Physikalische Chemie, Universität Rostock

Vorlesung: Wissenschaftliche Softwareentwicklung

Wintersemester 2023/24

Dr. habil. Till Biskup

— Glossar zu Lektion 12: „Funktionen“ —

Hinweis: Die nachfolgend genannten Begriffe und Definitionen erheben keinen Anspruch auf formale Korrektheit, sondern dienen lediglich dem besseren Verständnis der in der Vorlesung behandelten Themen und sind im jeweiligen Kontext zu sehen. Mehrfache, voneinander abweichende Definitionen in unterschiedlichen Kontexten sind daher möglich. Englische Begriffe werden zwar nach Möglichkeit übersetzt, erscheinen aber ggf. unter ihrem englischen Namen in der Liste. Verweise untereinander sind durch ↑ gekennzeichnet.

Abstraktion Nach Edsger Dijkstra [1] das einzige mentale Werkzeug, das es erlaubt, eine große Vielzahl von Fällen abzudecken. Zweck der Abstraktion ist es nicht, vage zu sein, sondern im Gegenteil ein neues Bedeutungsniveau zu schaffen, das präzise Beschreibungen erlaubt.

Abstraktionsebene Summe aller ↑Abstraktionen eines bestimmten Abstraktionsgrades. ↑Funktionen (bzw. ↑Methoden) sollten immer nur Anweisungen enthalten, die zur gleichen Abstraktionsebene gehören.

Attribut im Kontext der ↑objektorientierten Programmierung eine Variable, die innerhalb einer ↑Klasse definiert wird. ↑Methoden operieren auf den Attributen einer ↑Klasse bzw. dem daraus erzeugten ↑Objekt.

Boolescher Wert „Wahrheitswert“, einer der zwei Werte der Booleschen Algebra, repräsentiert durch „0“ („*False*“) und „1“ („*True*“). Bedingungen einer ↑Selektion müssen immer zu einem Booleschen Wert evaluierbar sein.

Compiler Im Deutschen meist als „Übersetzer“ bezeichnetes Programm, das den Quellcode eines Programms in direkt auf der Hardware ausführbaren Maschinencode übersetzt. Kompilierte Sprachen sind im Gegensatz zu interpretierten Sprachen (↑Interpreter) meist deutlich schneller, aber in binärer Form (Maschinencode) an eine spezifische Hardwareplattform gebunden. Moderne Compiler beinhalten häufig einen ↑Linker.

DRY „*Don't repeat yourself*“, (nicht nur) in der angelsächsischen Programmierwelt verbreitetes Akronym und wichtige Regel für die Programmierung. Doppelungen im Code sind meist ein guter Hinweis darauf, dass eine ↑Abstraktion fehlt bzw. nicht erkannt wurde. Darüber hinaus sind Doppelungen im Code ein zentraler Grund für schlechte Qualität und Wartbarkeit.

dyadisch hier: zwei Argumente (↑Parameter) verlangend. Es gibt sowohl dyadische (binäre) Operatoren als auch dyadische Funktionen. Beispiele für dyadische (binäre) Operatoren sind die arithmetischen Operatoren („+“, „−“, . . .). Dyadische Funktionen sind in der Programmierung weit verbreitet. Deren Namen sollten so gewählt werden, dass die Reihenfolge der Parameter offensichtlich ist. Vgl. ↑niladisch, ↑monadisch und ↑triadisch.

Exception Ausnahmesituation, dienen in der Softwaretechnik dazu, Informationen über bestimmte Programmzustände – meistens Fehlerzustände – an andere Programmebenen zur Weiterbehandlung weiterzureichen.

Framework „Programmiergerüst“, stellt den Rahmen zur Verfügung, innerhalb dessen der Pro-

grammierer eine Anwendung erstellt, wobei u. a. durch die in dem Framework verwendeten Entwurfsmuster auch die Struktur der individuellen Anwendung beeinflusst wird.

Funktion im Kontext der ↑strukturierten Programmierung eine Liste von Anweisungen, die eine bestimmte Aufgabe erfüllt und der Programmiersprache unter einem festen Namen bekannt ist.

intellektuelle Beherrschbarkeit *intellectual manageability*, nach Edsger Dijkstra [1] das Hauptziel der Softwaretechnik (*software engineering*) – und letztlich des Projektmanagements. Unterschiedliche Lösungsansätze für ein Problem sind unterschiedlich gut intellektuell beherrschbar. Entsprechend ist die intellektuelle Beherrschbarkeit das zentrale Kriterium für die Entscheidung, welche Lösung für ein Problem bevorzugt wird.

