Programmiersprache Ada

Alexander Ettingshausen

Hochschule RheinMain, 65195 Wiesbaden, Deutschland
Fachbereich Design Informatik Medien
Informatik Technische Systeme
Embedded Systems WS 2019/2020
alexander.ettingshausen@student.hs-rm.de

Abstract. In dieser schriftlichen Ausarbeitung wird die Programmiersprache Ada vorgestellt. Am Anfang wird die Historie der Sprache in ihrer Entstehung als Militärauftrag und internationales Projekt, die Entwicklung mit ihren anfänglichen Schwierigkeiten und ihre erschienenen Standards betrachtet. Dann werden die Charakteristika von Ada dargestellt und darauf einige Umsetzungen genannt. Danach werden die charakteristischen Anwendungsgebiete der Programmiersprache behandelt und in zwei Beispielen nähergebracht. Als Beispiele sind die Ariane 5 und der Eurofighter Typhoon ausgewählt worden. Zum Ende der Arbeit gibt es eine Zusammenfassung, in der die wichtigsten Punkte dieser Arbeit noch einmal gesammelt werden und die Verwendung der Sprache Ada für sicherheitskritische Systeme empfohlen wird. Letztlich wird ein Ausblick auf den kommenden neuen Standard Ada 202x präsentiert.

Schlüsselwörter: Ada · Embedded Systems · Echtzeitsysteme.

1 Einleitung

Die Programmiersprache Ada ist sicherlich nicht jedem Informatiker bekannt. Dabei bietet sie viele Stärken, die in Embedded Systems, Echtzeitsystemen und sicherheitskritischen Anwendungen von großer Bedeutung sind.
In dieser Arbeit soll Neueinsteigern und Interessierten die Programmiersprache Ada nähergebracht werden. Hierzu wird ein tiefgründiger Einblick in die Historie von Ada gegeben, Charakteristika und Anwendungsgebiete der Sprache herausgearbeitet und ein paar Beispiele präsentiert, die ihr Vertrauen in Ada setzen.
Diese Ausarbeitung ist eine Literaturarbeit, das heißt, dass sich in Literatur zum Thema Ada eingelesen wurde und wesentliche Punkte herausgearbeitet wurden. Die Themen und der Aufbau der Arbeit orientieren sich an der gegebenen Aufgabenstellung aus der Lehrveranstaltung.

2 Historie

2.1 Entstehung

1973 sah sich das US Department of Defense (DoD), also das Verteidigungsministerium der Vereinigten Staaten von Amerika, steigenden Softwarekosten ausgesetzt. Dabei fielen ungefähr 56 Prozent der ca. 3 Milliarden US Dollar, die für dieses Jahr an Softwarekosten erwartet wurden, auf den Bereich Embedded Systems (vgl. [1, S. 24 f.]). Dazu trugen einige Faktoren bei. Innerhalb des DoD wurden über 450 verschiedene Programmiersprachen und nicht kompatible Dialekte verwendet (vgl. [2, Kapitel Wrap Up]). Daher gab es keine weit verbreitete Programmiersprache und auch nur eine begrenzte Anzahl an Tools für diese Sprachen. Dadurch fielen hohe Kosten für die Ausbildung des Personals und die Wartung von Systemen an, da beinahe jedes System einzigartige Anforderungen hatte. Und in den Bereichen, in denen das DoD kein eigenes Personal hatte, musste es sich auf den Support von Herstellern verlassen. Außerdem war nahezu keine Wiederverwendung der Software möglich.
1974 schlugen verschiedene Gruppen in den Military Departments vor bei der Softwareentwicklung von militärischen Systemen eine gemeinsame Programmiersprache zu verwenden. 1975 wurde daraufhin die High Order Language Working Group (HOLWG) gegründet mit dem Ziel die Kosten für die Entwicklung und Wartung von Software zu reduzieren und die Qualität der Software zu erhöhen. Die HOLWG setzte sich aus Mitgliedern der US Departments (z. B. Army, Navy, Air Force, NSA, NASA) und Vertretern aus anderen Ländern zusammen, aber auch die Öffentlichkeit hatte die Möglichkeit Input zu liefern (vgl. [2, Kapitel HOLWG]). Um die Ziele zu erreichen, sollten die technischen Anforderungen für eine General Purpose Language für Embedded Systems erfasst werden und mit existierenden Programmiersprachen verglichen werden. Die Anforderungen wurden über die Zeit weiterentwickelt und verfeinert. Im Juni 1978 wurde die finale Version Steelman veröffentlicht. Informationen zu den Anforderungen Strawman, Woodenman, Tinman, Ironman und Steelman sind in [3] zu finden.
Die Evaluation sah keine der existierenden Programmiersprachen den Anforderungen genügend entsprechen und daher wurde beschlossen, dass eine eigene Programmiersprache entwickelt werden soll. Das DoD startete darauf einen internationalen Wettbewerb, bei dem Designs für eine Programmiersprache gefragt waren. Aus den zahlreichen Einsendungen wurden vier Designs zur weiteren Ausarbeitung finanziert. Diese vier Designs bekamen Codenamen, um einen möglichst fairen Wettbewerb zu ermöglichen. Die Designs waren : Green von Cii Honeywell Bull und Jean Ichbiah, Red von Intermetrics, Blue von SofTech und Yellow von SRI International. Alle vier Designs basieren auf der Programmiersprache Pascal.
Im Mai 1979 wurde Green als der Gewinner des Wettbewerbs bekanntgegeben und in Ada umbenannt nach Augusta Ada Byron King, Countess of Lovelace, Mathematikerin und der ersten Programmiererin.
Umfangreiche Einblicke in die Entstehung von Ada sind in [1], [2] und [4] zu finden.

