ECARES-Projekt: Kombination von graphbasierten Redesign-Analysen

Christof Mosler, Tobias Walter
[mosler|twalter]@i3.informatik.rwth-aachen.de
http://www.se.rwth-aachen.de
Lehrstuhl für Informatik 3 (Softwaretechnik), RWTH Aachen

Abstract: Im ECARES-Projekt werden Methoden und Konzepte zur Unterstützung des graphbasierten Reengineerings von Telekommunikationssystemen entwickelt. Neben dem Reverse Engineering werden für verschiedene Anwendungsszenarien auch konkrete Redesign-Methoden entwickelt, die zur Veränderung der Systemsoftware führen. Hierfür sind Analysen notwendig, die dem Benutzer entsprechende Redesign-Vorschläge liefern. Diese werden in datenorientierte, funktionsorientierte oder anwendungsspezifische Ansätze unterteilt, die jeweils unterschiedliche Vorteile bieten. Dieses Papier zeigt, wie diese verschiedenen Redesign-Analysen flexibel miteinander kombiniert werden können, um insgesamt zu besseren Vorschlägen zu gelangen.

1 Einführung

Das ECARES-Projekt (**E**ricsson **C**ommunication **AR**chitecture for **E**mbedded **S**ystems) ist eine Kooperation zwischen der Firma Ericsson und dem Lehrstuhl für Informatik 3 der RWTH Aachen. Mit der Kooperation wird das Ziel verfolgt, Methoden, Konzepte und Werkzeuge zu entwickeln, die das Verständnis der Software von Telekommunikationssystemen fördern und die Restrukturierung dieser unterstützten sollen. Schwerpunkt der Untersuchung bildet die Software von Ericssons Mobile Switching Center AXE10 [Mar04].

Um die Software des AXE10 zu realisieren, führte Ericsson im Jahr 1970 die hauseigene **P**rogramming **L**anguage for **EX**change (PLEX) ein [Mar04]. PLEX-Systeme sind in funktionale Einheiten unterteilt, die im Folgenden Blöcke genannt werden. Es handelt sich um eine imperative, modular aufgebaute Programmiersprache, die auf einem Signalparadigma basiert, d.h. Blöcke kommunizieren miteinander über Signale.

Im ECARES-Projekt [FMP06] werden alle drei Phasen des Reengineering-Prozesses unterstützt: Bei der Abstraktion wird beim *Reverse Engineering* aus den Textdokumenten (Quellcode, Traces, etc.) eine Graphstruktur erzeugt - im Folgenden als Strukturgraph bezeichnet - die alle relevanten Informationen zum Daten-, Kontroll- und Signalfluss enthält. Auf dieser abstrakten Repräsentation kann der Benutzer Zusammenhänge im System einfacher nachvollziehen und auf anschauliche Weise *Modifikationen* vornehmen. Nach den Modifikationen wird im *Forward Engineering* aus dem Strukturgraph wieder ein lauffähiger Quellcode generiert. In Abb. 1 ist die im Projekt entwickelte Reengineering-Umgebung RePLEX dargestellt.

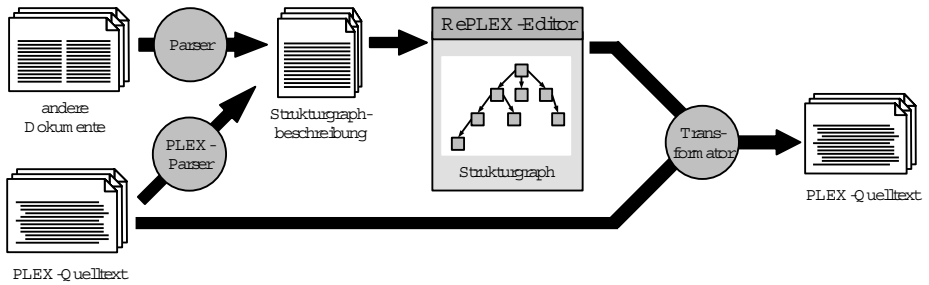

Abbildung 1: Die Reengineering-Umgebung RePLEX einschließlich der verwendeten Dokumente

Zeitkritische Telekommunikationssysteme weisen oft Hardware-bedingte Restriktionen auf, die beim Software-Reengineering zu berücksichtigen sind. So können beispielsweise PLEX-Blöcke des AXE10-Systems nur eine bestimmte Größe aufweisen. Durch neue Standards und Richtlinien wird das Mobilfunknetz und damit auch die Software stetig erweitert, so dass die Blockgrößen mit der Zeit an diese Grenzen stoßen. Hier liegt die Lösung in einer Aufteilung eines solchen Blocks in zwei oder mehrere kleine Blöcke (*Blockdekomposition*). Da diese Arbeit bei Ericsson bis heute manuell durchgeführt wird, wurde im ECARES-Projekt nach Möglichkeiten der Unterstützung des Aufteilungsprozesses gesucht. Das Ziel sollte sein, die Kopplung zwischen zwei Teilblöcken minimal zu halten und gleichzeitig die Bindung einzelner Programmartefakte in einem Teilblock zu maximieren. Die drei in diesem Papier beschriebenen Verfahren sowie ihre Kombination beziehen sich auf dieses spezielle Szenario.