Klasse *class*, im Kontext der ↑objektorientierten Programmierung die Blaupause für die Erzeugung eines ↑Objektes; Definition der Daten (↑Attribute) und des zugehörigen Verhaltens (↑Methoden).

Methode im Kontext der ↑objektorientierten Programmierung eine ↑Funktion, die innerhalb einer ↑Klasse definiert wird und auf den ↑Attributen einer ↑Klasse bzw. dem daraus erzeugten ↑Objekt operiert.

Modularisierung Aufteilung der Gesamtaufgabe in kleinere Abschnitte. Die Aufteilung wird so lange fortgesetzt, bis die Lösung für den aktuellen Abschnitt unmittelbar in Form von Quellcode offensichtlich ist. Setzt die Definition von ↑Schnittstellen voraus.

monadisch hier: nur ein Argument (↑Parameter) verlangend. Es gibt sowohl monadische (unäre) Operatoren als auch monadische ↑Funktionen. Ein Beispiel für einen monadischen (unären) Operator wäre der Inkrement-Operator („++“). Vgl. ↑niladisch, ↑dyadisch und ↑triadisch

Nebenwirkung *side effect*, Wirkung; in der theoretischen Informatik die Veränderung des Programmzustands einer abstrakten Maschine. In der Praxis der Programmierung meist die Auswirkung einer Zuweisung eines Wertes zu einer Variablen, die außerhalb des konkret betrachteten Kontextes liegt. Da in der ↑funktionalen Programmierung auf Zuweisungen gänzlich verzichtet wird, sind funktionale Programmiersprachen frei von Nebenwirkungen, was sie für die parallele Verarbeitung prädestiniert.

niladisch hier: kein Argument (↑Parameter) verlangend. Niladische ↑Funktionen sind selten, niladische ↑Methoden in der ↑objektorientierten Programmierung durch den vom Objekt gegebenen Kontext hingegen relativ einfach zu bewerkstelligen. Vgl. ↑monadisch, ↑dyadisch und ↑triadisch

Objekt *object*, im Kontext der ↑objektorientierten Programmierung der grundlegende Baustein eines Programms, bestehend aus den Daten (↑Attribute) und dem zugehörigen Verhalten (↑Methoden).

objektorientierte Programmierung (OOP) ein ↑Programmierparadigma, bei dem Daten (Variablen zugewiesene Werte, als ↑Attribute bezeichnet) und Funktionen (↑Methoden), die auf diesen Daten (Attributen) operieren, eine Einheit bilden. Die in den ↑Attributen gespeicherten Daten lassen sich i.d.R. nur vermittelt durch (öffentlich zugängliche) ↑Methoden der ↑Klasse bzw. des daraus erzeugten ↑Objektes ansprechen. Es gibt eine klare Trennung zwischen öffentlicher ↑Schnittstelle und internen Verarbeitungsroutinen. Wichtige Vertreter objektorientierter Programmiersprachen sind Smalltalk, C++ und Java, aber auch Python.

Paradigma nach Thomas S. Kuhn [2] ein Satz allgemein anerkannter wissenschaftlicher Leistungen, der für eine gewisse Zeit einer Gemeinschaft von Fachleuten maßgebende Probleme und Lösungen liefert

Parameter hier: Argument einer ↑Funktion oder ↑Methode

Programmierparadigma ein ↑Paradigma der Art zu programmieren. Wichtige Beispiele sind

↑strukturierte Programmierung, ↑objektorientierte Programmierung und funktionale Programmierung.

Schlüssel-Wert-Paar Kombination einer benannten Variable und ihres zugewiesenen Wertes. Wird häufig in Datenstrukturen abgelegt, die dann über den Schlüssel einen Zugriff auf den damit assoziierten Wert erlauben. Ermöglicht bis zu einem gewissen Grad ein semantisches Verständnis (↑Semantik) im Quellcode.

