GUPRO

Generische Umgebung zum PROgrammverstehen

Jürgen Ebert, Bernt Kullbach, Thomas Pühler, Andreas Winter

Einleitung

Das Verstehen von umfangreichen Quelltexten sowie die Ermittlung von Informationen daraus ist eine Schlüsselaktivität sowohl in großen Software-Reengineering-Projekten als auch in der alltäglichen Softwarewartung.

In GUPRO wurde zur Unterstützung dieser Tätigkeiten ein konfigurierbares Werkzeug entwickelt. Dabei wurde ein generischer und anfragebasierter Ansatz gewählt, da es im vorhinein nicht möglich ist, alle Anforderungen und somit alle möglichen Anfragen zu erfassen und in das Werkzeug zu integrieren. So hat der spätere Anwender die Möglichkeit, über beliebige Anfragen die gerade von ihm benötigten Informationen zu erfragen und ist nicht durch Vorgaben des Werkzeugs eingeschränkt. Mittels der Generizität ist es ermöglich, Anfragen an beliebige, frei wählbare Problembereiche (z.B. Programmiersprachen) mit derselben Anfragesprache aus der dergleichen Oberfläche zu stellen. [EBERT et al. 1998]

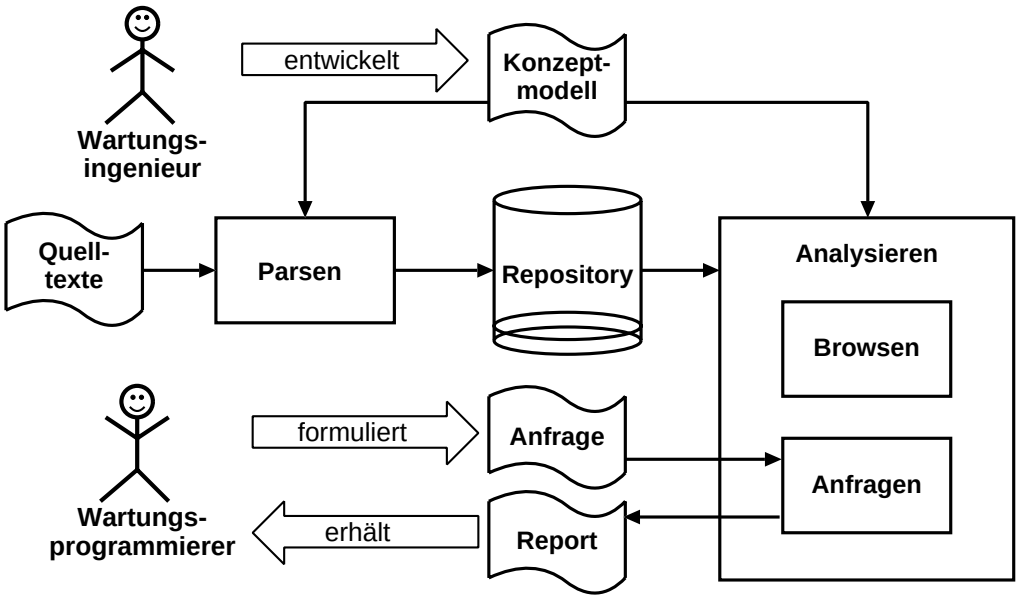

Abbildung 1: GUPRO-Architekturschema

Architektur

Die Umsetzung dieses Ansatzes soll durch das datenflußartige Architekturschema in Abbildung 1, das gleichzeitig die Tätigkeiten in GUPRO beschreibt, verdeutlicht werden. Dieses idealisiert die einzelnen Phasen der Werkzeugerstellung bzw. -verwendung. Die zentrale Tätigkeit ist die Entwicklung eines Konzeptmodells, das den zu analysierenden Problembereich auf einem in dem jeweiligen Kontext angemessenen Abstraktionsniveau beschreibt. Zur Beschreibung dieses Modells werden erweiterte Entity-Relationship-Diagramme (EER) mit zusätzlichen Integritätsbedingungen in der Sprache GRAL (GRaph Specification Language, eine Z-ähnliche Spezifikationssprache) verwendet. [EBERT et al. 1996], [FRANZKE 1997] Das zu dem Modell gehörende Repository ist vollständig graph-basiert. Zu diesen Konzeptmodellen werden mittels eines Generators Parser erzeugt, die zu untersuchende Quelltexte in das Repository überführen.