In Abb. 2 ist ein Ausschnitt eines Strukturgraphen dargestellt. Die Knoten des Graphen repräsentieren die Programmartefakte, aus denen ein Block besteht. Es gibt zum Beispiel Signaleingänge, Anweisungsfolgen, Subroutinen und Datenelemente. Die Beziehungen zwischen den Knoten werden durch Kanten dargestellt. Neben strukturellen Kanten (*Has*- und *Contains*-Kanten), die den Aufbau eines Blockes verdeutlichen, werden durch semantische Kanten (z.B. *Goto*-, *Call*-, *Modifies*- oder *Reads*-Kanten) unbedingte Sprünge, Aufrufe von Subroutinen und Zugriffe auf Datenelemente beschrieben. All diese Graphelemente beinhalten mehrere Attributen, die hier aber nicht dargestellt sind.

2 Kombination von Analysen für das Redesign

Für das Redesign von Blöcken existieren verschiedene Analysen mit unterschiedlichen Schwerpunkten. Wir unterscheiden zwischen datenorientierten, funktionsorientierten und anwendungsspezifischen Analysen. Ergebnis jeder Analyse ist eine Gewichtung aller semantischen Kanten im Strukturgraph mit normierten Werten von 0 bis 1. Diese Gemeinsamkeit ist notwendig, um sie später kombinieren zu können. Für den Benutzer des RePLEX-Werkzeuges ist es dabei jederzeit möglich, die Kanten des Strukturgraphen interaktiv nachzugewichten, um so durch sein Fachwissen das Clustering beeinflussen zu können.

