TECHNISCHE UNIVERSITÄT DRESDEN

DIL

Fakultät Informatik - Inst. für Software- und Multimediatechnik, AG Didaktik der Informatik/Lehrerbildung

Masterarbeit

Implementierung einer Webanwendung zur Visualisierung von Programmiergrundlagen im Informatikunterricht

Willi Zobel

Matr.-Nr.: 3662805

Studiengang: Medieninformatik

Betreut durch:

Prof. Dr. paed. habil. Steffen Friedrich

Dr. rer. nat. Holger Rohland

Eingereicht am 02. August 2016

Kurzfassung

Einfache Algorithmen bilden die Grundlage für das Programmieren. Obwohl die meisten Funktionen und Abfolgen logisch und trivial sind, bereiten sie vielen Anfängern Schwierigkeiten. Um diese zu beheben, ist 2008 im Rahmen einer Diplomarbeit ein Werkzeug zur Visualisierung solcher grundlegenden Programmstrukturen geschaffen worden, welches in dieser Arbeit technisch und inhaltlich überarbeitet wurde.

Zu Beginn der Arbeit wurde eine Untersuchung des aktuellen Lehrplans für den Informatikunterricht durchgeführt. Dabei wurden die Anforderungen an das zu erstellende Werkzeug erfasst. Im Anschluss wurden aktuell existierende Werkzeuge getestet und evaluiert, die dem Zweck dienen, Grundlagen von Programmierkenntnissen mithilfe von Visualisierungen zu vermitteln. Dadurch konnten die Anforderungen weiter spezifiziert werden.
Im Hauptteil der Arbeit wurde das zu überarbeitende Werkzeug analysiert und mithilfe aktueller Webtechnologien in eine Webanwendung transformiert. Anschließend wurden zusätzliche Module implementiert, die Objekte und verkettete Listen für den Nutzer verständlich visualisieren. Am Ende der Arbeit erfolgte eine Evaluation des Werkzeugs.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Algorithmen haben in der Informatik eine große Bedeutung. Durch festgelegte Schritte können Probleme dadurch strukturiert gelöst werden. Dies ist bei der Programmierung unabdingbar. Jedoch spielen Algorithmen auch in unserem Alltag eine wichtige Rolle. Bei der Bedienung eines Kaffeekochers muss klar definiert werden, in welcher Reihenfolge bestimmte Schritte ausgeführt werden müssen, damit am Ende ein guter Kaffee entsteht. Diese Algorithmen so abzubilden, dass auch Maschinen sie verstehen, bereitet vor allem Anfängern große Schwierigkeiten. Im Informatikunterricht sollte der Vermittlung dieses Themengebiets daher viel Zeit eingeräumt werden.

Das Verständnis von Algorithmen ist eine wichtige Voraussetzung für das Erlernen einer Programmiersprache. Weiterhin müssen, wie auch bei einer Fremdsprache, Vokabular und dessen Syntax-Regeln gelernt werden. Die Schwierigkeit bei Programmiersprachen liegt darin, dass schon kleine Fehler das gesamte Programm zum Absturz bringen können. Der Lernvorgang erfordert somit viel Zeit und eine verständliche und einfache Herangehensweise.

Wie bereits erwähnt, ist der Lernprozess von den Grundlagen der Programmierung sehr zeitintensiv. Bei Anfängern kommt es zu häufigen Verständnisproblemen. Daher gilt es, möglichst einfache und praxisnahe Beispiele zu wählen, um die Motivation und das Verständnis zu erhöhen. Ein weiteres sehr wichtiges Mittel sind Visualisierungen. Durch sie können den SchülerInnen schwer verständliche Algorithmen grafisch deutlich gemacht werden, was den Lernprozess verkürzt und intensiviert.

1.1 Ziel der Arbeit

Ziel dieser Arbeit ist die Erstellung einer Webanwendung, die unterrichtsbegleitend im Informatikunterricht eingesetzt werden kann und den SchülerInnen die Grundlagen der Programmierung mithilfe von Visualisierungen vermittelt. Als Vorlage dient dafür eine Flash-Anwendung, die im Jahr 2008 im Rahmen der Diplomarbeit von Claudia Schindler entstand [11]. Die Umsetzung als Webanwendung ist im Gegensatz zu einer Flash-Anwendung plattformunabhängig und benötigt keine Browser-Erweiterungen. Ein weiterer wichtiger Schritt ist die Überarbeitung der aktuell existierenden Module: Wertzuweisungen, Verzweigungen, Schleifen und Unterprogramme. Ein weiteres Ziel dieser Arbeit ist die Implementierung zusätzlicher Module bezüglich der Arbeit mit verketteten Listen.

