SAP R/3 ANALYSIEREN

Zusammenfassung
Das SAP R/3 System gilt als die derzeit umfassendste und komplexeste verfügbare betriebswirtschaftliche Standardsoftware. Es handelt sich dabei nicht um ein statisches System, sondern um eine stetig wachsende und verändernde Software. Diese Situation ist Anlass für die Frage, ob die mit Wachstum und Veränderung verbundenen Wartungsaufgaben optimiert werden können. Da bei der Durchführung einer Wartungsarbeit 47 Prozent des notwendigen Aufwands auf die Analyse der bestehenden Software und der Wartungsaufgabe entfallen, wurde dieser Aspekt zum Ausgangspunkt bei der Suche nach Optimierungsmöglichkeiten.

Die Ergebnisse dieser Untersuchung werden im Folgenden dargestellt. Dabei wird das derzeitige Programmvolumen des R/3 Systems, die allgemeine Problematik der Softwarewartung und die mentalen Prozesse bei der Analyse von Quelltexten erläutert. Darauf aufbauend wird schließlich der Magellan Explorer dargestellt. Der Magellan Explorer ist die prototypische Implementierung eines Werkzeugs, dass SAP R/3 Programme automatisch analysieren und die verschiedenen Design – Aspekte eines ABAP Quelltextes visualisieren kann.

Business Computing in ABAP

ABAP hat in den vergangenen 15 Jahren eine bemerkenswerte Entwicklung durchlaufen: Von der “Allgemeinen Berichts- und Abfragesprache” des R/2 – Systems zur “Advanced Business Application Language” des R/3 – Systems. Von der ursprünglichen Idee eines Berichtsprozessors ausgehend hat sich ABAP zu einer der bedeutendsten Programmiersprachen unserer Zeit entwickelt. Diese Bedeutung bezieht sie nicht nur durch ihre allgemeine Einsatzfähigkeit oder ihre breite Verwendung, sondern besonders durch ihren Einsatz an den empfindlichsten Stellen der modernen Unternehmenswelt – jenen Stellen, an denen die Informationsströme einer Firma zusammenlaufen und auf deren Basis Entscheidungen getroffen werden: der integrierten Standardanwendung SAP R/3. Diese herausgehobene Stellung von ABAP ist Motivation für eine Untersuchung des in ihr implementierten Programmvolumens.

Art	SAP R/3 4.6C	SAP R/3 4.7 Enterprise Edition
Transaktionen	57.930	70.948
Programme	665.175	1.032.074
Tabellen	96.940	138.464
Tabellenfelder	1.377.231	1.888.050
Fremdschlüssel	437.259	536.469
Programmzeilen	81.794.230	109.051.858

Tabelle 1: Programmvolumen des SAP R/3 – Systems

Tabelle 1 zeigt die quantitative Entwicklung der im SAP R/3 System enthaltenen Programmelementen vom 4.6C Release hin zur aktuellen Version, der Enterprise

Edition. Die ohnehin schon gewaltige Anzahl von Programmelementen ist durch den letzten Releasewechsel noch um etwa 25% gestiegen. Auf Basis dieser Zahlen kann angenommen werden, dass es sich um eines der komplexesten heute im Einsatz befindlichen und kommerziell zugänglichen Systeme handelt. Zum Vergleich: die Software des Apollo – Mondlandeprogramms beinhaltete ca. 1 Millionen Anweisung (bei der letzten Mission 1973), die Software des Space Shuttle ca. 8 Millionen Anweisung (im Jahre 1985). Für die umfassende und integrierte Implementierung betriebswirtschaftlicher Prozesse sind also ca. 100 mal mehr Anweisungen notwendig, als für eine Landung auf dem Mond.[1]

Softwarewartung

Über die Wartung eines solchen Programmvolumens, wie es im R/3 – System enthalten ist, liegen bis heute wenig publizierte Erkenntnisse vor. Dennoch gibt es vergleichbare Situationen. Im Zusammenhang mit dem Jahr-2000-Problem und er EURO – Umstellung mussten komplexe Systeme, die über einen Zeitraum von 20 oder mehr Jahren entstanden waren, entweder durch neue Systeme ersetzt oder umgestellt werden. Um Arbeiten dieser Art besser einschätzen zu können, entstanden zwischen dem Beginn und der Mitte der 90er Jahre verschiedene Studien, in der dieses Problemfeld, das Reengineering, detailliert untersucht wurde [vgl. Balzert, Sneed, Müller].

