Integrierter Deduktiver Software-Entwurf

Reiner Hähnle Wolfram Menzel Peter H. Schmitt

Zusammenfassung

Trotz unbestreitbarer lokaler Erfolge beschränkt sich die Anwendung von Deduktion im Bereich der Software-Entwicklung auf die universitäre Forschung.

Langfristiges Ziel des Projektes „Integrierter Deduktiver Software-Entwurf“ ist es, formale Methoden in betriebliche Anwendungen zu transferieren. Konkret soll ein gebräuchliches kommerzielles Meta-CASE-Werkzeug um deduktive Komponenten erweitert werden. Als begleitende Fallstudie ist der Entwurf und die Verifikation von Java Card-Programmen für Chipkarten vorgesehen.

1 Motivation

Deduktive Methoden im Software-Entwurf sind in den letzten Jahren erneut in forcierter Weise sowohl kritisch wie optimistisch aufgegriffen worden. Insbesondere hat sich das im DFG-Schwerpunktprogramm „Deduktion“ gezeigt.

Für Systeme zum formalen Beweisen lassen sich im wesentlichen zwei Paradigmen unterscheiden: einerseits das *vollautomatische Beweisen*, das trotz deutlichem Leistungszuwachs (Systeme wie etwa Otter, $_3T^AP$ [BHOS96] oder SETHEO [MIL+97]) gemessen an realen Anforderungen noch immer recht enge Grenzen besitzt, und andererseits das *interaktive* (oder taktische) Beweisen, wie es z.B. von den Systemen ISABELLE [Pau94], PVS [ORS92] oder KIV [Rei95] verfolgt wird. In einem Vorläuferprojekt zum hier vorgestellten wurde die gewinnbringende Kombination beider Paradigmen durch die Integration der Systeme KIV und $_3T^AP$ begonnen, und ein erstes Stadium hierzu abgeschlossen [ABH+98]. Die weitere Entwicklung setzt sich derzeit intensiv fort.

Auf seiten der Anwendung wurden bereits verschiedene größere Fallstudien durchgeführt, die gezeigt haben, daß Software-Verifikation inzwischen auch bei umfangreichen Projekten erfolgreich eingesetzt werden kann [CW96, GCR93, BJ95].

Trotz dieser unbestreitbaren lokalen Erfolge steckt der Transfer von formalen Methoden und Deduktion aus der universitären Forschung in betriebliche Anwendungen noch in den Anfängen. Die Hinderungsgründe sind vielfältig. Einerseits fällt die Einschätzung des wirtschaftlichen Nutzens sehr unterschiedlich aus und führt im industriellen Bereich zu einer deutlichen Zurückhaltung, vgl. [BS92]. Dies hängt mit dem Stellenwert zusammen, den die Qualitätssicherung bei der Software-Produktion im betrieblichen Umfeld einnimmt. Andererseits begründet sich die geringe Akzeptanz dadurch, daß die vorhandenen Verifikationswerkzeuge dem Benutzer weitreichende formale Kenntnisse abverlangen. Es mangelt insbesondere an der Integration formaler Methoden in bestehende Softwareentwicklungspraktiken und an Hilfsmitteln, die die Welt der Kalküle und Beweisstrategien zugänglich machen.

Jedoch ist ein Wandel zu beobachten, verbunden mit der Methodik objektorientierter Modellierung (OOM) im Software-Entwurf. Die Vielzahl der Entwurfssprachen führte Anfang

1997 zum vereinheitlichenden Vorschlag der *Unified Modeling Language* (UML)[1], einer formalen (Diagramm-)Sprache mit festgelegter Semantik. Sie wurde von der Object Management Group (OMG) als Standard verabschiedet und wird von etlichen Software-Produzenten wie Dec, HP und Microsoft mitgetragen.

Im Projekt „Integrierter Deduktiver Software-Entwurf“ wird diese Situation genutzt. Die Integration formaler Methoden in bestehende Softwareentwicklungspraktiken soll durch Erweiterung eines gebräuchlichen kommerziellen Meta-CASE-Werkzeuges, welches UML unterstützt (z.B. Paradigm Plus, ObjectMaker, ObjectTeam), erreicht werden. Zudem soll in der Weise einer „halbformalen“ Spezifikationsmethodik der Übergang von noch vorläufigen zu formalen Beschreibungen unterstützt werden.

Langfristiges Ziel bleibt das Nutzbarmachen formaler Methoden für die betriebliche Praxis. Das geplante Vorgehen läßt sich wie folgt untergliedern.

2 Erweiterung eines CASE-Werkzeuges

Konkreter Ausgangspunkt sollen Methoden und Werkzeuge sein, die im industriellen Kontext akzeptiert sind. Es soll ein gebräuchliches kommerzielles Meta-CASE-Werkzeug um die entsprechenden deduktiven Komponenten erweitert werden. Das einzusetzende Deduktionssystem, welches ebenso z.T. noch zu entwickeln ist, wird auf die anfallenden Aufgaben zugeschnitten und soll, wie oben geschildert, automatische und interaktive Methoden vereinen.