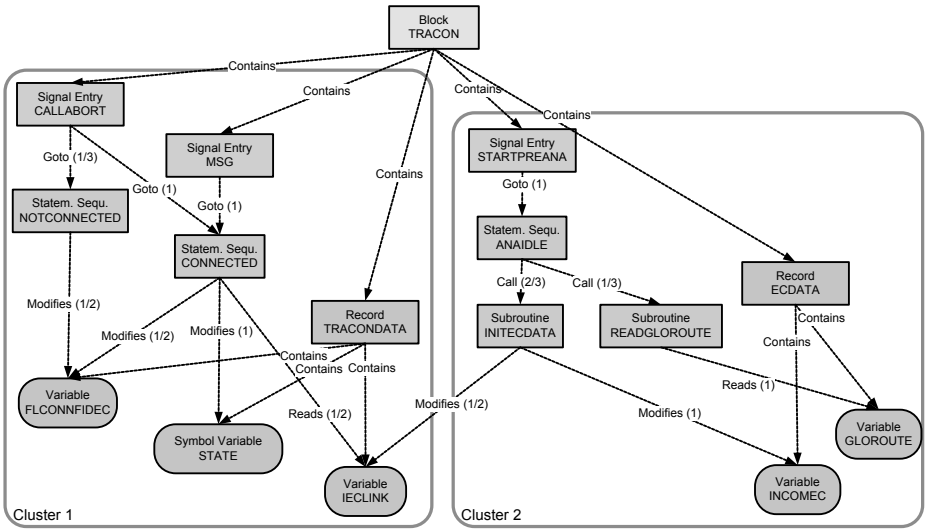

Abbildung 2: Datenorientiert gewichteter Strukturgraph

Datenorientierte Analysen verfolgen das Ziel, auf Grundlage von Datenelementen oder -strukturen zu entscheiden, welche Programmteile eines Blockes stark zusammenhängen. Die datenorientierte Gewichtung der Kante zwischen a und b ist das Verhältnis der Anzahl von Datenelementen im Strukturgraph, die von beiden Knoten im Laufe aller Kontrollflüsse gemeinsam genutzt werden und der Anzahl von Datenelementen, die von beiden zusammen oder getrennt benutzt werden. Die Funktion $data(v)$ liefert für einen beliebigen Knoten v eine Menge von Datenelementen, auf die von allen aus v ausgehenden Kontrollflüssen zugegriffen wird. Für die datenorientierte Gewichtung gilt: $Sim_{data}(a,b) = \frac{|data(a) \cap data(b)|}{|data(a) \cup data(b)|}$. Man erkennt, dass die Gewichtung $Sim_{data}(a,b)$ zwischen 0 und 1 liegt. Benutzen die Knoten a und b keine gemeinsamen Daten, so ist $|data(a) \cap data(b)|$ gleich 0 und entsprechend gilt auch $Sim_{data}(a,b) = 0$. Die Bindung der Knoten a und b ist minimal, da sie in Bezug auf die Daten keine Gemeinsamkeiten besitzen. Nutzen beide Knoten a und b die selben Daten, so ergibt sich offensichtlich die Gewichtung $Sim_{data}(a,b) = 1$, was auf ein starke, datenorientierte Bindung der Knoten hinweist. In Abb. 2 sind hinter den Kantenbezeichnug in Klammern deren datenorientierte Gewichtung angegeben. Man erkennt, dass die Gewichtung der Kante zwischen den Knoten *CALLABORT* und *NOTCONNECTED* $\frac{1}{3}$ ist. Gemeinsam nutzen beide Knoten unter Betrachtung aller ausgehenden Kontrollflüsse nur das Datenelement *FLCONNFIDEC*, der Knoten *CALLABORT* dagegen nutzt alle Elemente *FLCONNFIDEC*, *STATE* und *IECLINK*. Durch die Anwendung des datenorientierten Ansatzes wird vermieden, dass nach der Dekomposition auf unnötig viele Variablen über die Blockgrenzen hinweg zugegriffen wird.

Funktionsorientierte Analysen orientieren sich an Schnittstellenbereichen in Form von Signaleingängen eines Blockes. Ausgehend von speziellen Signalen, die ein Block empfängt,

werden bestimmte funktionale Bereiche im Block aufgerufen. Die funktionsorientierte Gewichtung einer Kante zwischen zwei Knoten a und b entsteht aus dem Verhältnis der Anzahl von Signaleingängen, über die beide Knoten gemeinsam erreichbar sind und der Anzahl von Signaleingängen, über die beide zusammen oder getrennt erreichbar sind. Betrachtet werden hier die Kontrollflüsse, die ausgehend von den Signaleingängen zu den Knoten a und b laufen. Die Funktion $signal(v)$ gibt für einen Knoten v eine Menge von Signaleingängen zurückt, die jeweils alle einen Kontrollfluss besitzen, der zum Knoten v führt. Für die funktionsorientierte Gewichtung gilt: $Sim_{func}(a,b) = \frac{|signal(a) \cap signal(b)|}{|signal(a) \cup signal(b)|}$. Bei der Analyse liefert $Sim_{func}(a,b)$ einen Wert zwischen 0 und 1. Fließen von denselben Signaleingängen jeweils Kontrollflüsse zu den Knoten a und b, so ist $Sim_{func}(a,b)$ gleich 1, was auf eine starke funktionale Bindung hindeutet. Gilt $Sim_{func}(a,b) = 0$, so existieren keine Signaleingänge, deren ausgehende Kontrollflüsse zu beiden Knoten a und b führen. Dies deutet auf eine schwache Bindung der Knoten a und b hin. So können beim funktionsorientierten Ansatz funktional zusammenhängende Programmbereiche als solche erkannt und berücksichtigt werden.

Anwendungsspezifische Analysen basieren auf Mustern, die man auf Quellcode- und Entwurfsebene findet. Ziel dieser Analyse ist es, Muster beim Redesign als unzertrennliche Einheiten im Graph zu betrachten. Bei diesen Analysen werden mit Hilfe von Pattern Matching im Graph Muster gesucht und alle induzierten Kanten dieser mit dem Wert 1 gewichtet. So wird die starke Bindung der Knoten innerhalb eines Musters ausgedrückt. Alle übrigen Kanten, die keinem Muster zugeordnet werden konnten, erhalten die Gewichtung 0. Auf diese Weise können Restrukturierungen vermieden werden, die entweder technisch nur schwer möglich sind oder aber die Performanz des Systems verschlechtern.

Wünschenswert ist es, die Vorteile der drei Verfahren gemeinsam nutzen und kombinieren zu können, was erfahrungsgemäß zu den besten Ergebnissen führt. Analysen können technisch miteinander verknüpft werden, indem eine neue, kombinierte Analyse aus den Implementierungen der bereits existierenden neu erstellt wird. Eine weitere Möglichkeit besteht darin, die Ergebnisse der Analysen und nicht die Techniken miteinander zu kombinieren. Letztere Variante, wie auch in [Kos00] vorgeschlagen, ist wesentlich flexibler.

