Flow Graph Manipulator (FGM) – Reverse Engineering Tool für komplexe Softwaresysteme

Anja Beier, Denis Uhlig
pro et con Innovative Informatikanwendungen GmbH, Annaberger Straße 240, 09125 Chemnitz
anja.beier@proetcon.de, denis.uhlig@proetcon.de

Die erste Phase von Migrationsprojekten besteht im Reverse Engineering der zu migrierenden Software. Das Programmverstehen und die Redokumentation von Legacy-Systemen, die über einen langen Zeitraum historisch gewachsen sind, bilden die Voraussetzung einer erfolgreichen Migration. Das vom Verfasser entwickelte Software-Werkzeug Flow Graph Manipulator (FGM) wird für die Programmanalyse und Redokumentation von Migrationsprojekten eingesetzt. Der vorliegende Beitrag beschreibt die neue Version 3.0, bei der grundlegende Verbesserungen in Architektur, Funktionalität und Benutzeroberfläche gegenüber vorherigen Versionen vorgenommen wurden.

1 Motivation und Ausgangssituation

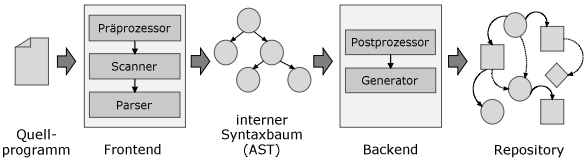
Nachdem FGM bereits seit 10 Jahren kommerziell vertrieben und in verschiedenen Migrationsprojekten - so z.B. bei der Migration von HP NonStop-COBOL nach MF-COBOL oder im Rahmen einer SPL-Konvertierung nach C++ - erfolgreich eingesetzt wurde, ergaben sich Anforderungen an eine Erweiterung der Funktionalität des Werkzeugs. Die bisherige Version ermöglichte vorrangig Daten- und Steuerflussanalysen für einzelne Programmquellen. Neben der detaillierten Betrachtung von Einzelprogrammen ist jedoch die Sicht auf die Gesamtapplikation ein wesentlicher Aspekt beim Reverse Engineering komplexer Systeme. In einer neuen Version sollte deshalb eine Komponente *Applikationswissen* als zentrale Funktionalität integriert werden, welche die Analyse der Systemarchitektur des Softwarepakets in den Vordergrund stellt. Daraus ergab sich gleichzeitig die Notwendigkeit der Konzeption einer neuen Benutzeroberfläche, die sowohl den Kriterien modernen Oberflächendesigns entspricht, als auch die flexible, funktionale Erweiterung des Werkzeugs gewährleistet.

2 Feingranulare Analyse

Die feingranulare Analyse stellt die erste Stufe des Analyseprozesses des FGM dar. Sie dient der Ermittlung quellcodenaher Informationen der analysierten Software auf der Basis eines einzelnen Programms, z.B. der Berechnung von Datenflüssen über bestimmte Programmobjekte.

Zunächst erfolgt eine lexikale, syntaktische und semantische Analyse des Quellcodes und die Abbildung in einen internen Syntaxbaum. Dazu wurden die im Hause pro et con entwickelten Sprachfrontends, so z.B. für COBOL, SPL [Er06] und JAVA in FGM integriert. Ausserdem werden eingebettete Systeme wie SQL, DL/I oder CICS analysiert.

Ein Backend erzeugt aus der internen Darstellung das so genannte FGM-Repository, weches die Objekte und Beziehungen des Programms in einer speziellen Graphnotation darstellt.



Das folgende Beispiel zeigt die Umsetzung einer MOVE-Anweisung in COBOL in die Graphnotation des Repositories:

```
MOVE 10 TO CNT-VALUE.

(cobol_data 6417 CNT-VALUE (loc 2 497 9) …)
(cobol_constant 6420 "10" (loc 2 721 23))
(cobol_move 4024 (loc 2 721 18) (6417) (6420))
```

Das Repository eines durchschnittlichen COBOL-Programms umfasst ca. 10.000 Objekte und 15.000 Beziehungen. Für jedes innerhalb eines Softwareprojekts analysierte Quellprogramm wird ein Repository generiert. Der Zugriff auf den Repository-Inhalt erfolgt über ein FGM-Server-Interface mit Hilfe einer interpretativen Abfragesprache FGM-LANguage.

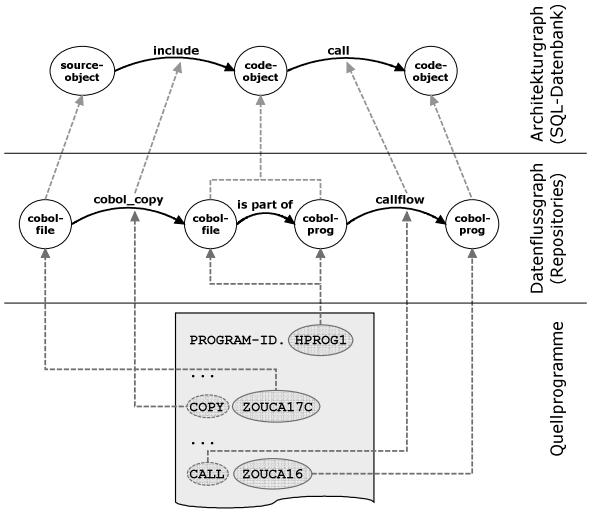
3 Applikationswissen

Aus der Informationsmenge aller Programme lassen sich mittels des Applikationswissens vorrangig Aussagen über die Architektur des analysierten Systems, die Aufruf- und Includebeziehungen zwischen den Programmen sowie den Zugriff auf globale Ressourcen treffen. Zur Gewinnung dieser Informationen ist es erforderlich, von der feingranularen Ebene hin zu einer Architekturebene zu abstrahieren. Im Rahmen der *Architekturanalyse* erfolgt eine Auswertung und Zusammenführung aller erzeugten Repositories hinsichtlich ihrer programmübergreifenden Informationen. Dabei entsteht zunächst ein Objektgraph, welcher alle Objekte des Softwarepakets, z.B. Programme, Dateien und Datenbanken, zusammenfasst. In