2.2 Entwicklung

1980 wurde Ada zum amerikanischen Militärstandard (MIL-STD 1815) erhoben und 1983 von der amerikanischen Organisation für Standardisierung anerkannt (ANSI/MIL-STD 1815A). Der erste internationale Standard von Ada wurde 1987 veröffentlicht, bekannt als Ada 83 (ISO/IEC 8652:1987). Die weiteren Standards werden in 2.3 dargestellt.
Ada hatte Startschwierigkeiten. Dies lag vor allem daran, dass zugehörige Compiler zu teuer, langsam und umfangreich waren. Man benötigte zum Teil sogar spezielle Speicherkarten, um Ada Compiler nutzen zu können (vgl. [5]). Dadurch konnten sich vor allem Privatpersonen die Nutzung von Ada nicht leisten.
Mit dem neuen Standard Ada 95 (ISO/IEC 8652:1995) kam auch ein neuer Compiler für Ada heraus, der GNAT Compiler. Dieser Ada Compiler wurde an der New York University entwickelt und von der US Air Force unterstützt (vgl. [6, S. 48]). Der GNAT Compiler ist in Ada geschrieben, nutzt GCC als Backend und ist erhältlich unter der General Public License (GPL) (vgl. [7, S. 9 ff.]). Dadurch stand der Verbreitung und Nutzung von Ada in der Öffentlichkeit nichts mehr im Weg. Ein sehr ausführliches Werk über den GNAT Compiler ist [7].

2.3 Standards

Ada 83 ist der erste Standard von Ada und wurde von Cii Honeywell Bull und Jean Ichbiah entwickelt. Er basiert auf der Programmiersprache Pascal und richtet sich nach den Steelman Anforderungen. Der internationale Standard ISO/IEC 8652:1987 wurde im Juni 1987 festgelegt und übernahm den ANSI/MIL-STD 1815A aus dem Jahr 1983. Ada 83 beinhaltet bekannte Kontrollstrukturen, aber auch spezialisierte Konzepte, wie die Erstellung von Tasks innerhalb der Programmiersprache, eine starke Typisierung, Echtzeitprogrammierung und die Behandlung von Ausnahmefällen (vgl. [8, Kapitel 1]). Für ausführliche Informationen sei auf das Ada 83 Language Reference Manual [8] verwiesen.
Ada 95, auch bekannt als ISO/IEC 8652:1995, ist der zweite Standard von Ada und erschien im Jahr 1995. Dieser zweite Standard fügte die Möglichkeit der objektorientierten Programmierung hinzu. Außerdem "wird zwischen Kernsprache und einem allgemeinen Standard einerseits und speziellen Erweiterungen (Annexes) für bestimmte Anwendungsfelder andererseits unterschieden" [9, S. 6]. Es standen nun Erweiterungen für Ada zur Verfügung, die sich speziellen Anwendungen widmeten, wie zum Beispiel Systemprogrammierung, Interfacing zu anderen Sprachen, Echtzeitsysteme, verteilte Systeme oder Security. Weitere Informationen sind in [9] und im Ada 95 Reference Manual [10] zu finden.
Ada 2005 ist eine Erweiterung des zweiten Standards und wurde 2007 als internationaler Standard ISO/IEC 8652:2007 anerkannt. Es wurden verschiedene Arten der Vererbung hinzugefügt und die Erweiterung für Echtzeitsysteme um das Ravenscar Profil erweitert. Das Ada 2005 Reference Manual [11] bietet einen detaillierten Einblick.
Ada 2012 ist der aktuelle und dritte Standard von Ada und wurde im Jahr