Schnittstelle *interface*, Begriff mit mehreren leicht unterschiedlichen Bedeutungen; (1.) ↑Signatur einer ↑Methode. (2.) Im weiteren Sinne die Gesamtheit der öffentlichen ↑Attribute und ↑Methoden einer ↑Klasse bzw. eines ↑Objekts. Der Nutzer kennt nur die Schnittstelle, die Implementierung ist irrelevant und kann sich problemlos jederzeit ändern, solange die Funktionalität erhalten bleibt. Das dient der Trennung von Verantwortlichkeiten und ermöglicht ↑Modularisierung und ist in der Folge ein wesentlicher Aspekt der Softwarearchitektur.

Seiteneffekt siehe ↑Nebenwirkung

Semantik 1. Bedeutung bzw. Inhalt eines Wortes, Satzes oder Textes; 2. Teilgebiet der Linguistik, dessen Untersuchungsobjekt die Bedeutung sprachlicher Zeichen und Zeichenfolgen ist.

semantische Information Bedeutungsebene einer Information (vgl. ↑Semantik).

Signatur hier: Name und Parameter einer ↑Funktion bzw. ↑Methode, also alles, was ein Nutzer braucht, um diese Funktion oder Methode verwenden zu können.

strukturierte Programmierung ein ↑Programmierparadigma, das die Zahl möglicher Kontrollstrukturen auf nur zwei (↑Iteration, ↑Selektion) beschränkt, insbesondere den `goto`-Befehl eliminiert (E. Dijkstra, [3]). Idealerweise hat ein Codeblock nur jeweils genau einen Ein- und Ausgang. Nach D. Knuth [4, S. x] der systematische Einsatz von Abstraktion, der es ermöglicht, große Programme aus kleine(re)n Komponenten zusammenzusetzen. Wichtige frühe Vertreter strukturierter Programmiersprachen sind C und Pascal. Die meisten heutigen Programmiersprachen (mit Ausnahme der funktionalen Programmiersprachen) unterstützen die strukturierte Programmierung.

Selektion eine von zwei Kontrollstrukturen der ↑strukturierten Programmierung, neben der ↑Iteration. In den meisten Sprachen über Bedingungen (`if...else...end`) realisiert, die sich ggf. beliebig verschachteln lassen.

triadisch hier: drei Argumente (↑Parameter) verlangend. Es gibt sowohl triadische (ternäre) Operatoren als auch triadische ↑Funktionen. Ein Beispiel für einen triadischen (ternären) Operator wäre die Kurzschreibweise einer ↑Selektion wie in der Programmiersprache `C` verbreitet (`condition ? value_if_true : value_if_false`). Triadische Funktionen sollten nach Möglichkeit vermieden werden, da es fast nicht möglich ist, die Reihenfolge der Parameter im Funktionsnamen zu codieren, und sie damit schwer nutzbar sind. Vgl. ↑niladisch, ↑monadisch und ↑dyadisch

YAGNI „*You ain't gonna need it*“, (nicht nur) in der angelsächsischen Programmierwelt verbreitetes Akronym und wichtige Regel für die Programmierung. Zielt darauf ab, jeweils nur das zu implementieren, was im Augenblick wichtig ist oder von dem zweifellos klar ist, dass es in Kürze gebraucht wird. Versuch, durch einen pragmatischen Ansatz ein „zu viel“ an Abstraktion zu vermeiden.

Zuständigkeitshierarchie allg. die Aufteilung von Zuständigkeiten auf unterschiedliche Ebenen. Eine wesentliche Aufgabe von ↑Funktionen (bzw. ↑Methoden) in einem Programm ist genau diese Aufteilung und die Erzeugung eines lokalen Kontextes sowie von passenden ↑Abstraktionsebenen, die es ermöglichen, sich immer nur auf eine Ebene der Abstraktion und damit eine Art von Problemen zu fokussieren.

Literatur

[1] Edsger W. Dijkstra. The humble programmer. *Communications of the ACM* 15 (1972), S. 859–865.

[2] Thomas S. Kuhn. *Die Struktur wissenschaftlicher Revolutionen*. Frankfurt am Main: Suhrkamp, 1976.

[3] Edsger W. Dijkstra. Go to statement considered harmful. *Communications of the ACM* 11 (1968), S. 147–148.

[4] Donald E. Knuth. *Literate Programming*. Stanford: Center for the Study of Language and Information, 1992.