Die Webanwendung soll SchülerInnen den Einstieg in die Programmierung erleichtern. Dies soll jedoch nicht nur in Form eines Textes geschehen. Die Motivation und der Spaß beim Lernen sind entscheidend für den Lernprozess. Eine interaktive Umgebung fördert die SchülerInnen und regt sie so zum eigenständigen Nachdenken an. Zudem können mittels Visualisierungen komplizierte Vorgänge auf eine verständliche Weise beschrieben werden. Diese beiden Funktionen, die Interaktivität und die Visualisierungen, sind ein wichtiger Bestandteil der entstehenden Webanwendung.

1.2 Vorgehensweise

Zu Beginn dieser Arbeit wird eine Untersuchung bezüglich des Informatikunterrichts durchgeführt. Dies erfolgt mithilfe des sächsischen Lehrplans und den Prüfungsanforderungen für Informatik. Durch diese wird in Erfahrung gebracht, welche Lerninhalte in welchen Klassenstufen vermittelt werden. Außerdem gibt die Untersuchung Aufschluss über die Zielgruppe der Webanwendung. Im Anschluss wird geklärt, welche Lerninhalte für den Informatikunterricht empfohlen werden. Dafür werden zunächst verschiedene Bewertungsmaßstäbe vorgestellt. Durch die Untersuchung verschiedener Studien und Veröffentlichungen werden die sinnvollen Lerninhalte anschließend bestimmt. Dabei spielen im Besonderen die Algorithmen eine wichtige Rolle. Des Weiteren stellt sich die Frage, auf welche Art und Weise die Lerninhalte vermittelt werden können. Dafür wird geklärt, welche Lehrmaterialien sich für den Informatikunterricht eignen und welche Ansprüche sie besitzen. Abschließend geht es um den Einsatz von Visualisierungen im Unterricht. Dabei werden die verschiedenen Typen von Visualisierungen miteinander verglichen.

Nach der Untersuchung des Informatikunterrichts folgt die Analyse aktueller Werkzeuge, die sich mit der Lehre von Programmiergrundlagen beschäftigen. Dafür werden vorerst Bewertungsgrundlagen festgelegt, die zum Teil auf der *DIN EN ISO 9241* beruhen. Anschließend erfolgt der Test verschiedener Werkzeuge anhand dieser Bewertungsrichtlinien. Der Abschnitt stellt die Konkurrenzanalyse in dieser Arbeit dar. Dadurch kann in Erfahrung gebracht werden, wie sich der aktuelle Markt in diesem Bereich in den letzten Jahren entwickelt hat. Zusätzlich werden dadurch neue Ideen gewonnen, die in die Webanwendung integriert werden können.

Anschließend erfolgt die Bewertung der Werkzeugvorlage, die im Rahmen der Diplomarbeit von Claudia Schindler entstand. Dabei werden die gleichen Bewertungsgrundlagen wie bei den aktuellen Werkzeugen verwendet. Dadurch ist auch hier ein direkter Vergleich möglich. Daraufhin werden die Evaluationsergebnisse, die während der Diplomarbeit entstanden, ausgewertet. Durch die dort aufgezeigten Probleme und Verbesserungsvorschläge kann eine differenziertere Implementierung der Webanwendung stattfinden.

Im Anschluss an die Analyse erfolgt die Implementierung. Diese teilt sich wiederum in die Abschnitte **Entwurf** und **Prototyp**. In der Entwurfsphase werden zuerst die Anforderungen an die Webanwendung gestellt und anschließend nach ihrer Priorität sortiert. Aufgrund dieser Anforderungen werden Szenarien für die verschiedenen Module erstellt. In den Szenarien wird festgelegt, was die Nutzer in diesem Modul erlernen, welche Aufgabe sie erledigen müssen und auf welche Weise die Visualisierung erfolgt. Auf der Basis dieser Szenarien wird im Anschluss der Aufbau der Anwendung definiert und die ersten Mockups erstellt. Im zweiten Teil der Implementierung wird erläutert, welche Anforderungen letztendlich im Prototyp umgesetzt worden sind und welche Änderungen es im Bezug zum Entwurf gab. Anschließend wird geklärt, welche Techniken für die Implementierung verwendet wurden und wie die Webanwendung aufgebaut ist.