2012 veröffentlicht als ISO/IEC 8652:2012. Wichtige Erweiterungen dieses Standards sind das Konzept Design by Contract mit Pre- und Postconditions, die Möglichkeit Funktionen mit allen Formen von Ein- und Ausgabeparametern auszustatten, die Unterstützung von Multiprozessorarchitekturen und die Zuweisung von Tasks an einen bestimmten Prozessor und neue Formen von Containern (vgl. [12, Preface xix f.]). Mehr zum aktuellen Standard von Ada ist in [12] und [13] zu finden.
Weitere Dokumente zu den Standards von Ada sind in [14] zusammengestellt.

3 Charakterisierung

Die Programmiersprache Ada ist eine General Purpose Language, das heißt, dass sie nicht nur für einen bestimmten Anwendungsbereich genutzt wird sondern grundsätzlich in allen Bereichen verwendet werden kann. Ada ist eine imperative Programmiersprache und folgt dem prozeduralen Ansatz, also der Folge von Anweisungen und Prozeduren. Sie unterstützt aber auch die Möglichkeit auf objektorientierte Programmierung zuzugreifen. In Ada gibt es eine Kernsprache und dazu die Möglichkeit Erweiterungen einzubinden, die für spezielle Anwendungsgebiete entwickelt wurden (vgl. [9, S. 6]). Es stehen Erweiterungen für Systemprogrammierung, Echtzeitsysteme, verteilte Systeme, Informationssysteme, komplexe Mathematik und High Integrity Systems zur Verfügung. In diesen Erweiterungen existieren Profile, die das jeweilige Thema behandeln. Die Programmstruktur in Ada wird durch eine strikte Trennung von Deklaration und Implementierung geprägt und der Wiederverwendbarkeit von Code durch Modularisierung. Des Weiteren herrscht in Ada das Prinzip der starken Typisierung. Ada ist case insensitive, das heißt die Groß- und Kleinschreibung muss nicht beachtet werden, jedoch ist dies zu empfehlen. Der Aufbau eines Ada Programms und die Kontrollstrukturen sind besonders auf eine gute Lesbarkeit zugeschnitten. Zum Beispiel wird der Anfang einer Prozedur oder einer Funktion mit dem Keyword begin und das Ende mit dem Keyword end und dem Prozedur- bzw. Funkktionsnamen gekennzeichnet. Auch das Ende von if Verzweigungen oder Schleifen wird mit end if bzw. end loop markiert. Außerdem bietet Ada Laufzeitumgebungen und Multitasking mithilfe von Tasks als Teil der Programmiersprache an. Daher kann nicht nur die Einhaltung von Anforderungen beim Kompilieren sondern auch während der Ausführung gewährleistet werden (vgl. [15, S. 146 f.]). Ada wird in einigen internationalen Sicherheitsstandards, wie in den Normen IEC 61508, EN 50128 oder DO-178B, empfohlen. In der Norm IEC 61508 wird “Ada als ’Highly Recommended’ ” [15, S. 182] für sicherheitskritische Systeme betitelt. Ada bietet auch die Möglichkeit des Interfacing in anderen Sprachen. “Dies kann in Maschinensprache oder in eine andere Hochsprache erfolgen, wie Fortran oder C” (übersetzt aus [12, S. 798]).