Diese Studien zeigten, dass bis zum Beginn der 90er Jahre für die Wartung einer Software etwa 80 Prozent der gesamten Entwicklungszeit erforderlich waren. Mit Beginn der 90er Jahre konnte ein Absinken dieses Anteils festgestellt werden. Entscheidender Faktor war der als Paradigmenwechsel bezeichnete Übergang von klassischen Mainframe - Anwendungen zu Client / Server - Anwendungen. Während des Paradigmenwechsels wurden viele ältere Anwendungen im betriebswirtschaftlichen Umfeld bevorzugt durch neue Standardlösungen oder individuelle Neuentwicklungen ersetzt. Ein weiterer Grund für den sich verringernden Wartungsaufwand war die PC-Revolution, mit der sich in den 80er Jahren die Softwareentwicklung veränderte: der Einsatz von CASE-Tools wurde direkt am Entwickler-Arbeitsplatz möglich, so dass sich die Qualität der zwischen 1980 und 1990 erstellten Software veränderte.

Die CASE-Werkzeuge sind heute Teil eines jeden Software-Entwicklungsprozesses. In den meisten Fällen endet ihr Leistungsspektrum jedoch mit dem Ende der Entwurfsphase und dem Beginn der Implementierung. Wird die Implementierung heute durch moderne Entwicklungsumgebungen hinreichend unterstützt, so finden sich für die auf die Implementierung folgende Wartungsphase (die 80 Prozent der Entwicklungszeit in Anspruch nimmt) nur eingeschränkt unterstützende Werkzeuge.

Begleitet wird der weitgehende Ausfall von Werkzeugen in der Wartungsphase durch die oft unzureichende Dokumentation komplexer Systeme. Fragestellungen nach Zusammenhängen und Strukturen, die zum Beispiel zehn Jahre nach der Dokumentation gestellt werden, können nicht mehr hinreichend beantwortet werden. Das System wurde weiterentwickelt, die ursprünglichen Strukturen modifiziert und so unter Umständen die gesamte Architektur verändert, ohne dass dies hinreichend vermerkt wurde.

[1] Ein Vergleich dieser Art ist zulässig, da eine Programmzeile in ABAP auf 72 Zeichen begrenzt ist und durch die Länge der Anweisungen i. d. R. nicht mehr als eine Anweisung je Zeile möglich ist.

An dieser Stelle muss in der Regel der Quelltext manuell und zeitaufwendig analysiert werden. Dieser Verstehensprozess nimmt, unabhängig von der Art einer Änderungen (Korrektur oder Erweiterung), einen Anteil von 47 Prozent an dem Gesamtaufwand für eine Änderung ein. Die verbleibenden 53 Prozent teilen sich in Implementierung (19 Prozent), Test (28 Prozent) und Dokumentation (6 Prozent). Die tatsächliche Codierung einer Änderung nimmt also weniger als ein Viertel des Aufwands für eine Änderung ein. Der Aufwand wird also in überwiegendem Maß durch das Verstehen und Lokalisieren der Änderung bestimmt, also den mentalen Fähigkeiten eines Entwicklers.

Quelltextanalyse von SAP R/3 Programmen

Die Bewältigung des Verstehensprozesses, die Analyse, eines R/3 Quelltextes stellt einen Entwickler vor zwei grundsätzliche Probleme: zunächst einmal kann ein geschulter Entwickler nicht mehr als drei bis fünf Quelltextzeilen auf einen Blick erfassen und verstehen. Für das Verstehen eines umfangreichen Programms ergibt sich so ein erheblicher zeitlicher Aufwand. Ursache für dieses Problem ist das aus der Psychologie bekannte „Concept Assignment Problem“. Dieser Begriff beschreibt einen Abstraktionsvorgang, bei dem versucht wird, die Bedeutung eines Quelltextabschnitts zu erfassen, mit dem Wissen des Entwicklers über den tatsächlichen betriebswirtschaftlichen Prozess in Einklang zu bringen und so festzustellen, welcher Teil eines betriebswirtschaftlichen Prozesses durch den gerade analysierten Quelltextabschnitt implementiert wird.