[DAHM 1998] Das Repository besteht aus typisierten, attributierten, angeordneten und gerichteten Graphen (TGraphen) [DAHM et al. 1998], deren Struktur durch das Konzeptmodell beschrieben wird und die durch die Parser erzeugt werden. Für die Realisierung der TGraphen wird die leistungsfähige Klassenbibliothek „Graphenlabor“ verwendet. [DAHM und WIDMANN 1998]

Mittels der Analysekomponente können beliebige Anfragen an ein Repository gestellt werden. Dabei ist allerdings Kenntnis der Struktur des Repositories, d.h. des Konzeptmodells, nötig, da nur die Zusammenhänge erfragt werden können, die auch im Repository enthalten sind. Zur Anfrage wurde eine auf GRAL aufbauende Sprache, GReQL (GUPRO Repository Query Language) [KAMP 1998a], entwickelt, die speziell auf die Graph-Struktur des Repositories optimiert ist. Für die interne Repräsentation der Anfragen werden ebenfalls wieder TGraphen verwendet. Um die Anfragen auswerten zu können, wurden neben dem GReQL-Parser auch ein Auswerter [DAHM 1997] sowie zur Performanzsteigerung ein Optimierer [POLOCK 1997] entwickelt. Die Ergebnisse einer Anfrage werden dem Benutzer in tabellarischer Form analog gängigen Tabellenkalkulationen (z.B. MS Excel) optisch aufbereitet angezeigt.

Neben der Anfragekomponente besitzt das GUPRO-Werkzeug auch eine Browsing-Komponente. Mit dieser kann der Anwender sich Quelltextauschnitte zu Anfrageergebnissen anzeigen lassen. So wird die Quelle des Ergebnisses und auch der umgebende Quelltext direkt angezeigt, um das Verständis zu maximieren. Außerdem ist es möglich, direkt innerhalb des Quelltextes und auch über mehrere Quelltextgrenzen hinaus zu navigieren, wodurch sich Fragen über das Programm direkt durch eine Untersuchung des Quelltextes beantworten lassen.

Instanzen

Basierend auf diesem Ansatz wurden in mehreren Projekten Instanzen, die die Leistungsfähigkeit dieses Ansatzes gezeigt haben, realisiert. Dabei bestand die Realisierung aus der Entwicklung und Definition eines Konzeptmodells und der Generierung von entsprechenden Parsern für dieses Konzeptmodell. Die nur einmal realisierte generische Analysekomponente konnte für alle Instanzen ohne Anpassungen direkt verwendet werden.

Konkret wurden bisher die folgenden vier Instanzen realisiert:

MAKRO Dieses Konzeptmodell entstand in einem Projekt in Zusammenarbeit mit den Projektpartnern IBM und Volksfürsorge Unternehmensgruppe, das vom Bundesministerium für Bildung, Forschung und Technologie gefördert wurde (Förderkennzahl 01 IS 504). [EBERT et al. 1998] Mit dem Schema MAKRO wurde die Anwendungslandschaft der Volksfürsorge modelliert, die aus diversen Programmiersprachen (z.B. Cobol, Csp, JCL) und Datenbanken (z.B. PSB, IMS, relationale DB) bestand. [KULLBACH et al. 1998] Dadurch wurde es möglich, nicht nur Informationen über einzelne Programme, sondern auch über Zusammenhänge über Sprachgrenzen hinweg zu analysisieren. Der Fokus bei diesem Modell wurde nicht auf die feingranulare Modellierung der einzelnen Sprachen gelegt, sondern auf Wunsch der Projektpartner auf die Zusammenhänge zwischen den einzelnen Programmen und Modulen (grobgranulare Sicht). Dabei wurden alle Programme unabhängig von der Programmiersprache in ein gemeinsames Repository überführt. [KAMP 1998b]

C Die feingranulare Analyse von C-Programmen wird mit diesem Konzeptmodell ermöglicht, d.h. die Auflösung geht bis auf Bezeichner- und Ausdrucksniveau herab (feingranular). Auch hier ist es möglich, mehrere Quellen (C-Programme) in ein Repository einzufügen und so Zusammenhänge über Quelltextgrenzen hinaus abzufragen.

CPDG Hierbei handelt es sich ebenfalls um ein Schema, das C-Programme beschreibt. Über die beiden vorherigen Modelle hinaus werden hier auch der Kontroll- und der Datenfluss innerhalb der Programme dargestellt (PDG, Program Dependency Graph). Dieses Modell wird z.Zt. für die Sprache C++ in einem Projekt zur Anwendung in der Sicherheitsüberrüfung mit dem Bundesamt für Sicherheit in der Informationstechnik (BSI) erweitert.

METAMETA Dieses Schema beschreibt die Struktur der Konzeptmodelle. Es handelt sich somit um das Meta-Metamodell. Es dient zu Analyse und Abfrage der Konzeptmodelle. Damit es ist möglich, auch die dem Werkzeug zugrundeliegende Konzeptmodelle zu analysieren, und demonstriert so die Leistungsfähigkeit des Werkzeugs.