Um die Qualität der Anwendung zu bewerten, erfolgt nach der Implementierung eine Evaluation. Dadurch können Fehler, Probleme und Verbesserungsvorschläge aufgezeigt werden. Um im Besonderen auch auf den Einsatz im Informatikunterricht einzugehen, wurden bei der Evaluation LehrerInnen und Studierende zu ihrer Meinung bezüglich des Tools befragt.

Abschließend werden die entstandenen Ergebnisse zusammengefasst und zukünftige Schritte aufgezeigt, die zur Verbesserung und Weiterentwicklung der Webanwendung beitragen.

2 Untersuchung des Informatikunterrichts

2.1 Aktuelle Situation

Am 01.12.1989 wurde in einer Kultusministerkonferenz ein Beschluss zu einheitlichen Prüfungsanforderungen im Fach Informatik gefasst und veröffentlicht. Dieser Beschluss wurde in einer späteren Konferenz (am 05.02.2004) überarbeitet und gilt seit 2007 für alle Bundesländer in den Abiturprüfungen [4].

Im ersten Teil beschäftigt sich der Beschluss mit fachlichen Inhalten und Qualifikationen. Die folgenden fachlichen und methodischen Kompetenzen werden von den SchülerInnen bei der Abiturprüfung erwartet:

1. Erwerb und Strukturierung informatischer Kenntnisse
2. Kennen und Anwenden informatischer Methoden
3. Kommunizieren und Kooperieren
4. Anwenden informatischer Kenntnisse, Bewerten von Sachverhalten und Reflexion von Zusammenhängen

Im ersten Teil der Webanwendung werden den SchülerInnen die Grundlagen bestimmter Programmierkomponenten vermittelt. Im zweiten Teil sollen sie dieses Wissen an einem Beispiel anwenden und es somit vertiefen. Die zweite Kompetenz „Kennen und Anwenden informatischer Methoden“ wird dabei im Gegensatz zur ersten Kompetenz nur zu einem kleinen Teil erworben. Den SchülerInnen sollen laut der Prüfungsanforderung weiterhin die drei folgenden fachlichen Inhalte vermittelt werden:

1. Grundlegende Modellierungstechniken
2. Interaktion mit und von Informatiksystemen
3. Möglichkeiten und Grenzen informatischer Verfahren

Zu den grundlegenden Modellierungstechniken gehört unter anderem die *Modellierung von Abläufen mit Algorithmen*. Dabei geht es um den Algorithmusbegriff, Ablaufstrukturen, einfache und höhere Datenstrukturen, Zerlegen in Teilalgorithmen, Struktogramme und spezielle Verfahren, wie Sortier- oder Suchverfahren. Diese Technik spielt eine wesentliche Rolle bei der Erstellung der Webanwendung. Jedoch werden auch noch weitere Techniken vermittelt. Dazu gehören Objekte aus der *Objektorientierten Modellierung* und das Variablenkonzept aus der *Zustandsorientierten Modellierung*.

Um einen genauen Überblick über den Informatikunterricht zu erhalten, muss auch der aktuelle Lehrplan untersucht werden. In dieser Arbeit wird dabei nur der Lehrplan für Gymnasien in Sachsen betrachtet. Dieser stammt von 2004, wurde jedoch in den Jahren 2007, 2009 und 2011 vom Sächsischen Bildungsinstitut in Zusammenarbeit mit LehrerInnen überarbeitet [7]. Die folgenden fachlichen Ziele sind für das Fach Informatik festgesetzt worden:

- Umgehen mit Daten und Informationen
- Kennen lernen von Aufbau und Funktionalität ausgewählter Informatiksysteme
- Modellieren von Zuständen und Abläufen
- Realisieren von Problemlöseprozessen
- Bewerten von gesellschaftlichen Aspekten der Informatik

Die in dieser Arbeit entstehende Anwendung deckt kein Ziel komplett ab, kann jedoch einige Teilschritte der ersten vier Ziele bearbeiten. Der Informatikunterricht setzt sich aus verschiedenen Prinzipien zusammen. Dazu gehören die Problemorientierung, die Objektorientierung, die Handlungsorientierung und die Projektorientierung. Diese Prinzipien sollten möglichst gleichmäßig im Unterricht und somit auch in der Anwendung genutzt werden.