nachfolgenden Berechnungsschritten werden aus diesem Basisgraph und den entsprechenden Abfrageergebnissen der Repositoryanalysen weitere Teilgraphen über das Gesamtsystem berechnet. Für die Sprache COBOL sind dies z.B.:

- ein CALL-Graph, der die Aufrufbeziehungen extrahiert,
- ein INCLUDE-Graph zur Darstellung der Nutzungshierarchie und
- ein Datennutzungsgraph, welcher u.a. die Verwendung gemeinsamer Datenfiles und den Zugriff auf SQL-Tabellen beschreibt.

Das Ergebnis der Berechnung dieser Teilgraphen ist ein komplexer Architekturgraph. So entsteht auf der Basis der feingranularen Graphmodelle der Einzelprogramme das Applikationswissen als neue Abstraktionsebene:



Die Objekte und Beziehungen des Architekturgraphen werden in einer SQL-Datenbank abgelegt und verwaltet. So muss die Architekturanalyse nur einmal für das gesamte System erfolgen; alle später neu eingebundenen bzw. modifizierten Quellen werden lediglich in die bestehende Datenmenge integriert.

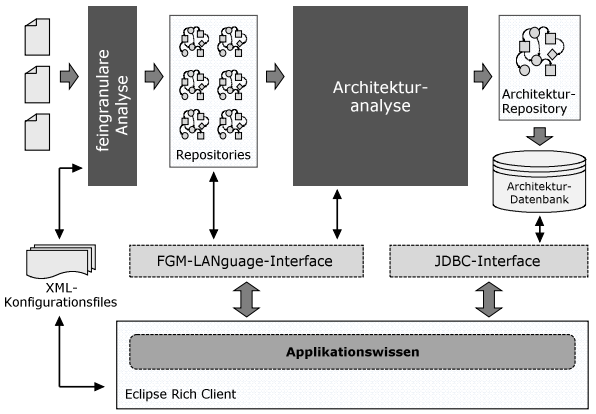
4 Neue Funktionalitäten in der Version 3.0

Die Analyse eines Softwaresystems erfolgt in FGM 3.0 stets ausgehend von der Sicht auf die Gesamtapplikation. Auf der Basis des Applikationswissens lassen sich so z.B. (Teil)Architekturgraphen und Datennutzungsgraphen berechnen und darstellen. Weiterhin ist es möglich, mit Hilfe verschiedener Metriken und Statistikfunktionen u.a. den Wartungsaufwand über das gesamte Softwarepaket oder die Komplexität der Programme zu ermitteln und auf verschiedene Art, z.B. als Säulen- oder Punktdiagramme darzustellen. Darüber hinaus wird - wie bereits in der bisherigen Version - der detaillierte Zugriff auf die Repositories und die Analyse der Steuerflüsse ausgewählter Einzelprogramme realisiert.

Der Anwender ist in der Lage, selbst Skripts in FGM-LANguage zur Gewinnung zusätzlicher Funktionalität zu formulieren. Alle Berechnungen und die Darstellung der gewünschten Resultate werden durch entsprechende Interaktionen mit dem Anwender gesteuert. Die Aufbereitung der Ergebnisse erfolgt tabellarisch und grafisch mit direkter Verbindung zum Quellcode. Dazu wurden u.a. eine dynamische Graphdarstellung sowie ein dialektspezifisches Syntaxhighlighting in die Benutzeroberfläche integriert.

5 Architektur des Werkzeugs

Aufgrund der durch bereits realisierte Projekte gewonnenen Erfahrungen mit der Plattform Eclipse [Uh07] wurde FGM 3.0 in Java als Eclipse Rich-Client-Applikation implementiert. Sämtliche Konfigurationseinstellungen der Benutzeroberfläche und des Analyseprozesses werden über eine XML-Schnittstelle vorgenommen. Dadurch ist z.B. die flexible Einbindung verschiedener Frontends möglich. Die folgende Grafik zeigt die Architektur des Softwarewerkzeugs:



6 Fazit und Ausblick

Die Funktionalität des Applikationswissens wurde erfolgreich realisiert und als Kernkomponente in eine neu konzipierte Benutzeroberfläche des FGM 3.0 eingebunden. Gleichzeitig wurden bewährte Module aus früheren Versionen übernommen. Derzeit existiert eine COBOL-Version, aktuell wird ein bei pro et con entwickeltes Frontend für JAVA 1.6 integriert .

Literatur

[Er06] Erdmenger, U.: SPL-Sprachkonvertierung im Rahmen einer BS2000-Migration.
GI – Softwaretechnik-Trends, Band 26, Heft 2, ISSN 0720-8928, Mai 2006.

[Uh07] Uhlig, D.: Eine Entwicklungsumgebung (IDE) für die BS2000-Migration auf Eclipse-Basis.
GI – Softwaretechnik-Trends, Band 27, Heft 1, ISSN 0720-8928, Februar 2007.