden Zusammenhängen - sich die Anwendung zusammensetzt. Bei der Code-Optimierung werden die für die Wartung und das

Verstehen relevanten Anteile der Anwendung ermittelt, dabei wird z.B. toter Code festgestellt, dokumentiert und ggf. entfernt. In der Logikanalyse werden aus den Informationen der vorangegangenen Schritte Modelle erzeugt, die sowohl unabhängig von den verwendeten Programmiersprachen als auch den –paradigmen sind. Auf diese Weise wird die Anwendungslogik extrahiert und verständlich repräsentiert.

Die Inventur sowie die Dead Code-Analyse und –Bereinigung aus der Code Optimierung haben wir im CleanUp-Paket von AMELIO Logic Discovery zusammengefasst. Dieses Paket ist dazu gedacht, existierende Anwendung zu re-dokumentieren und vor allem unnötigen Ballast zu erkennen, zu dokumentieren und zu entfernen und somit effizient, zuverlässig und automatisch den Frühjahrsputz durchzuführen.

Inventur – Mehr als nur eine Bestandsaufnahme

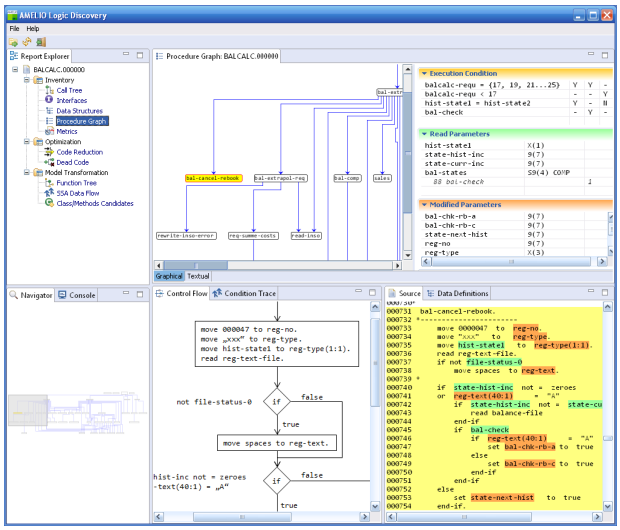
Der erste Schritt, sowohl beim Frühjahrsputz als auch beim Verstehen einer Anwendung, besteht darin festzustellen, was sich in der vergangenen Zeit alles angesammelt hat. In der Inventur wird deshalb z.B. ermittelt, aus welchen Programmen sich die Anwendung zusammensetzt, welche Schnittstellen und Datenstrukturen es gibt und welche Zusammenhänge bestehen. Hierbei geht AMELIO Logic Discovery über eine reine Bestandsaufnahme hinaus, stattdessen werden bereits Analysen durchgeführt:

- Klassische Metriken, z.B. Halstead oder McCabe
- Auflistung existierender Datendefinitionen
- Auflistung aller Datenbank- und Dateizugriffe
- Auflistung aller Schnittstellendefinitionen und möglicher Konflikte
- Darstellung von Programm- und Unterprogrammaufrufen
- Darstellung von Prozeduren, ihren Aufrufhierarchien, Bedingungen und Schnittstellen

In der Analyse-Workbench von AMELIO Logic Discovery werden die Ergebnisse der verschiedenen Analysen dargestellt. Als Beispiel für eine solche Analyse und der Darstellung ihrer Ergebnisse wird im Folgenden ein besonderes Highlight der Inventur vorgestellt: Die Repräsentation von Prozeduren zusammen mit ihrer Aufrufhierarchie, den Schnittstellen, verwendeten Parametern und vor allem den Bedingungen, wann die jeweilige Prozedur ausgeführt wird. Da es in COBOL keine expliziten Prozeduren gibt, wird jedes Programm der Anwendung analysiert. Anhand der vorgefundenen Aufrufstrukturen werden dann verschiedene Abschnitte des Programms zu Prozeduren zusammengefasst. Für PL/I, wo es explizite Prozeduren gibt, können diese als Ausgangsbasis für die folgenden Analysen verwendet werden, allerdings verbergen sich hier viele Prozeduraufrufe hinter Entry-Variablen und Generics, die erst aufgelöst werden müssen. Die gefundenen Prozeduren und ihre Aufrufstruktur werden in Form eines Graphen dargestellt. Das Analyse-Ergebnis ist beispielhaft in der folgenden Grafik dargestellt:

- Prozedurgraph: Stellt die ermittelten Prozeduren und ihre Aufrufhierarchie dar, ebenso die Aufrufe von Unterprogrammen und „wilde“ Verzweigungen mit GOTO.
- Bedingungen: Im Rahmen der Inventur erfolgt bereits eine Bedingungsanalyse. Bei dieser Analyse wird ermittelt, welche Bedingungen erfüllt sein müssen, damit eine Prozedur überhaupt ausgeführt wird. Das Ergebnis wird in Form von Entscheidungstabellen dargestellt.
- Datenstrukturen (COBOL): Es wird dargestellt, welche Datenstrukturen durch die Prozedur gelesen und welche modifiziert werden.
- Datenstrukturen (PL/I): Es wird dargestellt, welche globalen Datenstrukturen durch die Prozedur gelesen und welche modifiziert werden.

- Schnittstellen: Speziell für PL/I-Anwendungen wird zusätzlich die Schnittstelle einer Prozedur angegeben und falls vorhanden auch deren Rückgabewert.
- Kontrollfluss: Zu jeder Prozedur wird deren Kontrollfluss angezeigt.
- Code-Anteile: Darstellung, welche Codeanteile die Prozedur bilden, darin werden gelesene und modifizierte Datenstrukturen farblich markiert.



Toter Code – Unnötigen Ballast entfernen

Je länger eine Anwendung lebt, desto öfter wurde sie erweitert und angepasst. Von daher ist davon auszugehen, dass mit dem Alter der Anwendung die Menge toten Codes in der Anwendung steigt. Also Code, der zwar vorhanden ist, aber nie ausgeführt werden kann. Solcher Code muss bei Wartungsmaßnahmen immer mitberücksichtigt werden, er erschwert das Verstehen der Anwendungslogik und belegt Speicherplatz. Neben der Inventur ist deshalb die Erkennung und Beseitigung des toten Codes ein zentraler Bestandteil des Frühjahrsputzes.

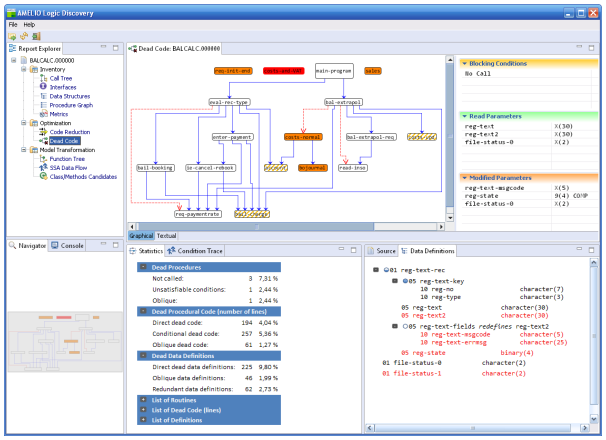
Die Dead Code-Analyse wird pro Programm durchgeführt, sie setzt sich aus den folgenden Bestandteilen zusammen, die aufeinander aufbauen:

- Direct Dead Code: prozeduraler Code, der nicht ausgeführt wird, da der entsprechende Abschnitt nie aufgerufen wird
- Conditional Dead Code: prozeduraler Code, der nur unter bestimmten Bedingungen ausgeführt wird (z.B. innerhalb einer If-Anweisung), die entsprechende Bedingung jedoch nie erfüllt werden kann
- Oblique Dead Code: Dies sind prozedurale Code-Anteile, die nur aus dem zuvor gefundenen toten Code aufgerufen werden
- Direct Dead Data Definitions: Datenstrukturen, die definiert wurden, aber nie verwendet werden
- Oblique Dead Data Definitions: Datenstrukturen, die definiert wurden, aber nur innerhalb von totem prozeduralem Code verwendet werden
- Redundant Data Definitions: Stellen in dem Sinne keine Dead Data Definitions dar, sondern sind durch kopieren und einfügen entstanden, dennoch können sie im Rahmen der Dead Code-Analyse ermittelt werden.

Das Ergebnis der Dead Code-Analyse wird in der folgenden Graphik dargestellt:

- Reduzierter Prozedurgraph: Dieser Graph enthält alle toten Prozeduren (in orange eingefärbt, die selektierte Prozedur ist rot eingefärbt) und alle Prozeduren, die toten Code enthalten (orange schraffiert). Wird eine Prozedur als tot erkannt, weil die Bedingung zu ihrer Ausführung nie erfüllt werden kann, so ist der entsprechende Aufruf orange markiert. Außerdem werden alle Prozeduren angezeigt, die entweder in der Aufrufhierarchie oberhalb einer der zuvor genannten Prozeduren liegen oder alternative Aufrufpfade zu einer Prozedur darstellen, die aus einer toten Prozedur aufgerufen werden.
- Blocking Condition: Die Bedingungen, die eine Ausführung der Routine verhindern.
- Data Definitions: Stellt die Datendefinitionen dar. Tote Datendefinitionen sind rot markiert.

- Statistics: Statistik für das gesamte Programm. Sie gibt an wie viele tote Routinen, Zeilen prozeduraler Code und Datendefinitionen ermittelt wurden. Außerdem werden die entsprechenden Elemente aufgelistet.



Erweiterte Dead Code-Analyse

Die Dead Code-Analyse von AMELIO Logic Discovery geht jedoch noch einen Schritt weiter. Sowohl bei der Entwicklung von COBOL- als auch PL/I-Anwendungen werden Copybooks bzw. Includes oder, im Falle von ADS-Anwendungen, Makros verwendet. In diesen Fällen reicht die Analyse einzelner Programme nicht aus, denn der tote Code in den Programmen kann durch die Copybooks, Includes oder Markos entstanden sein. Darüber hinaus ist es auch möglich, dass diese Code enthalten, der nie kompiliert bzw. generiert wird. Solcher Code wird ebenfalls durch AMELIO Logic Discovery ermittelt. Um die Analyse auch für Copybooks und Includes zu ermöglichen, bildet AMELIO Logic Discovery die entsprechende Compile-Funktionalität nach. Das Ergebnis der Coverage Analyse enthält die folgenden Informationen:

- wie viele Primärsourcen das Copybook, den Include oder das Makro verwenden
- wie oft das Copybook oder der Include bei der Kompilierungen bzw. das Makro bei Generierungen aufgerufen wurden
- wie oft bestimmte Abschnitte eines Copybooks oder Includes bei einer Kompilierung bzw. eines Makros bei der Generierung verwendet wurden
- in wie vielen Fällen der erzeugte Code tatsächlich ausführbar ist

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	

Codebereinigung

Nachdem der tote Code in all seinen Facetten ermittelt und dokumentiert wurde, kann er nun entfernt werden. Eine manuelle Bereinigung birgt allerdings die Gefahr, dass einige Zeilen nicht gelöscht oder zu viel gelöscht werden, so dass die Programmlogik verändert wird. Sicherer und schneller ist es daher die Transformationsfunktion von AMELIO zu nutzen. Diese entfernt den gesamten toten Code, Anweisungen ebenso wie Datendefinitionen.