Um ein Programm auch dann verstehen zu können, wenn das detaillierte Prozesswissen nicht ausreicht, wird die Analyse von Quelltexten häufig intuitiv mit verschiedenen Lösungsstrategien verknüpft: der Top-Down-Strategie, der Bottom-Up-Strategie oder einem opportunistischen Ansatz. Bei Anwendung der Top-Down-Strategie wird versucht, die Implementierung eines Prozessabschnitts in einem Programm zu verifizieren. Für die Reservierung von Sitzplätzen in einem Flugbuchungssystem werden zum Beispiel Datenstrukturen wie *seat* oder Funktionen wie *reserve* erwartet. Durch das Suchen nach solchen Strukturen oder Funktionen werden die anfangs getroffenen Annahmen bestätigt oder verworfen. Die Implementierung eines Prozesses durch ein Programm wird so schrittweise erforscht bis schließlich die gesamte Prozessimplementierung bekannt ist. Dem umgekehrten Weg folgt die Bottom-Up-Strategie. Dabei wird versucht, aus dem Quelltext eines Programms den gesamten implementierten Prozess zu rekonstruieren. Der opportunistische Ansatz verknüpft beide Strategien: über den Top-Down-Ansatz werden die für den Prozess entscheidenden Quelltextabschnitte identifiziert und anschließend über den Bottom-Up-Ansatz im Detail analysiert.

Bedeutsam für die Analyse von Quelltexten sind zudem die so genannten Software – Metriken, deren bekanntester Vertreter die Halstead - Metriken sind. Beim Einsatz dieses Metriksystems ist festzustellen, dass es (unabhängig von der verwendeten Programmiersprache) einen numerischen Zusammenhang zwischen dem Aufwand für das Verstehen eines Quelltextes und dem Volumen eines Quelltextes gibt. Diese, von Maurice Halstead als „Difficulty“ bezeichnete charakteristische Konstante eines Programms gibt an, welcher Aufwand für das Verstehen, das Ändern oder erneute Implementieren eines Quelltextes notwendig ist. Da die Konstante auf der Zählung von Quelltextelementen (zum Beispiel der Anzahl unterschiedlicher Anweisungen) basiert, ist sie frei von Maßeinheiten und kann deshalb als empirische Größe in einer

Gleichung zur Aufwandsabschätzung eingesetzt werden. Die verwendete Gleichung ist jedoch individuell abhängig von einem Entwickler, einer Entwicklergruppe oder der verwendeten Programmiersprache. Davon unabhängig ist jedoch die Tatsache, dass der Aufwand für das Verstehen eines Quelltextes nicht zufällig, sondern quantifizier- und vorhersagbar ist.

Neben der Quantifizierung des für die Analyse eines Programms notwendigen Aufwands ist die Frage entscheidend, wie sich die Analyse durch Werkzeuge besser unterstützen lässt. Hintergrund dieser Frage ist dabei weniger ein Rationalisierungs- als ein Effizienzgedanke, da es sich bei der Analyse eines Quelltextes um einen komplexen mentalen Prozess handelt, der anstrengend und zeitaufwendig ist.

