

Objektorientierung in der Ausbildung

Montag 30.3.1998, 13.30 - 18.00 Uhr, Raum 17.21

Diskussionsrunde 2 Auswahl einer geeigneten Programmiersprache

Positionspapier zu

Oberon-2 als erste Lehrsprache für Ingenieure

Karlheinz Hug, Helmut Ketz (FH Reutlingen)

Zur Didaktik der Informatik

Wie gut sich eine Programmiersprache eignet - das kann man nur auf eine konkrete Anwendung bezogen diskutieren. Unsere Anwendung ist die Lehre und die Kriterien ergeben sich aus einer Didaktik der Informatik. Wir wollen die Informatikausbildung an der Fachhochschule **nicht** auf einen Programmierkurs beschränken, Informatik **nicht** nur von einer Sprache ausgehend lehren. Eine Programmiersprache ist nicht Zweck, sondern Mittel informatischer Ausbildung. Das Mittel muß Raum lassen für den eigentlichen Zweck, die Beschäftigung mit informatischen Strukturen. Abstraktions-, Strukturierungs- und Modellierungsfähigkeiten sind per se programmiersprachen-unabhängig. Das Umsetzen von Algorithmen, Datenstrukturen und Objekten in eine Programmiersprache ist zwar wichtig, aber nachrangig.

Ausgehend von diesen Grundsätzen setzen wir Oberon-2 als erste Lehrsprache in der Informatik-Grundausbildung in den Studiengängen Automatisierungstechnik und Elektronik ein, weil es als klare, schlanke Hochsprache folgende Vorteile bietet.

Zur Programmiersprache

1. **Einfachheit.** Oberon-2 ist leicht erlernbar, da es aus relativ einfachen, einheitlichen Konstrukten besteht. Der Umfang der Sprache ist überschaubar, sie ist daher vollständig innerhalb der Grundausbildung (8-10 SWS Vorlesung + 4-8 SWS Praktikum) vermittelbar. Dies ist zwar kein Ziel an sich, denn man könnte auch mit einer geeigneten, in sich abgeschlossenen Teilmenge einer komplexen Sprache arbeiten. Vorteilhaft ist aber sicher, daß die Sprache durch ein handliches Referenzmanual exakt, eindeutig und klar definiert ist.

Ein Beispiel: Die Syntax von Oberon-2 umfaßt 33 Nichtterminale, die Sprachreferenz mit Beispielen 24 Seiten. Zum Vergleich: Die C++-Syntax kennt 125 Nichtterminale, C++-Referenzmanuale gibt es mehrere, das ursprüngliche hat 156, das annotierte 470 und der Standard über 700 Seiten.

2. **Kompaktheit.** Oberon-2 unterstützt einerseits alle wesentlichen Konzepte, die man von einer modernen, imperativen Programmiersprache fordern kann, andererseits enthält es kaum überflüssige, gefährliche oder veraltete Sprachkonstrukte. Dies erleichtert es Lernenden, Wesentliches aufzunehmen. Wir können uns in der Grundausbildung auf Basiskonzepte der Programmierung konzentrieren, ohne kostbare Zeit mit dem Behandeln von Sprachdetails zu vergeuden. Oberon-2 läßt durch seine Einfachheit und Kompaktheit in der Vorlesung genug Raum, sich programmiersprachenunabhängig mit Grundlagen der Informatik zu beschäftigen.

Oberon-2 schützt gerade auch den Anfänger durch sinnvolle Restriktionen. Wirth weist zu Recht darauf hin, „daß eine Sprache sich nicht nur durch das auszeichnet, was sie auszudrücken erlaubt, sondern noch mehr durch das, was sie verbietet.“ Indem Oberon-2 auf Ballast verzichtet, bieten mit Oberon-2 erworbene Programmierkenntnisse eine exzellente Basis dafür, danach andere Sprachen zu lernen.

3. **Konzeptuelle Basis.** Oberon-2 unterstützt strukturiertes, modulares und objektorientiertes Programmieren mit zugehörigen Konzepten wie getrennte Übersetzbarkeit, Kapselung, Datenabstraktion in Form abstrakter Datenstrukturen und -typen, Vererbung, Polymorphie, dynamisches Binden. Gleichzeitig ist unstrukturiertes und nichtmodulares Programmieren in Oberon-2 kaum möglich.

Oberon-2 ist zwar nicht stark verbreitet, aber durchaus **praxisrelevant**, weil seine Konzepte im Haupttrend der industriellen Softwarepraxis liegen. Für den Anwendungsbereich, für den wir ausbilden wollen - nämlich für System- und Anwendungssoftware für technische Systeme - weist Oberon-2 die erforderliche **Einsatzbreite** auf.

