

Dynamische Komponenten-Programm-Visualisierung

Ludger Martin*

Anke Giesl*

Johannes Martin⁺

*Technische Universität Darmstadt
Fachbereich Informatik
Wilhelminenstr. 7

64283 Darmstadt, Deutschland

{lumartin—giesl}@gkec.informatik.tu-darmstadt.de

⁺University of Victoria

Department of Computer Science

PO Box 3055

Victoria, BC V8W 3P6, Canada

jmartin@csr.csc.uvic.ca

Abstrakt

Dynamische Programm-Visualisierung, d.h. die Visualisierung des Laufzeitverhaltens von Programmen im Gegensatz zu der statischen Untersuchung von Quelltext, ist keine neue Methode. Seit langem wird sie an verschiedenen Arten von Programmen (prozedural, objekt-orientiert usw.) angewandt.

In diesem Papier untersuchen wir dynamische Programm-Visualisierung für Komponentenprogramme. Wir diskutieren, wie die Kommunikation zwischen Komponenten visualisiert werden kann. Wir schlagen eine dreidimensionale Darstellung vor und zeigen die Integration in die HOTAGENT Komponenten-Entwicklungsumgebung.

1 Einleitung

Komponenten-Technologie wird immer häufiger für die Entwicklung von verschiedenen Arten von Programmen verwendet. Johnson [Joh97] erklärt, daß die Verwendung von Komponenten die Programmierung sehr effizient macht. Die Anwendungen sind leichter zu erstellen, zuverlässiger und einfacher zu warten.

Programm-Analyse dient dem Verstehen und Erforschen der Struktur und des Verhaltens von Programmen. Es gibt zwei verschiedene Ansätze für die Analyse: statische Analyse nutzt den Quelltext eines Programms und dynamische Analyse versucht das Laufzeitverhalten von Programmen zu untersuchen. *Visualisierungstechniken* werden angewandt, um die Ergebnisse der Analyse darzustellen. Dafür werden verschiedene Arten von Graphen benutzt. Die gesammelten Daten haben oft einen geringen Informationsgehalt und machen die Visualisierung oft groß und unübersichtlich. Einige Untersuchungen zur optimalen Visualisierung und Navigation in diesen Graphen wurden durchgeführt.

In diesem Papier zeigen wir eine Möglichkeit für die Analyse und Visualisierung von Komponenten-Programmen. Wir nutzen dazu *dynamische* Analyse und schlagen eine aussagekräftige Visualisierung für die Analyseergebnisse vor. Wir zeigen einen Prototyp des dynamischen Analyse-Werkzeugs als Teil der HOTAGENT Komponenten-Entwicklungsumgebung.

Die HOTAGENT Entwicklungsumgebung wurde zuerst in [Mar01] beschrieben. Sie ist ein Komponentenentwurfsumgebung zur Konstruktion von Agenten für den elektronischen Markt. Die Agenten können Routinearbeit wie z.B. die Untersuchung von elektronischen Dokumenten, die Extraktion von wichtigen Informationen und die Zusammenstellung eines Formulars übernehmen. Die HOTAGENT Entwicklungsumgebung bietet verschiedene Werkzeuge zur Erstellung von Komponenten, den Test von Komponenten und die Erzeugung von Agenten mit Komponenten.

2 Grundlegende Konzepte

Um dynamische Komponentenprogramm-Visualisierung zu untersuchen, ist es notwendig, den Begriff *Komponente* zu definieren. Danach betrachten wir dynamische Analyse und Programm-Visualisierung.

2.1 Komponenten

Eine *Komponente* besteht aus mehreren Klassen, die von einer einheitlichen *Schnittstelle* gekapselt werden. Die Schnittstelle ist der einzige öffentliche Teil der Komponente und erlaubt die *Kommunikation* zwischen den Komponenten.

In unserem Modell stellt jede Komponente Ein- und Ausgänge zu Verfügung. Ein *Eingang* kann Daten empfangen und liefert ggf. ein Ergebnis. Ein *Ausgang* wird aktiviert, um Änderungen in der Komponente anzuzeigen. Die Kommunikation zwischen Komponenten geschieht nur über

diese Ein- und Ausgänge. Um einen Agenten aus Komponenten zusammenzustellen, können die Ausgänge mit den Eingängen verbunden werden.

Lüer und Rosenblum [LR01] fordern Selbstbeschreibung. Dazu bietet das HOTAGENT Komponentenmodell verschiedene Möglichkeiten. Jede Komponente hat einen Beschreibungsnamen und auch Exemplarnamen. Zusätzlich hat auch jeder Ein- und Ausgang einen eindeutigen Namen.