Der Botttom-Up-Ansatz stellt einen Reduktionsprozess dar, bei dem mehrere Quelltextzeilen sinnvoll zu logischen Einheiten zusammengefasst und in Beziehung zueinander gesetzt werden. Dieses Reduzieren und Verknüpfen führt schließlich zur Rekonstruktion des implementierten Prozesses. Da der Reduktionsprozess durch die Grammatik der Programmiersprache vorgegeben ist kann dieser weitgehend automatisiert werden. Ein solches statisches Quelltextanalysetool ist ein Programm, dass die inneren Zusammenhänge eines Programms analysiert ohne dieses auszuführen und so die technische Implementierung eines Prozesses wiedergeben kann. Resultat des automatisierten Reduktionsprozesses kann sowohl ein Satz charakteristischer Konstanten als auch eine Visualisierung sein, wobei die Visualisierung für das Verstehen von Struktur und Funktion eines Programms hilfreich ist, während ein Satz von Konstanten die quantitative Beurteilung eines Programms ermöglicht.

Visualisierung von SAP R/3 Programmen

Für eine qualitativ hochwertige Visualisierung eines R/3 Quelltextes gilt, dass sie grundsätzlich expressiv, effektiv und angemessen sein muss. Diese Ziele werden durch die Implementierung eines entsprechenden Visualisierungsprozesses erreicht, dessen Ziel die Veranschaulichung abstrakter Daten (Quelltexten) in Form von Grafiken ist. Dabei durchläuft der Visualisierungsprozess drei Schritte, die in der so genannten Visualisierungspipeline angeordnet werden können. Diese drei Schritte, Filterung, Abbildung und Darstellung, beschreiben die Wandlung abstrakter Daten zu Grafiken.

Für die Filterung bilden die Quelltextzeilen den Ausgangpunkt. Innerhalb der Filterung werden diese reduziert und interpoliert. Die Reduktion fasst einzelne Quelltextelemente sinnvoll in Blöcken zusammen. Die Regeln der Reduktion werden von der Art der gewählten Analyse und den Eigenschaften der dem Quelltext zu Grund liegenden Programmiersprache bestimmt. Neben dem Quelltext können auch zusätzliche Informationen (zum Beispiel Programmbeschreibungen) in die Analyse mit einbezogen werden, wenn der Inhalt des Codings für eine expressive und effektive Visualisierung nicht ausreichend ist.

Die Abbildung stellt das Kernstück der Visualisierungspipeline dar. Hier werden die in der Filterung aufbereiteten Quelltexte auf ein Koordinatensystem abgebildet und mit den entsprechenden Symbolen verknüpft. Eine Funktion wird in diesem Prozessschritt zum Beispiel auf einen Kreis abgebildet und entsprechend ihrer Verwendung eingefärbt. Da es sich bei Quelltexten um Daten handelt, die in der Natur keinerlei Gestalt haben, stellt die Abbildung derselben eine erhebliches