4. **Sicherheit.** Oberon-2 enthält ein strenges Typkonzept mit statischer Typprüfung wo sinnvoll und praktikabel; wo erforderlich - bei Reihungen und polymorphen Größen - schließt es potentielle Sicherheitslücken durch Laufzeitprüfungen. Im Unterschied zu Sprachen, die auf dynamischer Typbindung oder impliziter und expliziter Typanpassung basieren, setzt Oberon-2 auf Typprüfung. (Bei numerischen Typen kennt es den Typeinschluß, der den Teilmengenbeziehungen in der Mathematik entspricht.) Je früher ein Fehler erkannt wird, desto besser: Warum also sollte erst das Laufzeitsystem prüfen, was schon der Übersetzer prüfen kann? Daß andere Fehlerquellen Laufzeitprüfungen erfordern, ist kein Grund, auf statische Typprüfung zu verzichten.

Oberon-2 bietet mit Zusicherungen auch ein wichtiges Mittel zum systematischen Entwickeln zuverlässiger, korrekter Software und eine nützliche Testhilfe. Ein hohes Maß an Sicherheit ist sowohl für Programmieranfänger als auch für kritische Anwendungen wichtig. Als saubere, sichere Sprache kann Oberon-2 das Bewußtsein von Studierenden dafür schärfen, daß es sichere Programmkonstrukte gibt, und daß die Zeit der Sprachen, die fehleranfällige Konstrukte enthalten, eigentlich längst abgelaufen ist.

5. **Hybride Sprache.** Oberon-2 kennt sowohl Module als auch Klassen. Wir halten es für dringend geboten, unsere Studenten mit einem klaren Modulbegriff auszustatten, bevor sie in die C/C++-Arena eintreten. Das heißt: Module als anwendungsorientiertes Abstraktions- und Strukturierungsmittel, das den Übergang zur objektorientierten Denkweise nicht behindert, so daß man nahtlos von Modulen zu Klassen übergehen kann. Mit Oberon-2 ist rein modulares Programmieren direkt möglich. Dabei ist ein Programm eine Menge von Modulen, zwischen denen eine Benutzt-Beziehung (Kunde-Lieferant, Import-Export) besteht. Studierende gewinnen daran Einsicht in Strukturen, die auch in großen Softwaresystemen vorkommen.

Module kann man als Objekte betrachten. Zur objektorientierten Programmierung kommt man durch einen Abstraktionsschritt, das Zusammenfassen gleichartiger Objekte zu einer Klasse. Der neue Begriff der Klasse baut auf Bekanntem auf, dadurch ist er leichter lernbar. Ein weiterer Vorteil einer hybriden Sprache ist, daß mit ihr verschiedene Programmierstile lehrbar sind.

6. **Weiterführende Konzepte.** Mit seiner Komponentenorientierung, insbesondere dem dynamischen Laden von Komponenten, hat Oberon den Programmiersprachen ein neues Konzept gebracht, das für die zukünftige Entwicklung der Softwaretechnik richtungweisend und erfolgversprechend ist. Mit automatischer Speicherplatzbereinigung, schreibgeschütztem Export von Variablen bzw. Attributen usw. bietet es zudem sehr nützliche Merkmale, die z.B. in C++ fehlen.
7. **Neuigkeitsgrad.** Viele Studienanfänger bringen einige Pascal-Kenntnisse von der Schule mit. Nun soll die Lehrveranstaltung nicht von der Schule Bekanntes wiederholen, die Sprache soll auch nicht einigen Studierenden Vorteile auf Kosten anderer verschaffen. Oberon-2 bietet einen Kompromiß: Es ermöglicht allen einen leichten Einstieg, und wer Pascal schon kennt, lernt auch etwas hinzu.
8. **Lehrmaterial.** Es gibt zwar nicht viele, dafür aber sehr gute Lehrbücher. Wir sehen es eher als Vorteil an, daß sich die Kioskliteratur bisher nicht um Oberon kümmert.

Zu den Entwicklungsumgebungen

Wir betonen Konzepte und Methoden, nicht Werkzeuge und andere Hilfsmittel. Daher sind folgende Argumente für die Entwicklungsumgebungen zweitrangig, aber nicht irrelevant, da sie unseren didaktischen Ansatz unterstützen.

1. **Benutzbarkeit.** Oberon/F ist für Anfänger schnell erlernbar und leicht handhabbar. Da es für professionellen Einsatz konzipiert ist, ist es sehr mächtig, aber für Lehrzwecke nicht überdimensioniert. Die Studierenden erreichen seine (und ihre) Grenzen nicht so schnell. Jeder Benutzer kann sich seine eigenen Menüs zusammenstellen und die Oberfläche seinem Arbeitsstil individuell anpassen.
2. **Verfügbarkeit.** Oberon-Entwicklungsumgebungen sind Studierenden frei zugänglich. Oberon microsystems Inc. stellt kostenlose Ausbildungsversionen von Oberon/F und BlackBox Component Builder zur Verfügung. Dies ist um so mehr positiv zu erwähnen, als weitaus größere Softwarehersteller auch für Studentenversionen saftige Preise verlangen.