4 Umsetzungen

Das Ravenscar Profil ist eine Erweiterung für Echtzeitsysteme und "ermöglicht eine statische Analyse und zugleich die Zertifizierung bis zu den höchsten Sicherheitslevels (SIL 4)" [15, S. 182]. Um dies zu erreichen bietet das Ravenscar Profil Determinismus, Task Planbarkeit, Synchronisation und Kommunikation und auch Speicherbeschränktheit (vgl. [15, S. 182 f.]).
Die Programmiersprache SPARK ist eine Umsetzung von Ada, die sich darauf fokussiert ein Programm vor der Ausführung auf Korrektheit zu überprüfen. Daher eignet sie sich für sicherheitskritische Anwendungen von denen die Umwelt und Menschenleben abhängen, wie High Integrity Systems (vgl. [12, S. 839]).
Der GNAT Compiler ist an der New York University entwickelt worden mit der Unterstützung der US Air Force (vgl. [6, S. 48]). Er wurde 1995 zusammen mit dem zweiten Standard von Ada unter der General Public License (GPL) veröffentlicht. Der GNAT Compiler ist in Ada geschrieben und nutzt GCC als Backend (vgl. [7, S. 9 ff.]). Ein sehr ausführliches Werk über den GNAT Compiler ist [7].

5 Anwendungsgebiete

Charakteristische Anwendungsgebiete der Programmiersprache Ada sind allgemein Embedded Systems, Echtzeitsysteme und sicherheitskritische Anwendungen. Um dies ein wenig zu konkretisieren, werden hier einige Anwendungsgebiete von Ada genannt :

- Luft- und Raumfahrt
- militärische Systeme
- Prozesssteuerung
- Systeme zur Überwachung
- Telekommunikation
- Schienenverkehr
- Medizin

Diese Anwendungsgebiete besitzen gewisse Eigenschaften. Meistens nimmt die Entwicklung dieser Systeme große Ausmaße an und ist aufwendig. Daher soll ein solches System auch um die 20 bis 30 Jahre in Betrieb sein (vgl. [9, S. 9]). Außerdem sind sie "reaktiv, da sie auf Einflüsse der Außenwelt reagieren müssen und zwar in Realzeit" [9, S. 9] und "hardwareabhängig wegen der Anbindung an die Außenwelt und der Effizienzanforderung" [9, S. 9]. Deshalb spielt die Zuverlässigkeit dieser Systeme eine große Rolle (vgl. [9, S. 9]).

6 Beispiele

6.1 Ariane 5

Die Ariane 5 ist eine Trägerrakete, die Satelliten in die Erdumlaufbahn befördert. Leider zerstörte sich die Ariane 5 bei ihrem Erstflug am 4. Juni 1996 selbst. Die Selbstzerstörung trat ca. 40 Sekunden nach dem Start ein aufgrund eines Integer Overflows. Dies lässt sich aber nicht auf die Nutzung von Ada zurückführen, sondern auf fahrlässige Softwareentwicklung. Denn die ermittelten Ursachen für den Unfall sind zum einen die Übernahme von Code der Vorgängerrakete Ariane 4 und zum anderen die fehlende Behandlung von Ausnahmefällen. Ein detaillierter Bericht über den Vorfall ist in [16] zu finden.
Der Unfall ist jedoch nicht das Ende des Projektes Ariane 5 gewesen. Sie wurde in mehreren Versionen weiterentwickelt und wird heute noch verwendet. Die aktuelle Version ist die Ariane 5 ECA.

6.2 Eurofighter Typhoon

Der Eurofighter Typhoon ist ein Kampfjet, der aus einem europäischen Projekt zwischen Deutschland, Spanien, Italien und Großbritannien entstanden ist. Der Großteil der Bordcomputer ist in Ada programmiert. Die Bordcomputer kontrollieren die Flug- und Waffensysteme des Kampfjets. Weitere Informationen zu der Verwendung von Ada im Eurofighter Typhoon gibt es hier [17].

7 Zusammenfassung

Ada ist eine besondere Programmiersprache. Dies lässt sich allein bei der Betrachtung ihrer Historie erkennen. Ada entstand aus der Intention die Entwicklung von Software zu vereinfachen und zu verbessern, um damit die Kosten für die Entwicklung zu verringern. Dazu wurde ein internationaler Wettbewerb zur Gestaltung einer Programmiersprache gestartet, der der breiten Öffentlichkeit die Möglichkeit bat Beiträge zu liefern und aus dem Ada hervorkam. Die Sprache hat seine Stärken in der strikten Trennung von Deklaration und Implementierung, der starken Typisierung, dem Task Konzept und den Laufzeitumgebungen, der Modularisierung oder auch den Erweiterungen für spezielle Anwendungsgebiete. Jedoch eignet sich Ada wegen dieser Regulierungen nicht so gut, um schnell ein kleines Programm zu schreiben. Hierzu sind andere Programmiersprachen wie zum Beispiel Python besser geeignet. Dafür sollte man die Nutzung von Ada in Betracht ziehen, wenn man große Projekte in den Bereichen Embedded Systems, Echtzeitsystemen oder in sicherheitskritischen Anwendungen anfertigen möchte. Besonders wenn Faktoren wie die Lebensdauer und die Zuverlässigkeit eines Systems eine wichtige Rolle spielen, ist Ada eine gute Wahl. Mehrere internationale Sicherheitsstandards empfehlen Ada für sicherheitskritische Anwendungen.