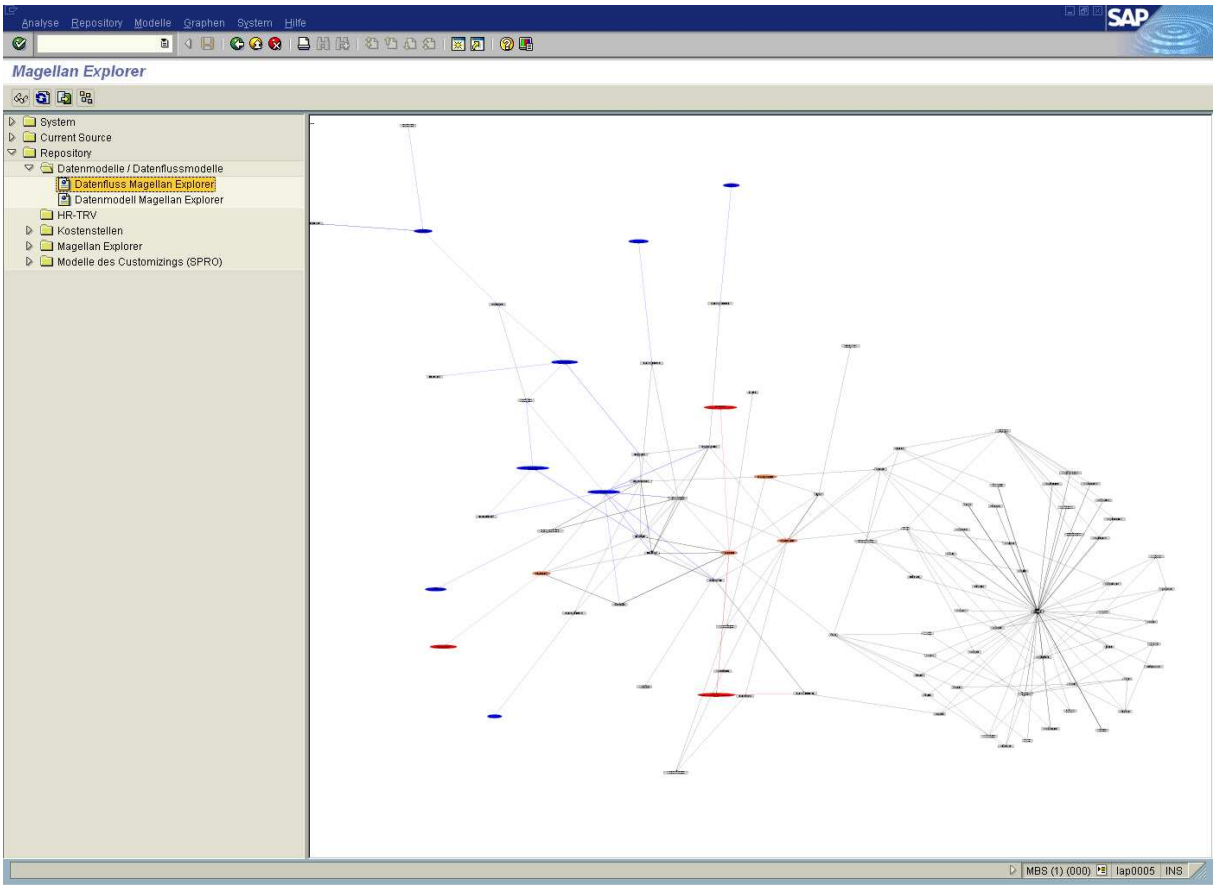

Abb. 1: Visualisierung des Datenflusses in einem ABAP Programms

Problem dar. Durch die Hinterlegung von physikalischen Modellen aus der Molekulardynamik kann dennoch eine plastische und charakteristische Visualisierung erreicht werden.

An die Abbildung schließt sich innerhalb der Visualisierungspipeline die Darstellung an. Hier werden die Ergebnisse der Abbildung dem Benutzer auf dem Bildschirm zur Anzeige gebracht. Abhängig von der Komplexität der Anzeigeapplikation kann ein Entwickler mit Hilfe explorativer Funktionen (Zoom, Drehung oder Begehen im Fall einer dreidimensionalen Visualisierung) die Struktur und technische Umsetzung eines Prozesses effizient und vollständig rekonstruieren.

Einen Prototypen für die Analyse und Visualisierung von SAP R/3 - Programmen stellt der Magellan Explorer dar. Seine Visualisierungsfunktionen unterstützen den Entwickler bei der Analyse eines beliebigen ABAP - Quelltextes. Für eine Demonstration der Leistungsfähigkeit der dargestellten Analysetechnologie wurden innerhalb des Magellan Explorer vier Analysefunktionen implementiert, die für das Verstehen von R/3 Programmen von entscheidender Bedeutung sind.

- Die Modulstrukturanalyse
- Die Datenmodellanalyse
- Die Aufrufanalyse
- Die Datenflussanalyse

Die Modulstrukturanalyse gibt einen Überblick über die Quelltextmodule, die ein Programm bilden und beantwortet so die Fragen, welche Module wie eingebunden