Die Komposition einzelner Analysen findet durch Überlagerung der zuvor beschriebenen Gewichtungen statt. Damit die Bindung einzelner Knoten weiterhin verglichen werden kann, soll trotz Überlagerung die Gewichtung einer Kante maximal 1 und minimal 0 betragen. Dies kann man beispielsweise durch die Einführung einer zusätzlichen natürlichen Zahl n erreichen, in der die Anzahl der überlagerten Gewichtungen gespeichert wird. Diese ist zu Beginn 1 und wird pro Überlagerung um 1 erhöht. Zur Kombination von Analysen verwendet man einen Kompositionsoperator *composition*, der zwei Gewichtungen Sim' und Sim'', zwei Knoten a und b und den Parameter n entgegen nimmt. Sim' und Sim'' gewichten die direkte Verbindung zwischen a und b und sollen mit Hilfe des Operators überlagert werden. Ergebnis der Anwendung des Operators ist eine neue Gewichtung Sim_{new}, die ebenfalls die direkte Verbindung zwischen a und b bewertet. Die Komposition zweier Gewichtungen Sim' und Sim'' wird dann folgendermaßen berechnet:

$$Sim_{new}(a,b) := composition\,(Sim', Sim'', a, b, n) := \frac{Sim'(a,b) + Sim''(a,b)}{n+1}$$

Um am Ende die Dekomposition eines Blockes zu realisieren, nutzt man einen Clusteralgorithmus, der auf dem mehrfach gewichteten Strukturgraph arbeitet. Hierfür kann ein hierarchischer, agglomerativer Algorithmus verwendet werden, der mit einem direkten Ähnlichkeitsmaß arbeitet. Die Aufgabe des Clusteralgorithmus ist es, solange die Cluster mit maximaler Ähnlichkeit zu vereinigen, bis der Block in die gewünschte Anzahl von Clustern zerlegt wurde (z.B. zwei Cluster in Abb. 2). Die daraus resultierenden Teilblöcke sollen eine minimale Kopplung besitzen, so dass nur wenige Daten in Form von Signalen zwischen ihnen ausgetauscht werden.

Das Reengineering-Szenario der Blockdekomposition bildet zwar ein sehr spezielles Problem, die verwendeten Grundtechniken findet man aber beispielsweise beim Architecture Recovery wieder, wo Softwarearchitekturen durch Clustertechniken rekonstruiert werden. In [Kos00] werden neben verwandeten Basistechniken (Liu&Wilde, Cimitile&Visaggio, etc.) auch Operatoren zur Komposition solcher Techniken vorgestellt. Diese arbeiten allerdings nicht direkt auf Graphen. Mit kleinen Anpassungen und entsprechend definierten Ähnlichkeitsmaßen konnten mit unserer Methode nach Validierung der Ericsson-Experten sehr gute Redesign-Vorschläge für das Blockdekompositionsszenario entwickelt werden.

3 Zusammenfassung

In diesem Papier wurden für das vorgestellte Reengineering-Szenario der Blockdekomposition drei gewichtungsbasierte Redesign-Analysen vorgestellt sowie ein Ansatz präsentiert, wie man ihre Ergebnisse kombinieren kann. Die abschließende Berechnung der Blockaufteilung wird von einem Clusteralgorithmus durchgeführt. Da prinzipiell nicht die Methoden sondern ihre Ergebnisse kombiniert werden, können auch andere Methoden zur Kantengewichtung herangezogen werden, so dass der Ansatz auch für andere Reengineering-Szenarien verallgemeinerbar ist. Der Ansatz wurde bereits im RePLEX-Werkzeug realisiert. Anhand einer Fallstudie konnte gezeigt werden, dass durch die Kombination der drei Gewichtungsmethoden verbesserte Redesign-Vorschläge erzeugt werden können.

Literatur

[FMP06] Christian Fuss, Christof Mosler und Marcel Pettau. RePLEX: A Model-Based Reengineering Tool for PLEX Telecommunication Systems. *Electronic Communications of the EASST*, 1, 2006. Homepage: http://www.easst.org/eceasst/.

[Kos00] Rainer Koschke. *Atomic Architectural Component Recovery for Program Understanding and Evolution*. Dissertation. Institut für Informatik, Universität Stuttgart, 2000.

[Mar04] André Marburger. *Reverse Engineering of Complex Legacy Telecommunication Systems*. Dissertation. Shaker Verlag, Aachen, Germany, 2004. ISBN 3-8322-4154-X.