

Vorwort zur zweiten Auflage

Die neue Auflage wurde an vielen Stellen formal und an wenigen Stellen auch inhaltlich korrigiert. Eine Anpassung an den neuen Standard, der 2011 verabschiedet wurde, fand nicht statt. Da in diesem Buch die praktische Anwendung moderner Paradigmen im Vordergrund steht, habe ich mich auf den Umfang der Sprache beschränkt, der durch die üblichen Produktivcompiler unterstützt wird. Dieser Umfang ist heute der ISO Standard von 1998 und dessen Korrekturfassung von 2003. Die Compiler haben nach der Standardisierung lange gebraucht, um der Norm zu genügen. Als wirklich allgemein konnte der Standard erst um das Jahr 2008 herum gelten. Zumindest war es dieses Jahr, in dem ich selbst zum ersten Mal in der Projektarbeit keine besonderen Abweichungen mehr irgend eines bestimmten Compilers vom Standard beachten musste. Da aber immer noch Codes existieren, die auf abweichendem Compilerverhalten basieren, habe ich die Abschnitte in diesem Buch, die sich mit dieser Problematik befassen, noch nicht entfernt. Der neue Standard wird von den heutigen Compilern noch nicht unterstützt, obwohl schon einige Teilbereiche experimentell zur Verfügung stehen. Meiner Meinung nach braucht dieser experimentelle Bereich noch eine gewisse Zeit der Reifung, um erfolgreich in Produktivcode eingesetzt werden zu können. Die letzten Jahre haben gezeigt, dass im Rahmen der Möglichkeiten des ersten ISO Standards neue Bibliotheken und Frameworks für die eingebettete Softwareentwicklung entstanden sind, die die im Buch vorgestellten Techniken intensiv nutzen. Es ist in den letzten fünf Jahren zu einer massiven Ausweitung des Frameworkeinsatzes in der eingebetteten Softwareentwicklung gekommen. C++ hat sich dabei bis heute als deterministische Programmiersprache in diesem Umfeld behauptet und wird umfangreich eingesetzt, wenn es um die Ansteuerung von Hardware geht oder um eine hohe Performance.

Vorwort zur ersten Auflage

Um im Jahr 2006 noch ein Buch über C++ zu schreiben, bedurfte es meinerseits einiger Vorüberlegungen, bis ich mich zu diesem Schritt entscheiden konnte. Da ist zum einen die aktuelle Veränderung in der Nutzung von Programmiersprachen in der Anwendungsentwicklung, die ein deutliches Zeichen für die Zukunft von Java und anderen moderneren Programmiersprachen ist und ein ebensolches Zeichen für das allmähliche Verschwinden von C++ aus dieser Domäne. Zum anderen ist da ein scheinbares Defizit in der Sprache C++ selbst, die sich ein Stück weit gegen die unmittelbare Anwendung neuer Technologietrends zu sträuben scheint. Die enorme Komplexität der Sprache ist dafür sicher ein Grund unter anderen. Was sind also meine Gründe, doch ein Buch über diese alte und noch in vielen Bereichen unersetzliche Sprache zu schreiben?

In den letzten Jahren fand eine rasante Entwicklung bei den Mobile Devices, den Embedded Systems und den vermittelnden Systemen in Netzen statt. Diese Entwicklung dauert auch heute noch an. Es entsteht viel mehr neue Hardware als in den neunziger Jahren und es entsteht die dazugehörige Betriebssystemlandschaft. Gerade für diese neuen Systeme braucht man heute C++ als systemnahe und deterministische Programmiersprache. Da heute aufgrund der besseren Portierbarkeit mancher gängiger Compiler C++ auch auf fast allen neuen Plattformen zur Verfügung steht, erfährt die Sprache genau dort eine späte Verbreitung. Häufig werden Systeme und Teile davon in Crosscompileumgebungen entwickelt, die Windows oder Linux als Entwicklungsbasis definieren und ein Embedded System als Zielsystem haben. Diese aktuelle Situation ist es, die C++ meiner Meinung nach noch zu einem starken Entwicklungspotential verhilft, denn sie stellt an den Entwickler dieser neuen Domäne viel höhere Anforderungen als an den reinen Anwendungsentwickler, der es in den neunziger Jahren gewohnt war, C++ zusammen mit der Unterstützung mächtiger Frameworks einzusetzen.

Während in den Neunzigern die verwendeten Methoden lange Zeit auf der klassischen Objektorientierung stagnierten, brachte der ANSI/ISO-Standard Ansätze mit, die in eine ganz andere Richtung weisen, als es die traditionellen C++-Techniken tun. Techniken, deren methodische Anwendung in der Praxis noch längst nicht vollständig erprobt und ausgeschöpft ist und die sich mit



<http://www.springer.com/978-3-642-21428-8>

Moderne C++ Programmierung
Klassen, Templates, Design Patterns
Schneeweiß, R.
2012, XIII, 393 S. 21 Abb., Hardcover
ISBN: 978-3-642-21428-8