2.2 Dynamische Analyse

Dynamische Analyse wird benutzt, um das Laufzeitverhalten eines Programms zu beschreiben. Während der Ausführung des Programms werden Informationen über die Kommunikation der Funktionen, Objekte oder Komponenten gesammelt. Oft kann die dynamische Analyse auch zum Test von Software benutzt werden.

Dynamische Analyse wird während eines Programmlaufs durchgeführt. Dazu ist es notwendig, daß das Programm durch die Analyse nicht langsamer wird, um nicht das Laufzeitverhalten zu beeinträchtigen. Souder *et al.* [SMS01] schlagen vor, Filter zu integrieren. Während der Analyse können eine Menge Daten anfallen, von denen eventuell nur ein Bruchteil interessant ist.

Die Kommunikation der Komponenten geschieht über die Ein- und Ausgänge. Für die dynamische Analyse von Komponenten, werden die Daten zwischen Ausgang und Eingang aufgezeichnet. Werden diese Daten in ihrer Reihenfolge festgehalten, ist es nicht schwer, die Kommunikation zwischen den Komponenten festzuhalten.

2.3 Programm Visualisierung

Nach der Analyse von Programmen ist es notwendig, die gesammelten Daten zu bearbeiten und eine passende Visualisierung zu finden.

Visualisierung darf nicht mit visueller Programmierung verwechselt werden. Visuelle Programmierung wird benutzt, um ein Programm unter Zuhilfenahme von graphischen Werkzeugen zu erstellen. Visualisierung präsentiert das Programm unabhängig davon, wie es erstellt wurde. Visualisierung ist nützlich, um ein Programm zu verstehen. Entwickler können dadurch leicht einen Einblick bekommen, wie ein Programm arbeitet. Es ist wichtig, daß eine Visualisierung klar und übersichtlich ist. Sie darf nicht mit Details überladen sein.

Nach Reiss [Rei97] liegt ein Problem darin, große Mengen an Daten zu bearbeiten. Ein Visualisierungswerkzeug muß in der Lage sein, große Datenmengen zu bearbeiten und muß trotzdem einfach zu bedienen sein.

3 HotAgent Visualize

Das HOTAGENT VISUALIZE Werkzeug beschreiben wir anhand eines Medizin-Beratungsagenten. Der Agent

empfängt eine Krankheitsbeschreibung per eMail und analysiert diese um eine Krankheit festzustellen. Darauf antwortet er mit einem Medikamentvorschlag.

Die dynamische Analyse beginnt mit der Sammlung von Daten der Komponentenkommunikation. Der Vorteil des HOTAGENT Komponentenmodells ist, daß alle Komponenten von einer Klasse abgeleitet sind. Dadurch müssen nur an dieser Klasse Änderungen für die Protokollierung vorgenommen werden. Das reduziert die Gefahr von zusätzlichen Fehlern und verlangsamt das Programm nicht sehr.

Die Daten, die während der Analyse gesammelt werden, sind die Quellkomponente mit Ausgangsname, die Zielkomponente mit Ausgangsname und die transportierten Daten. Diese Informationen werden sequentiell in eine Liste geschrieben. Diese Liste kann später gefiltert und ausgewertet werden.

Es ist lohnenswert, die Visualisierung ähnlich zu der statischen Visualisierung der visuellen Konstruktion des Komponentenprogramms zu halten (siehe auch Giesl [Gie01]). Dies erleichtert auch die Arbeit der Entwickler, da sie sich nicht neu in das Programm hineindenken müssen. Das Visualisierungswerkzeug HOTAGENT VISUALIZE baut deshalb auf dem visuellen Kompositionseditor HOTAGENT ASSEMBLY auf.

Die zweidimensionale Visualisierung des Kompositionsprozesses wird benutzt und in der dritten Dimension erweitert. Sie repräsentiert die Ausführungssequenz. Komponenten werden als Quader dargestellt, die deren Lebensdauer darstellen. Auf den Quadern ist jeweils ein Abbild der Komponenten gezeigt. In Abbildung 1 repräsentiert die oberste Ebene den Kompositionsprozess, der bei der eigentlichen Visualisierung nicht mehr sichtbar ist.

Das Laufzeitverhalten des Programms wird durch Pfeile dargestellt. Diese repräsentieren das protokollierte Laufzeitverhalten der Komponenten. Sie haben die gleiche Farbe wie bei der Komposition. Die Position auf der z-Achse symbolisiert die Ausführungszeit. Wird ein Pfeil unter einem anderen gezeigt, heißt es, daß die Kommunikation nach der anderen aufgetreten ist.

In unserem Beispiel startet die Kommunikation mit dem obersten Pfeil zwischen der submit und der split Komponente. Der erste Pfeil zwischen dem controller und der text analysis Komponente zeigt den Start der Textanalysearbeit des Agenten. Die Nachricht wird als eMail-Text an die text analysis Komponente geschickt. Darauf wird die symptom database (ISDB) nach Symptome gefragt. Diese Symptome werden ebenfalls zu der text analysis Komponente geschickt usw.