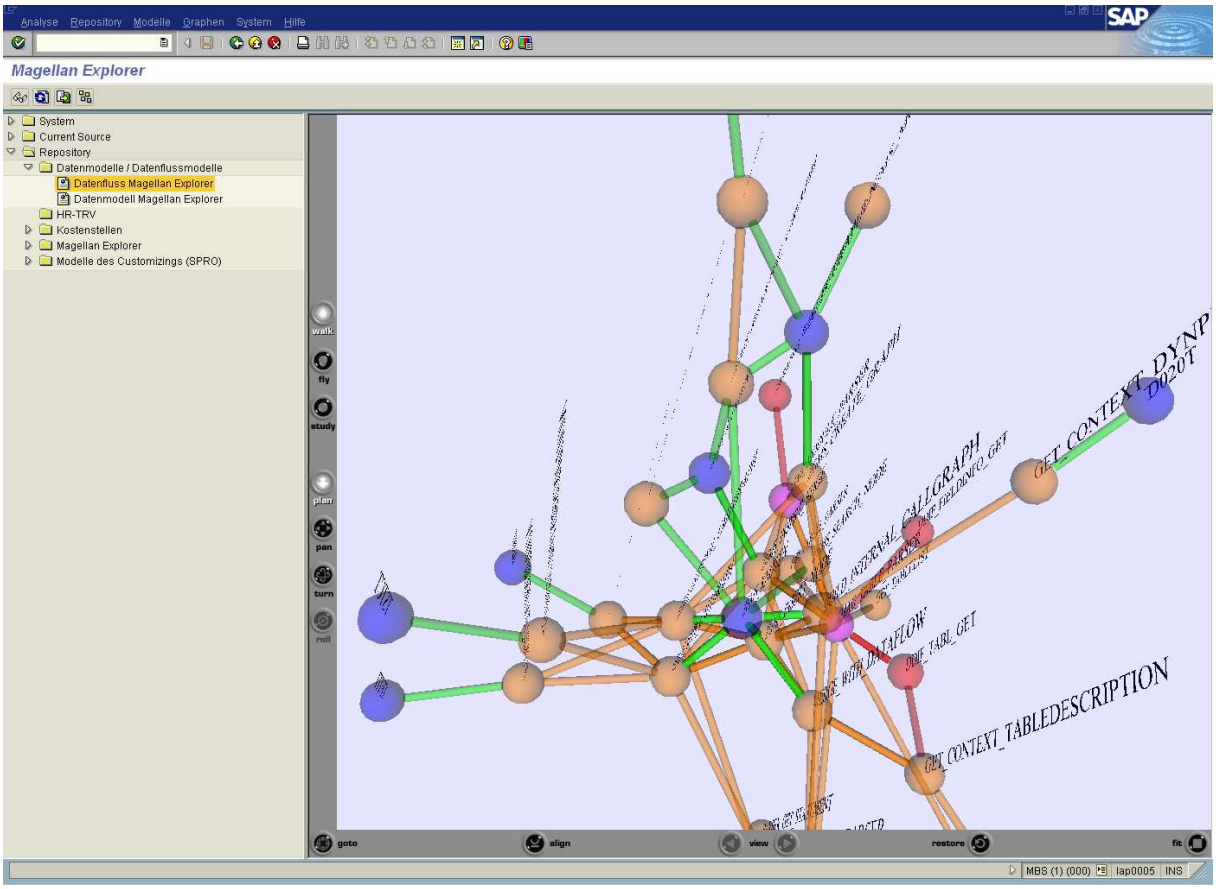

Abb. 2.: 3D - Visualisierung des Datenflusses in einem ABAP Programm

werden und welchen Modulen eine zentrale Bedeutung zukommt. Auf diesem Weg wird ein umfassender Überblick über die Organisation eines Programms möglich.

Neben der Organisation des Quelltextes sind Struktur und Inhalt des einem Programm zugeordneten Datenmodells von charakteristischer Bedeutung. Die Datenmodellanalyse gibt Auskunft darüber, auf welche Datenbanktabellen ein Programm explizit zugreift und wie die Beziehungen zwischen den Tabellen ausgestaltet sind. Darauf aufbauend bietet das Anschlussdatenmodell dann die Möglichkeit zu analysieren, wie das programmlokale Datenmodell in das globale Datenmodell des R/3 – Systems eingebettet ist.