Der Fokus der bisher realisierten GUPRO-Instanzen lag insbesondere auf der Untersuchung von Quelltexten mittels GReQL-Anfragen und durch Inspektion beim Browsen. Auf diese Anfragemöglichkeiten können auch weitere Komponenten zur Quelltextanalyse aufgesetzt werden. [KULLBACH and WINTER 1999] Zur Zeit werden diese Möglichkeiten neben der direkten Quelltextinspektion durch Anfragen auch zur Informationsgewinnung während des Parsens eingesetzt.[KAMP 1998b] Ebenfalls wurde ein auf dem Anfrageansatz basierendes Werkzeug zur konsistenten Quelltextmodifikation entwickelt.[KULLBACH et al. 1998] In weiteren Arbeiten wird dieser Anfrageansatz auf Komponenten zur Berechnung von Program-Slices und zur Berechnung von Metriken übertragen.

Weitere Auskünfte geben die im folgenden Literarturverzeichnis aufgeführten Veröffentlichungen, die neben weiteren Informationen unter http://www.uni-koblenz.de/ ist/gupro.html abgerufen werden können.

Literatur

[DAHM 1997] DAHM, P. (1997). *Architektur des GReQL–Auswerters*. Projektbericht 11/97, Universität Koblenz-Landau, Institut für Softwaretechnik, Koblenz.

[DAHM 1998] DAHM, P. (1998). *Parser Description Language — An Overview*. In EBERT, J., R. GIMNICH, H. H. STASCH, and A. WINTER, eds.: *GUPRO — Generische Umgebung zum Programmverstehen*, pp. 137–156.

[DAHM et al. 1998] DAHM, P., J. EBERT, A. FRANZKE, M. KAMP, and A. WINTER (1998). *TGraphen und EER-Schemata — Formale Grundlagen*. In EBERT, J., R. GIMNICH, H. H. STASCH, and A. WINTER, eds.: *GUPRO — Generische Umgebung zum Programmverstehen*, pp. 51–66.

[DAHM und WIDMANN 1998] DAHM, P. und F. WIDMANN (1998). *Das Graphenlabor*. Fachberichte Informatik 11/98, Universität Koblenz-Landau, Institut für Informatik, Koblenz.

[EBERT et al. 1998] EBERT, J., R. GIMNICH, H. H. STASCH und A. WINTER, Hrsg. (1998). *GUPRO — Generische Umgebung zum Programmverstehen*. Fölbach, Koblenz.

[EBERT et al. 1996] EBERT, J., A. WINTER, P. DAHM, A. FRANZKE, and R. SÜTTENBACH (1996). *Graph Based Modeling and Implementation with EER/GRAL*. In THALHEIM, B., ed.: *15th International Conference on Conceptual Modeling (ER'96), Proceedings*, no. 1157 in *LNCS*, pp. 163–178, Berlin. Springer.

[FRANZKE 1997] FRANZKE, A. (1997). *GRAL: A Reference Manual*. Fachbericht Informatik 3/97, Universität Koblenz-Landau, Fachbereich Informatik, Koblenz.

[KAMP 1998a] KAMP, M. (1998a). *GReQL - eine Anfragesprache für das GUPRO–Repository*. In: EBERT, J., R. GIMNICH, H. H. STASCH und A. WINTER, Hrsg.: *GUPRO — Generische Umgebung zum Programmverstehen*, S. 173–202.

[KAMP 1998b] KAMP, MANFRED (1998b). *Managing a Multi-File, Multi-Language Software Repository for Program Comprehension Tools – A Generic Approach*. In CARLINI, U. DE and P. K. LINOS, eds.: *6th International Workshop on Program Comprehension*, pp. 64–71, Washington. IEEE Computer Society.

[KULLBACH et al. 1998] KULLBACH, B., A. WINTER, P. DAHM, and J. EBERT (1998). *Program Comprehension in Multi-Language Systems*. In *Proceedings of the 5th Working Conference on Reverse Engineering 1998 (WCRE '98)*, pp. 135–143, Los Alamitos. IEEE Computer Society.

[KULLBACH and WINTER 1999] KULLBACH, BERNT and A. WINTER (1999). *Querying as an Enabling Technology in Software Reengineering*. In VERHOEF, C. and P. NESI, eds.: *Proceedings of the 3rd Euromicro Conference on Software Maintenance & Reengineering*, pp. 42–50, Los Alamitos. IEEE Computer Society.

[POLOCK 1997] POLOCK, D. (1997). *Ein statischer Optimierer für GRAL- und GReQL-Ausdrücke*. Diplomarbeit D 414, Universität Koblenz-Landau, Fachbereich Informatik, Koblenz.