Für den Benutzer soll die in den CASE-Werkzeugen vorhandene ausgereifte visuelle Unterstützung für Entwurf und Modellierung in homogener Weise ausgedehnt werden sowohl auf die Ansteuerung der deduktiven Komponenten als auch auf die Interpretation ihrer Ergebnisse. Damit wäre die Bedienfreundlichkeit garantiert.

Alle relevanten Informationen sollen aus verschiedenen Sichten visualisiert werden können und allen wichtigen Informationsverknüpfungen soll nachgegangen werden können. Das Ziel hierbei ist, erstmals eine für den Benutzer transparente Repräsentation informeller und formaler Objekte zu erreichen. Hierfür ist eine inhaltbezogene Synchronisation zwischen informeller Spezifikation, formaler Spezifikation und Zwischenergebnissen der deduktiven Komponente notwendig.

3 Fallstudie: Verifikation von Programmen für Chipkarten

Die eben beschriebenen Aktivitäten zur Entwicklung eines Werkzeugs zum integrierten deduktiven Software-Entwurf sollen von einer praxisrelevanten Fallstudie begleitet werden.

Als Anwendungsgebiet ist der Entwurf und die Verifikation von Java Card-Programmen für Chipkarten mit integriertem Prozessor vorgesehen.[2] Dieses eignet sich aus mehreren Gründen besonders gut. Zum einen ist die Aufgabenstellung von zukunftsträchtiger Bedeutung: Die Verwendung von Chipkarten reicht vom elektronischen Handel und „electronic banking“ über Krankenversicherungskarten und Benutzeridentifikation bis zur Speicherung von Kfz-Servicedaten. Zum anderen ist sowohl die Größe der auf Karten gespeicherter Pro-

[1] Dokumentation ist frei erhältlich unter `http://www.rational.com/uml/documentation.html`.

[2] Seit Mai 1997 sind Java-fähige Chipkarten verfügbar, auf denen eine Virtuelle Maschine für die Teilsprache *Java Card* von Java realisiert ist [Gut97].

gramme wie der Sprachumfang von Java Card beschränkt, was die Erfolgsaussicht formaler Verifikation erhöht. Schließlich liegen bei Anwendungen von Chipkarten hohe Sicherheitsanforderungen vor: E4, was mindestens ein formales Sicherheitsmodell beinhaltet, ist üblich.

Literatur

[ABH+98] W. Ahrendt, B. Beckert, R. Hähnle, W. Menzel, W. Reif, G. Schellhorn, and P.H. Schmitt. Integration of automated and interactive theorem proving. In W. Bibel and P. Schmitt, editors, *Automated Deduction: A Basis for Applications*, volume II, chapter 4, pages 97–116. Kluwer, 1998.

[BHOS96] B. Beckert, R. Hähnle, P. Oel, and M. Sulzmann. The tableau-based theorem prover $_3T^AP$, version 4.0. In M. McRobbie and J. Slaney, editors, *Proc. 13th Conference on Automated Deduction, New Brunswick/NJ, USA*, volume 1104 of *LNCS*, pages 303–307. Springer-Verlag, 1996.

[BJ95] M. Broy and S. Jähnichen, editors. *KORSO: Methods, Languages, and Tools for the Construction of Correct Software – Final Report*, volume 1009 of *LNCS*. Springer-Verlag, 1995.

[BS92] J. Bowen and V. Stavridou. Safety-critical systems, formal methods, and standards. Technical Report PRG-TR-5-92, Oxford University Computing Laboratory, UK, 1992. Available via ftp from `ftp.comlab.ox.ac.uk` under `/pub/Documents/techpapers/Jonathan.Bowen/sfs-sej.ps.Z`.

[CW96] E. Clarke and J. M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, Dezember 1996.

[GCR93] S. Gerhardt, D. Craigen, and T. Ralston. *An International Survey of Industrial Applications of Formal Methods*, volume 1: Purpose, Approach, Analysis and Conclusions. U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg, MD 20899, USA, March 1993.

[Gut97] Scott B. Guthery. Java Card: Internet computing on a smart card. *IEEE Internet Computing*, 1(1):57–59, 1997.

[MIL+97] M. Moser, O. Ibens, R. Letz, J. Steinbach, Ch. Goller, J. Schumann, and K. Mayr. SETHEO and E-SETHEO—the CADE-13 systems. *Journal of Automated Reasoning*, 18(2):237–246, April 1997.

[ORS92] S. Owre, J. Rushby, and N. Shankar. PVS: A protoype verification system. In D. Kapur, editor, *Proc. 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY*, volume 607 of *LNCS*, pages 748–752. Springer-Verlag, 1992.

[Pau94] L.C. Paulson. *Isabelle: A Generic Theorem Prover*. Springer-Verlag, Berlin, 1994.

[Rei95] W. Reif. The KIV-approach to software verification. In [BJ95], 1995.