Das Bindeglied zwischen Modulstruktur- und Datenmodellanalyse bilden Aufruf- und Datenflussanalyse. Die Aufrufanalyse stellt in grafischer Form dar, welche Funktionen innerhalb eines Quelltextes gerufen werden. Ein Beispiel einer Datenflussanalyse ist in Abb. 1 dargestellt. Man erkennt, dass in der Datenflussanalyse die Aufrufanalyse insofern ergänzt wird, als dass hier auch die lesenden und schreibenden Zugriffe auf Datenbanktabellen berücksichtigt werden, die von einer Funktion vorgenommen werden.

Ergänzt werden die vier analytischen Funktionen schließlich durch die Möglichkeit, die Ergebnisse einer beliebigen Analyse dreidimensional darzustellen (vgl. Abb. 2). Diese Zusatzfunktion wurde geschaffen, um der Komplexität der Analyseresultate gerecht zu werden und auch an dieser Stelle einen innovativen Lösungsansatz bieten zu können.

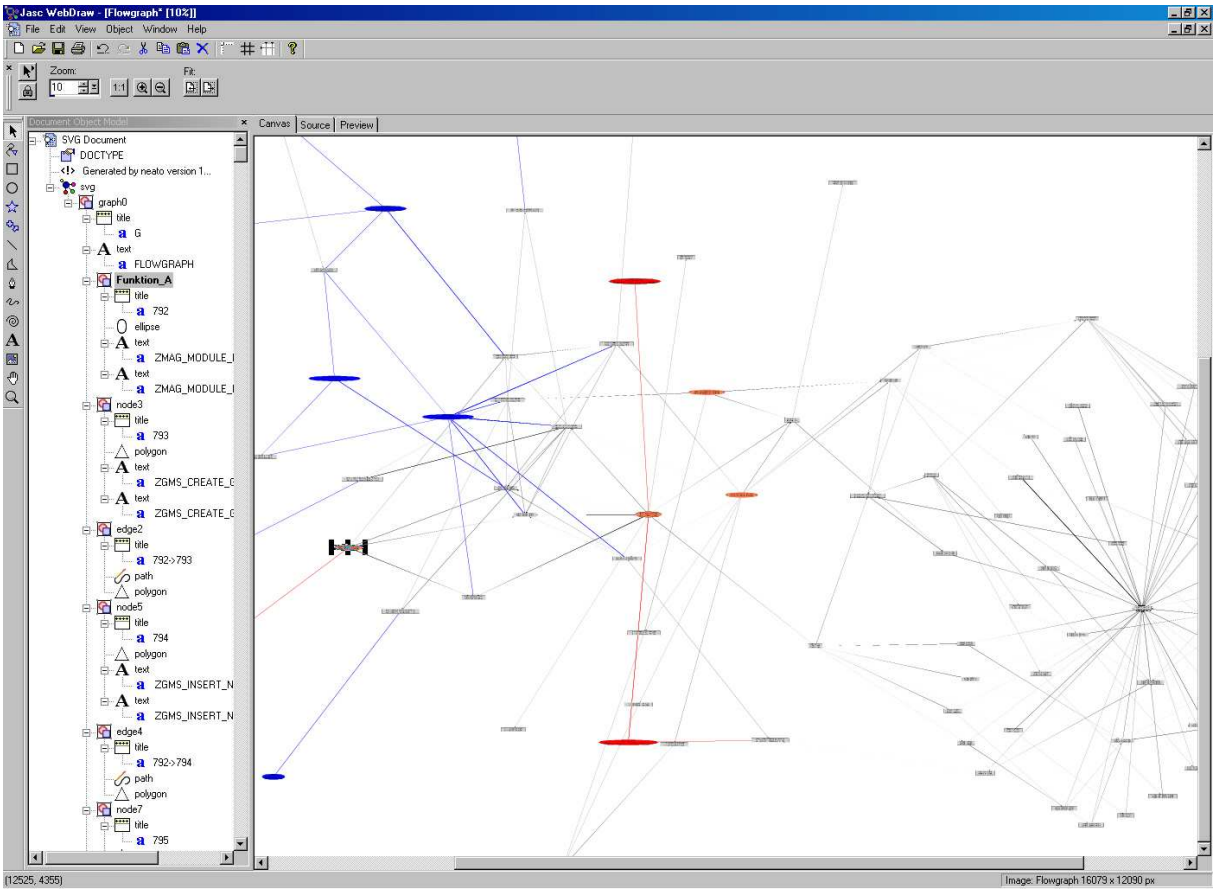

Abb. 3.: Bearbeiten eines Datenflussmodells in einem externen Editor

Prototyp für die SAP R/3 Quelltextanalyse: der Magellan Explorer

Der gesamte Magellan Explorer ist vollständig in ABAP implementiert und nahtlos in das R/3 – System integrierbar. Durch die vollständige Freiheit von R/3-eigenem Code kann der Magellan Explorer darüber hinaus auch in jede andere SAP – Software integriert werden, die auf ABAP basiert. Die technische Trennung wird dadurch erreicht, dass der Magellan Explorer aus einem Kernsystem besteht, in dem die analytischen Funktionen implementiert sind. Die Beschaffung der für die Analyse notwendigen Daten ist in eine Sonde ausgelagert, die das Lesen und Übersetzen der Quelltexte übernimmt, sowie die Beschaffung von Kontextinformationen (zum Beispiel Programmbeschreibungen) realisiert. Ergänzt werden Kernsystem und Sonde durch ein ausgelagertes plattformunabhängiges Visualisierungssystem, dass die grafische Aufbereitung der Analyseergebnisse implementiert. Das gesamte System ist vollständig in die SAPGUI integriert, so dass eine konsistente Oberfläche zur Verfügung steht, die keine weiteren Anforderungen am Arbeitsplatz eines Entwicklers stellt. Für eine Bearbeitung der Analyseergebnisse (zum Beispiel zur Dokumentation) können diese konvertiert, auf das Frontend eines Benutzers übertragen und dort mit Hilfe von Office- und Grafikprogrammen weiter aufbereitet und bearbeitet werden.

