Statische Extraktion von Protokollen

Gunther Vogel
Universität Stuttgart
Institut für Softwaretechnologie
Universitätsstraße 38
D-70569 Stuttgart

21. April 2006

Zusammenfassung

Der vorliegende Artikel stellt Möglichkeiten zur statischen Analyse von Programmabläufen vor. Die Regeln und Konventionen dieser Abläufe werden unter dem Begriff des Protokolls zusammengefasst.

1 Einführung

Die Auswirkungen von fehlerhaften Programmabläufen reichen von Fehlern in Berechnungen und falschen Ausgaben bis zum Einfrieren oder Abstürzen des Programms. Bekannt und gefürchtet sind unter anderem Verwendungen von uninitialisierten Variablen oder Zugriffe auf bereits freigegebenen Speicher. Fehlerhafte Synchronisierung von parallelen Zugriffen durch Threads auf gemeinsame Datenobjekte führen zu Race Conditions oder Deadlocks. Fehler in Abläufen sind jedoch nicht nur im Kleinen, sondern auf allen Abstraktionsebenen von Programmen zu finden. Gerade bei der komponentenbasierten Software-Entwicklung sind komplexe Interaktionen zwischen Komponenten zu beachten. Trotzdem werden diese selten ausreichend dokumentiert. Über die im Folgenden beschriebenen Techniken können diese Zusammenhänge aus dem Quelltext extrahiert werden.

Bei einem Protokoll handelt es sich hierbei um eine Menge von Konventionen und Regeln für die Abläufe innerhalb von Programmen. Diese Protokolle beschreiben beispielsweise die erlaubten Reihenfolgen von Zugriffen auf Variablen, Einschränkungen bei der Verwendung von Unterprogrammen oder die Interaktionen zwischen Software-Komponenten. Ein Teil des Protokolls besteht dabei immer aus syntaktisch-semantischen Beschränkungen, die von der Schnittstelle einer Komponente vorgegeben werden, wie zum Beispiel die Signatur von Unterprogrammen oder des Typs von Variablen. Auch die dynamischen Aspekte der Abläufe können über eine statische Analyse extrahiert werden. Unter anderem ergeben sich damit die folgenden Anwendungsmöglichkeiten:

Dokumentation Selten werden wichtige Informationen über die Abläufe und Zusammenhänge in der Software ausreichend dokumentiert. Oftmals ist die Dokumentation auch nur veraltet. Die statische Extraktion von Protokollen gibt immer den aktuellen Stand wieder, so dass sich Fragen bei der Software-Wartung präzise beantworten lassen.

Reflektion Die extrahierten Protokolle stellen die Zusammenhänge im System bezüglich eines bestimmten Aspekts dar. Werden bei der Inspektion Abweichungen zu den erwarteten Ergebnissen festgestellt, müssen die eigenen Vorstellungen gegebenenfalls angepasst werden. Ein besseres Verständnis der Software wird erzielt.

Fehlersuche Bei Abweichungen kann es sich nicht nur um mangelndes Verständnis sondern auch um Hinweise auf Fehler handeln, denen nachgegangen werden muss. Auch werden durch die Inspektion Unzulänglichkeiten, wie eine ineffiziente Implementierung, erkannt und können dann verbessert werden.

Verifikation Das Endziel bei der Beschäftigung mit Protokollen besteht aus einer automatischen Prüfung der

Korrektheit von Abläufen. Hierzu notwendig ist, dass sowohl das *erwartete* als auch das *verwendete* Protokoll formalisiert werden und somit ein automatischer Abgleich ermöglicht wird. Zur Extraktion beider Protokolle auf Basis des Quelltextes stehen Werkzeuge und Techniken zur Verfügung, die im folgenden Abschnitt kurz beschrieben werden.

2 Extraktion

Prinzipiell kann eine Extraktion von Protokollen aus dem Quelltext auf zwei verschiedene Weisen durchgeführt werden. Bei einem *Glass-Box* Verfahren werden die internen Zusammenhänge einer Komponente analysiert, um die Einschränkungen für die Verwendung herzuleiten. Ein Beispiel für ein solches Verfahren findet sich in [2].

