Zur Berechnung der Komplexität von einfachen objektorientierten Programmen

Marc Roßner[1]

Abstract: Das Komplexitätsmaß nach Peter Rechenberg lässt sich auf einfache objektorientierte Programme übertragen. Solche Programme sind Gegenstand des Informatik-Unterrichts. Dies stellt eine Grundlage dafür dar, unterschiedliche Programme zu einer Aufgabe zu vergleichen.

Keywords: Komplexitätsmaß, Objektorientierung, Abituraufgaben

1 Übertragen der Software-Metrik von Rechenberg auf einfache objektorientierte Programme und C++

Unter einfachen objektorientierten Programmen sollen in dieser Arbeit solche Programme verstanden werden, die sich ausschließlich auf die Grundkonzepte Objekt und Klasse beziehen. Diese Konzepte bieten die Möglichkeit des strukturierten Aufbaus von Softwareprojekten. Man kann Teillösungen in Klassen auslagern und diese in einem Hauptprogramm zur Gesamtlösung vereinen. Bei diesem Vorgehen kann die Implementierung der Methoden als quasi-prozedural angesehen werden; die Methoden können in der Software-Metrik wie gewöhnliche Prozeduren behandelt werden. Für das Berechnen der Anweisungs- und der Ausdruckskomplexität können die Regelungen aus [Re86] unmittelbar übernommen werden. Das Berechnen der Datenkomplexität kann bei den Methoden so erfolgen, wie es bei Rechenberg für die modulare Programmierung beschrieben ist. Bei der Datenkomplexität werden Namen in Modulen als lokal und erst bei deren Verwendung im Hauptprogramm als global bewertet. Sollten Module ineinander geschachtelt sein, greift die Bewertung mit der Blocktiefendifferenz. Für Attribute soll gelten, dass sie im Sinne gekapselter Zustandsdaten, auf die alle oder fast alle Methoden der Klasse zugreifen, als global bewertet werden – unabhängig davon, ob innerhalb der Implementierung der Klasse oder im Hauptprogramm.

2 Analysieren der Aufgabe „Plumpsack“

Das Spiel „Plumpsack“ [Ab08] ist mithilfe einer Ringliste zu realisieren. Die Ringliste wird per Array implementiert. In Tabelle 1 sind die Bewertungen der vollständigen prozeduralen und objektorientierten Programme zum Spiel „Plumpsack“ angegeben.

[1] Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, Ernst-Abbe-Platz 2, 07743 Jena, marc.rossner@uni-jena.de

	prozedural	objektorientiert
Anweisungskomplexität	121.75	112.75
Ausdruckskomplexität	118	110
Datenkomplexität	122	136
Gesamtkomplexität	361.75	358.75

Tab. 1: Bewertung von Programmen zur Aufgabe „Plumpsack“.

Die Gesamtkomplexitäten der erarbeiteten Programme sind praktisch gleich. Daraus lässt sich ableiten, dass man es in der Abiturprüfung dem Prüfungsteilnehmer überlassen kann, ob er eine Aufgabe prozedural oder objektorientiert löst (bei Beschränkung auf die Grundkonzepte Klasse und Objekt).

3 Analysieren der Aufgabe „Liste“

Eine einfach verkettete Liste ist mit Hilfe eines Arrays zu realisieren [Ab07]. In Tabelle 2 sind die Bewertungen von drei unterschiedlichen Lösungsmöglichkeiten der Aufgabe angegeben.

	Zwei 1D-Arrays	Ein 2D-string-Array	Ein 1D-struct-Array
Anweisungskomplexität	339.5	377	338.5
Ausdruckskomplexität	200	258	221.5
Datenkomplexität	308	312	361
Gesamtkomplexität	847.5	947	921

Tab. 2: Bewertung von drei Musterlösungen zur Aufgabe „Liste“.

Wenn man dem Prüfungsteilnehmer die konkrete Realisierung der Datenstruktur überlässt, muss man damit rechnen, dass sich die Gesamtkomplexitäten der entwickelten Programme in der Größenordnung 10% voneinander unterscheiden. Diese Größenordnung erscheint auch in einer Abiturprüfung vertretbar. Nachdenklich stimmt der Unterschied in den Gesamtkomplexitäten von Programmen zu den beiden Abituraufgaben, obwohl mit beiden Aufgaben die gleiche Anzahl von Bewertungseinheiten erreichbar war (jeweils 30 BE von 60 BE für die gesamte Abiturpüfung).

Literaturverzeichnis

[Ab07] Freistaat Thüringen: Abiturprüfung 2007 Leistungsfach Informatik (Haupttermin).

[Ab08] Freistaat Thüringen: Abiturprüfung 2008 Leistungsfach Informatik (Haupttermin).

[Re86] Rechenberg, Peter: Ein neues Maß für die softwaretechnische Komplexität von Programmen. In: Informatik Forschung und Entwicklung (1986) 1: 26-37.