Fazit

COBOL- und PL/I-Anwendungen sind über viele Jahre und Jahrzehnte gewachsen. Neben der wirklich notwendigen Funktionalität hat sich dabei auch eine Menge überflüssiger Ballast angesammelt, der die Wartung und das Verstehen der Anwendungen erschwert. AMELIO Logic Discovery hilft mit seinen Inventur-Funktionen einen Überblick darüber zu gewinnen, aus welchen Bestandteilen sich die Anwendung zusammen setzt und in welcher Beziehung diese zueinander stehen. Die Dead Code-Analyse ermittelt toten Code, sowohl tote Anweisungen als

auch Datendefinitionen, und kann diese auch automatisch entfernen. Als Ergebnis erhält man aussagekräftige Dokumentationen sowie bereinigte Anwendungen, die wesentlich leichter zu warten und zu verstehen sind. AMELIO Logic Discovery-CleanUp unterstützt also bei der Wartung und der Qualitätssicherung.

Beginnen Sie nun mit dem Frühjahrsputz Ihrer COBOL- und PL/I-Anwendungen. Das CleanUp-Paket von AMELIO Logic Discovery leuchtet auch die versteckten Ecken Ihrer Anwendungen aus, zeigt den Staub und entfernt Ihn für Sie - schnell, zuverlässig und effizient.

Prädikat „BEST OF 2015“ für AMELIO Logic Discovery

Die Initiative Mittelstand zeichnete AMELIO Logic Discovery bei der Verleihung des Innovationspreises mit dem Prädikat „BEST OF 2015“ aus.

AMELIO Logic Discovery - Keine „one size fits all“-Lösung

AMELIO Logic Discovery hilft, die vorhandenen COBOL- und PL/I-Anwendungen zu verstehen und senkt so die Kosten für die Neu-Implementierung der vorhandenen Funktionen sowie der Modernisierung der Anwendungen.
Weitere Informationen erhalten Sie unter: **www.delta-software.com/amld**

Delta Software Technology

Delta Software Technology ist Spezialist für generative Software-Werkzeuge, die die Modernisierung, Integration, Entwicklung und Wartung individueller IT-Anwendungen automatisieren.

Unsere Lösungen helfen Ihnen, Ihre Anwendungen schnell und sicher an neue Geschäftsanforderungen, Architekturen, Technologien und technische Infrastrukturen anzupassen.

AMELIO® Modernization Platform™
Maßgeschneiderte Factory für die Modernisierung großer IT-Anwendungen:
100% automatisch und deshalb sicher, zuverlässig und fehlerfrei.

AMELIO® Logic Discovery
COBOL- und PL/I-Anwendungen verstehen: Kosten und Risiken für Wartung, Modernisierung und Neu-Implementierung senken.

HyperSenses®
Integriertes System für modellgetriebene Entwicklung von DSLs und Software-Generatoren.

SCORE® Adaptive Bridges™
Intelligentes Service Enablement für die Wiederverwendung bewährter Anwendungen mit modernsten Technologien: Flexibel, rentabel und non-invasiv.

SCORE® Data Architecture Integration™
Daten als echte Business Services: Schnell, einfach und unabhängig von Datenarchitekturen und Speicherungsformen.

SCOUT²™ Development Platform
Optimierte und integrierte Entwicklungsprozesse über alle Software-Komponenten, Werkzeuge und Plattformen: Stoppt den „Kampf gegen die Infrastruktur“.

ADS™ Application Development for COBOL and PL/I
Plattformunabhängige Entwicklung für zukunftssichere Back-End-Anwendungen.

Delta liefert seit mehr als 35 Jahren erfolgreich fortschrittliche Software-Technologie an Europas führende Organisationen, zu denen u.a. AMB Generali, ArcelorMittal, Deutsche Telekom, Hüttenwerke Krupp Mannesmann, Gothaer Versicherungen, La Poste, RDW, Suva und UBS gehören.

Delta Software Technology GmbH
Eichenweg 16
57392 Schmallenberg

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

www.delta-software.com

Bestellnummer: MA 14‘017.02 – März 2015