Analog denken – analog programmmieren

Michael Weigend
Institut für Didaktik der Mathematik und der Informatik
Westfälische Wilhelms-Universität Münster
michael.weigend@uni-muenster.de

Aristoteles (384-322) verwendete den Begriff „geometrische Analogie“ zur Beschreibung der Gleichheit zweier quantitativer oder qualitativer Verhältnisse [Co02]. Beispiele: „1 verhält sich zu 2 wie 5 zu 10“ oder „Die Federn sind für den Vogel, was die Schuppen für den Fisch sind“. Eine informatische Modellierung beginnt manchmal mit der Formulierung qualitativer geometrischer Analogien. Ein Beispiel ist die Wiedergabe von Ferne und Nähe in einem Programm mit zweidimensionaler Grafik: (1) Entfernte Objekte befinden sich weiter oben und nahe Objekte weiter unten auf dem Bildschirm. (2) Entfernte Objekte erscheinen klein und nahe Objekte groß. (3) Entfernte Objekte sehen blass und bläulich aus, nahe Objekte haben eine hohe Farbbrillanz (Farbperspektive). Um diese Analogien in das Programm einfließen zu lassen, müssen sie quantifiziert werden. Für die Beschreibung eines plausiblen Zusammenhangs zwischen räumlicher Ausdehnung und vertikaler Position auf dem Bildschirm kann man auf den Strahlensatz zurückgreifen. Die Farbperspektive lässt sich so simulieren, dass über die Abbildung eines Objektes als Filter eine deckungsgleiche blaue Fläche gelegt wird, deren Transparenz in Abhängigkeit von der Position auf der Ebene eingestellt wird. Je „näher“ das Objekt ist, desto durchlässiger ist der Filter [We07].

Gebräuchlich ist die Unterscheidung zwischen analoger und digitaler Repräsentation von Wissen [Ma05]. Eine digitale Repräsentation verwendet Symbole einer Sprache und stellt die abzubildende Entität explizit durch ein Literal dar. So kann die Farbe des Himmels durch das Wort „blau“ oder das RGB-Tupel (30, 30, 250) beschrieben werden. Analoge Repräsentationen gründen auf geometrischen Analogien. Ein hoher Balken in einem Balkendiagramm verhält sich zu einem niedrigen Balken wie eine große Zahl zu einer kleinen. Analoge Repräsentationen werden unmittelbar „erlebt“, während digitale Repräsentationen (unter Rückgriff auf ein Symbolsystem) zuerst kognitiv verarbeitet werden müssen um wahrgenommen zu werden [Ma05]. Graphische Benutzungsoberflächen (GUI) enthalten analoge Ein- und Ausgabemechanismen. Dazu gehören z.B. die Veränderung der Fenstergröße, das Verstellen von Schiebereglern und andere direkte Manipulationen auf dem Bildschirm mit der Maus. Im Informatikunterricht können verschiedene Aspekte analoger Repräsentation von Daten im Rahmen eines Projektes aufgegriffen werden, bei dem die GUI-Gestaltung im Zentrum steht.

Analoges Denken spielt im Zusammenhang mit Polymorphie in der OOP eine Rolle. Eine Form der Polymorphie ist das Überladen von Operatoren. Das bedeutet, dass ein Operator je nach dem Objekt, auf das er angewendet wird, die Ausführung einer anderen Operation bewirkt (z.B. Operator + für Addition von Zahlen und Konkatenation von Strings). Polymorphie kann zu verständlicheren Programmtexten führen. Der kognitive Vorteil liegt darin, dass vertraute Konzepte (wie die Addition) analog auf neue Domänen angewendet werden. Eine Herausforderung im Rahmen eines Programmierprojektes in der Schule ist es, für eine selbst definierte Klasse eine Methode zu erfinden, die einer aus anderen Kontexten bereits bekannten Operation analog ist, und den entsprechenden

Operator zu überladen. Beispiel: Definiere eine Klasse „Color“, die Farben repräsentiert. Implementiere eine plausible Methode für die Addition zweier Farbobjekte.

Bei komplexeren Analogien werden nicht nur einzelne Begriffe oder Größen sondern strukturierte Systeme aus mehreren Komponenten zueinander in Beziehung gesetzt. Beispiele für den Mathematikunterricht findet man in [En97]. Solche strukturellen Analogien werden häufig bei Programmentwicklungen herangezogen, wenn es darum geht, eine grundsätzliche Lösungsidee für ein algorithmisches Problem zu finden. Im agilen Programmieren beginnt eine Softwareentwicklung mit der Formulierung einer Metapher, welche die Idee des gesamten Projektes beschreibt [Be99]. Für das Problem der Suche nach einem Objekt auf einer Fläche können beispielsweise folgende Alltagssituationen herangezogen werden: (1) Eine Gruppe von Polizisten durchkämmt linear ein Waldstück um ein Beweisstück zu suchen. (2) In ein Planschbecken mit schwimmenden Gummienten wirft jemand einen Stein. Es entsteht eine kreisförmige Welle, die sich in alle Richtungen ausbreitet. Die Gummiente, die der Einwurfstelle am nächsten liegt, wackelt zuerst. (3) Ein Wanderer steht auf einem Hügel. Er sucht sein Ziel, indem er sich langsam dreht und den Blick schweifen lässt.

Im Rahmen des Modellbildungsprozesses müssen solche Situationen, die Lösungsideen repräsentieren, zunächst einmal gefunden, dann hinsichtlich Struktur und Dynamik analysiert und im Hinblick auf Brauchbarkeit (Implementierbarkeit, Effizienz etc.) bewertet werden. Schließlich wird ein System entwickelt, das (in gewissen Grenzen) der gewählten Situation analog ist.

Literaturverzeichnis

[Be99] Beck, Kent (1999): Extreme Programming Explained. Addison Wesley, Boston u.a. 1999.

[Co02] Coenen, Hans Georg: Analogie und Metapher. Grundlegung einer Theorie der bildlichen Rede. Walter de Gruyter, Berlin 2002.

[En97] English, Lyn D. (Hrsg.): Mathematical Reasoning. Analogies, Metaphors, and Images. Lawrence Erlbaum Associates, Publishers, Mahwah New Jersey, London 1997.

[Ma05] Matthen, Mohan: Seeing, Doing, and Knowing: A Philosophical Theory of Sense Perception. Oxford University Press, Oxford 2005.

[We07] Weigend, Michael: Verwendung geometrischer und struktureller Analogien bei der Gestaltung von Flash-Animationen – Galerie. URL: http://www.creative-informatics.de/analog/

Workshop

„Didaktik der Informatik – aktuelle Forschungsergebnisse“