Fazit

Mit der in diesem Artikel vorgestellte Methode zur Analyse und Visualisierung von ABAP Programmen lassen sich in ABAP realisierte Business Prozesse leichter nachvollziehen und warten. Dort, wo keine oder nur fragmentarische Dokumentation

vorhanden ist, können mit hinreichendem Aufwand Fragen zur Implementierung eines Prozesses beantwortet werden, wie sie die Anpassung eines Prozesses auf neue Randbedingungen erfordern.

Der Magellan Explorer automatisiert den Bottom-Up Analyseprozess und bildet die technische Grundlage für die Analyse von Quelltexten im R/3 System. Mit seiner Hilfe können die entscheidenden Aspekte eines ABAP Quelltextes isoliert und in einer angemessenen Form visualisiert werden. Sein Ziel ist es, den Entwickler bei den täglich anfallenden Wartungsarbeiten zu unterstützen und grundlegende Informationen für die technische Dokumentation eines Programms bereitzustellen.

Innerhalb von SAP R/3 ist damit die technische Grundlage geschaffen worden, um für die Problematik der Analyse und Wartung des Quelltextes einen nachhaltigen und innovativen Lösungsansatz zu präsentieren.

Sebastian Rehbach und Dr. Gunther Schöppe

IXULT AG
Business Technology Group
Mangfallstrasse 37
82026 Rosenheim

Quellen

Balzert, Helmut, Lehrbuch der Software-Technik, Band 1, Heidelberg, Spekturm Akademischer Verlag, 1. Auflage, 1998

Halstead, Maurice Howard, Elements of Software Science, New York, Elsevier North Holland Inc., 2. Auflage, 1978

Müller, Bernd, Reengineering: Eine Einführung, Stuttgart, G. Teubner Verlag, 1. Auflage, 1997

Rehbach, Sebastian, Visualisierung von Programmquelltexten dargestellt am Beispiel der Sprache ABAP, Diplomarbeit im Studiengang Informatik, Fachhochschule Würzburg, 2003

Sneed, Harry M., Softwarewartung und -wiederverwendung, Band 1. Köln, Verlagsgesellschaft Rudolf Müller, 1. Auflage, 1991