Zusätzlich zu der Information der Ausführungssequenz ist es möglich, Informationen über die Kommunikation selbst zu erhalten. Durch Auswahl eines Pfeiles werden der Ausgangsname, der Eingangsname und die transportierten Daten gezeigt. Diese Informationen werden nur auf Nachfra-

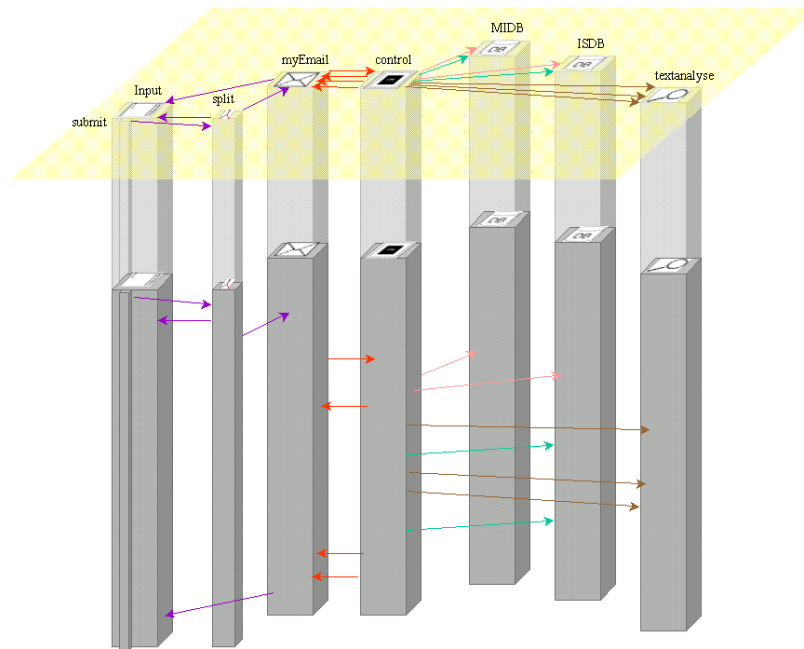


Abbildung 1. Programm Visualisierung abgeleitet von Visueller Programmierung

ge angeboten, damit die Visualisierung nicht überladen und unübersichtlich wird.

Ausserdem kann der Graph aus verschiedenen Gesichtspunkten betrachtet werden. Die dreidimensionale Ansicht kann rotiert und vergrößert/verkleinert werden. Dadurch können spezielle Teile eines Programms klar visualisiert werden.

4 Zusammenfassung und Ausblick

In diesem Papier wird das Werkzeug HOTAGENT VISUALIZE vorgestellt. Es bietet eine dreidimensionale Visualisierung des Laufzeitverhaltens von Komponentenprogrammen. Die Kommunikation wird durch dynamische Analyse ermittelt und die erlangten Daten können gefiltert werden. Die Visualisierung erlaubt sowohl die klare Sicht auf das gesamte Programm als auch auf einzelne Teile.

Eine nützliche Erweiterung wäre eine animierte Präsentation der Programmausführung. Das Werkzeug könnte die Verbindungen nacheinander mit den dazugehörigen Informationen hervorheben. Der Entwickler hätte die Möglichkeit, die Präsentation zu starten, zu stoppen oder zurückzuspulen. Der Vorteil dieser Methode ist, daß der Entwickler alle notwendigen Informationen über die Programmausführung erhält.

Literatur

[Gie01] GIESL, ANKE: *Evaluierung von Komponenten-Entwicklungsumgebungen*. Diplomarbeit, Fach-

bereich Informatik, FG Programmiersprachen und Übersetzer, Technische Universität Darmstadt, November 2001.

[Joh97] JOHNSON, RALPH E.: *Frameworks = (Components + Patterns)*. Communications of the ACM, Seite 39 – 42, October 1997.

[LR01] LÜER, CHRIS und DAVID S. ROSENBLUM: *Wren – An Environment for Component-Based Development*. In *Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Seite 207 – 217, September 2001.

[Mar01] MARTIN, LUDGER: *Visual Development Environment Based on Component Technique*. In *Proceedings IEEE Symposia on Human-Centric Computing Languages and Environments*, Seite 346 – 347, September 2001.

[Rei97] REISS, STEVEN P.: *Cacti: A Front End for Program Visualization*. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, Seite 46 – 49, 1997.

[SMS01] SOUDER, TIM, SPIROS MANCORIDIS und MAHER SALAH: *Form: A Framework for Creating Views of Programm Executions*. In *Proceedings IEEE International Konferenz on Software Maintenance*, Seite 612 – 620, November 2001.