3. **Schlankheit.** Oberon/F ist „lean“, nicht „fat“ Software - es ist leicht zu installieren und zeichnet sich durch sparsamen Ressourcenverbrauch aus. Studierende benötigen nicht den neuesten, größten, schnellsten PC, um damit sinnvoll arbeiten zu können.
4. **Portierbarkeit.** Oberon/F ist eine moderne Entwicklungsumgebung mit einer graphischen Benutzungsoberfläche, die auf gängigen Plattformen verfügbar, aber von diesen unabhängig ist und daher die Entwicklung portabler Anwendungen gestattet.
5. **Effizienz.** Die Entwicklungsumgebungen ermöglichen schnelle Editieren-Übersetzen-Ausführen-Zyklen. Der von Oberon/F erzeugte Code ist hinsichtlich Effizienz mit optimiertem C/C++-Code vergleichbar, teilweise sogar trotz Laufzeitprüfungen schneller.
6. **Neue Konzepte.** Sprache und Entwicklungsumgebungen realisieren neue, gut durchdachte Konzepte. Indem Studierende diese Technologie neben anderen kennenlernen, können sie Vergleiche anstellen und dabei technische Kritikfähigkeit im Sinne von Transferkompetenz entwickeln.

Kritikpunkte

Nichts ist perfekt - in jeder Sprache wird jeder etwas vermissen oder etwas finden, was ihm in anderer Form besser gefällt.

1. **Fehlende höhere Konzepte.** An sich sinnvolle Sprachkonzepte wie Generizität (die Anpassung von Typen mit statischer Typbindung vereint), Mehrfachvererbung und Ausnahmebehandlung fehlen Oberon-2 wegen seiner Schlankheit.
2. **Zugriffskontrolle.** Schutzeinheit ist das Modul, nicht die Klasse. Klassen sind daher Modulen untergeordnet, sie können Zugriffsrechte nicht selbst kontrollieren. Dies kann beim Programmieren zu Einschränkungen führen, die dem Offengehalten-Prinzip und dem Hellseher-Prinzip widersprechen.
3. **Hybride Sprache.** Dieses Merkmal zählen wir zu den Vorteilen von Oberon-2; es hat aber auch gewisse Nachteile. Innerhalb eines Programms kann ein Mischmasch verschiedener Programmierstile entstehen. Dieses Problem stellt sich jedoch in der Praxis auch beim Programmieren mit C++, so daß es auch wieder vorteilhaft sein kann, Studierende in disziplinierter Strukturierung zu schulen.

Klassen sind in Oberon-2 erweiterbare Verbunde mit typgebundenen Prozeduren. Möglicherweise kommt das „Revolutionäre“ am Klassenbegriff in dieser Verpackung nicht richtig zur Geltung.

Da es oft mehrere Möglichkeiten gibt, ein Problem zu lösen, denken die Lernenden vielleicht mehr darüber nach, welche der möglichen Lösungen sie wählen sollen, statt über das zu lösende Problem.

4. **Bibliotheksmodule.** Zwischen primitiven Übungs-E/A-Modulen (In, Out, XYplane) und eigentlichen, anspruchsvollen E/A-Modulen besteht eine Kluft, die didaktisch zu überbrücken ist.

5. **Sprachdetails.** Die aus Pascal und Modula-2 gewohnten Unterbereichs- und Aufzählungstypen sind Oberon-2 durch die Schlankheitskur abhanden gekommen. Reihungen haben immer in jeder Dimension den Anfangsindex 0. Oberon-2 bietet vier verschiedene Schleifenkonstrukte, im Widerspruch zur Forderung nach Minimalität. (Hier ist z.B. Eiffel mit einem einzigen Schleifenkonstrukt schlanker.) Parameterlose Funktionen brauchen bei Vereinbarung und Aufruf ein leeres Klammernpaar (wie in C/C++), was der Forderung nach Unabhängigkeit des Zugriffs von der implementierenden Struktur widerspricht.
6. **Standard.** Einerseits wäre ein Sprachstandard wünschenswert, andererseits gibt es von Oberon-2 ausgehende interessante Weiterentwicklungen, z.B. Component Pascal und Oberon System 3.

Schwächen dieser Art betrachten wir insgesamt als für eine erste Lehrsprache nicht nachteilig, sondern unvermeidbar und daher tolerierbar. Sie wiegen als Grund nicht so schwer, daß man ihretwegen Oberon-2 als erste Lehrsprache ablehnen sollte. Die ersten beiden Einschränkungen fallen in der Grundausbildung ohnehin nicht ins Gewicht, da sie sich erst bei größeren Programmsystemen nachteilig auswirken können. Manche Kritik ergibt sich auch daraus, daß Oberon-2 auf einem anderen Programmiermodell basiert, als man von verbreiteten Sprachen gewohnt ist.