8 Ausblick

Die Zukunft von Ada verspricht weitere Neuerungen und Verbesserungen. In diesem Abschnitt wird ein Ausblick auf den vierten Standard Ada 202x präsentiert, der in den nächsten Jahren zu erwarten ist. Ein zentraler Aspekt dieses neuen Standards ist die Einführung der parallelen Programmierung. Hierzu werden zurzeit verschiedene Konstrukte entwickelt, wie zum Beispiel parallele Blöcke oder parallele Schleifen. Außerdem soll ein weiteres Profil dem Ada Standard hinzugefügt werden, das Jorvik Profil. Dieses Profil soll umfangreichen Support für harte Echtzeitsysteme bieten (vgl. [18], [19]).

Literatur

1. Fisher, D.: DoD's Common Programming Language Effort. Computer. Volume 11, Issue 3, 24–33 (1978)
2. Ada - DoD HOLWG, Col Wm Whitaker, 1993, http://archive.adaic.com/pol-hist/history/holwg-93/holwg-93.htm. Zuletzt besucht am 02.03.2020
3. Overview of the Ada Computer Language Competition, http://iment.com/maida/computer/redref/index.htm. Zuletzt besucht am 02.03.2020
4. Introduction to Steelman On-Line, https://dwheeler.com/steelman/index.html. Zuletzt besucht am 02.03.2020
5. Make it Simple: A Tale about Robert Dewar, https://dwheeler.com/essays/make-it-simple-dewar.html. Zuletzt besucht am 02.03.2020
6. Schonberg, E., Banner, B.: The GNAT Project: A GNU-Ada 9X Compiler. Proceedings of the Conference on TRI-Ada '94. 48–57 (1994)
7. Miranda, J., Schonberg, E.: GNAT: The GNU Ada Compiler. Applied Microelectronics Research Institute University of Las Palmas de Gran Canaria Canary Islands Spain, Computer Science Department New York University U.S.A. (2004)
8. Ada '83 Language Reference Manual, http://archive.adaic.com/standards/83lrm/html/lrm-TOC.html. Zuletzt besucht am 02.03.2020
9. Nagl, M.: Ada - Eine Sprachinitiative mit weitem Horizont. CD der Ada Deutschland (2004)
10. Ada Reference Manual ISO/IEC 8652:1995, http://docs.adacore.com/live/wave/arm95/html/arm95/arm95.html. Zuletzt besucht am 02.03.2020
11. Ada Reference Manual ISO/IEC 8652:2007, http://docs.adacore.com/live/wave/arm05/html/arm05/arm05.html. Zuletzt besucht am 02.03.2020
12. Barnes, J.: Programming in Ada 2012. Cambridge University Press. Cambridge (2014)
13. Ada Reference Manual ISO/IEC 8652:2012, http://docs.adacore.com/live/wave/arm12/html/arm12/arm12.html. Zuletzt besucht am 02.03.2020
14. Ada Standard, Rationale and other Documents, http://www.ada-europe.org/resources/online/. Zuletzt besucht am 02.03.2020
15. Keller, H.: Entwicklung von Echtzeitsystemen. Springer Vieweg, Wiesbaden (2019)

16. ARIANE 5 Failure - Full Report, http://sunnyday.mit.edu/nasa-class/Ariane5-report.html. Zuletzt besucht am 02.03.2020
17. Case Study BAE Systems Eurofighter Typhoon, https://www.adacore.com/customers/eurofighter-typhoon. Zuletzt besucht am 02.03.2020
18. Taft, T.: A 20-20 View of Ada An Evolutionary Perspective. AdaCore. Warschau (2019)
19. Ada 202x Language Reference Manual, http://www.ada-auth.org/standards/ada2x.html. Zuletzt besucht am 02.03.2020