Bei einem *Black-Box* Verfahren werden im Gegensatz dazu keine Interna von Komponenten analysiert. Der Quelltext für diese Komponenten muss also auch gar nicht vorliegen. Statt dessen wird lediglich die Verwendung der Komponente untersucht. Aus korrekten Verwendungen wird dann auf das *erwartete* Protokoll geschlossen. Oftmals gestaltet sich dieses Vorgehen einfacher als das, des *Glass-Box* Verfahrens. Es erfordert jedoch auch eine Eingriffsmöglichkeit für den Benutzer, da dieser Verwendungen als korrekt klassifizieren muss.

Bei *Spurgraphen* handelt es sich um interprozedurale Kontrollflussgraphen, die die dynamischen Aspekte des *verwendeten* Protokolls wiedergeben. Beispielsweise wurden in [1] statische Objektspurgraphen vorgestellt. Diese beschreiben die möglichen Reihenfolgen von Operationen bezüglich eines Objekts. Ein Spurgraph entsteht durch Transformation des ursprünglichen Kontrollflussgraphen des Programms. Es sind hierzu die folgenden zwei Teilprobleme zu lösen.

Filterung In einem ersten Schritt werden die, für den aktuellen Aspekt relevanten Knoten des zu Grunde liegenden Kontrollflussgraphen bestimmt. Für diese werden dann Knoten im Spurgraphen erzeugt. Beispiele für diese Knoten sind solche, die Zugriffe auf ein bestimmtes Objekt oder eine Komponente darstellen. Dabei kann es auch vorkommen, dass für manche Knoten im Kontrollflussgraphen mehrere Knoten erzeugt werden müssen, da sie unter verschiedenen Kontexten ausgeführt werden. Beispielsweise kann bei einem Kopiervorgang das selbe Objekt einmal als Ziel und einmal als Quelle herangezogen werden. Um die Struktur des relevanten Kontrollflusses nachzubilden, können optional weitere Knoten für Verzweigungen und Unterprogrammaufrufe erzeugt werden.

Verbindung Die getrennt voneinander erstellten Knoten im Spurgraphen müssen im zweiten Schritt über Kontrollfluss-, Aufrufs-, und Rückkehrkanten gemäß dem ursprünglichen Kontrollflussgraphen miteinander verbunden werden. Hierbei handelt es sich um ein ähnliches Problem wie das der First- und Follow-Mengen kontextfreier Grammatiken. Zur Lösung stehen effiziente Algorithmen zur Verfügung.

Zur Weiterverarbeitung können weitere Transformationen auf den Spurgraphen durchgeführt werden. Spurgraphen, die Teile eines gemeinsamen Aspekts darstellen, können kombiniert werden. Dies ist zum Beispiel notwendig, wenn überprüft werden soll, dass während der Traversierung einer Datenstruktur mittels eines Iterators keine Änderungen an der Datenstruktur vorgenommen werden.

Für die automatische Verifikation von Protokollen können die Spurgraphen leicht in endliche Automaten transformiert werden. Ob eine Verwendung ein gefordertes Protokoll einhält, kann dann durch den Vergleich zweier Automaten überprüft werden.

3 Fazit

Die statische Analyse bietet mächtige Werkzeuge für die Extraktion und Verifikation von Protokollen. Neue Möglichkeiten eröffnen sich durch semi-automatische Analysetechniken, die durch das Einbeziehen von Benutzern bessere Ergebnisse erzielen.

Literatur

[1] Thomas Eisenbarth, Rainer Koschke, and Gunther Vogel. Static Object Trace Extraction for Programs with Pointers. *Journals of Systems and Software*, 2005.

[2] John Whaley, Michael C. Martin, and Monica S. Lam. Automatic extraction of object-oriented component interfaces. In *Proceedings of the International Symposium on Software Testing and Analysis*, July 2002.