An diesem Punkt stellt sich die Frage, in welchen Klassenstufen die Webanwendung genutzt werden sollte, um das Wissen entsprechend des Lehrplans zu vermitteln.

In der 7. Klassenstufe geht es hauptsächlich um die Verwendung von Computersystemen. Dazu gehören unter anderem die Anpassung von Benutzeroberflächen, der richtige Umgang mit Computeranwendungen, Grundlagen zum Modell von Informatiksystemen und Techniken zum gezielten Suchen von Inhalten. Die Webanwendung ist an dieser Stelle daher noch nicht sinnvoll einsetzbar.

In der 8. Klassenstufe lernen die SchülerInnen Grundlagen der Objektorientierung kennen. Dabei wird der Zusammenhang von Klasse, Objekt, Attribut und Methode erläutert. Auch wird der Algorithmusbegriff einge-

führt, womit eine Grundlage für die Webanwendung geschaffen wird. Je nachdem, wie tief die LehrerInnen in die jeweiligen Bereiche einsteigen, kann in dieser Klassenstufe die Webanwendung erstmals genutzt werden.

Aufbauend auf dem Algorithmusbegriff lernen die SchülerInnen in der 9. und 10. Klassenstufe die Grundlagen der Programmierung, um einfache Probleme in Programmierumgebungen lösen zu können. Auch die Grenzen der Algorithmen sollen dabei deutlich gemacht werden. Die 9. Klasse ist somit der Haupteinsatzpunkt für die entstehende Webanwendung.

Im Grund- und Leistungskurs der 11. und 12. Klassenstufe wird vertieft in das Themengebiet der Algorithmen und Programmierung eingestiegen. Die SchülerInnen sind an diesem Punkt jedoch schon so weit, dass die Webanwendung hier innerhalb der Unterrichts nicht mehr angewendet wird. Diese soll lediglich die Grundlagen abdecken.

2.2 Empfohlene Lerninhalte

In diesem Abschnitt geht es um die Frage, welche Inhalte im Informatikunterricht vermittelt werden sollten und was man unter sinnvollen Lerninhalten versteht. Zendler et al. beschäftigten sich bereits im Jahre 2006 intensiv mit dieser Fragestellung [13]. Informatikunterricht sollte sich demnach nicht an kurzfristigen, sondern an wesentlichen, alltäglichen und konstanten Entwicklungen orientieren. Es sollte somit nicht darum gehen, konkrete Programmiersprachen zu lehren, sondern die Grundlagen für das Verständnis von Programmiersprachen zu vermitteln. Dabei können konkrete Sprachen jedoch unterstützen. [13]

Es gibt viele Kataloge, die verschiedene Ideen und Konzepte für den Informatikunterricht gesammelt haben. Diese sind jedoch meist sehr spezifisch und auch subjektiv. Zendler et al. haben daher eine empirische Untersuchung durchgeführt, um die Ideen und Konzepte zu bewerten. Dafür schickten sie Fragebögen an verschiedene Professoren für Informatik. Die Fragen waren dabei in vier Kriterien eingeteilt [13]:

Horizontalkriterium
Stellt sicher, dass ein Konzept in unterschiedlichen Bereichen der Disziplin relevant ist.

Vertikalkriterium
Besagt, dass zentrale Inhalte in allen Alterstufen behandelt werden können müssen.

Sinnkriterium
Die Inhalte sollten in ihren Bezügen zur Alltagswelt die Bedeutsamkeit eines Fachs für unsere Kultur aufzeigen können.

Zeitkriterium
Die Inhalte und Konzepte sollten langlebig sein.

Tabelle 2.1: Grundsätze für den Informatikunterricht [9]

Equity	Für alle Schülerinnen und Schüler wird Chancengleichheit gefordert, damit alle optimal gefördert werden können.
Curriculum	Mit einem Curriculum müssen fachlich bedeutende, individuell und gesellschaftlich relevante Inhalte in zusammenhängender Weise dargestellt werden.
Teaching	Dies beinhaltet die Forderung, dass der Unterricht grundsätzlich von hochqualifiziertem Personal durchzuführen ist.
Learning	Hier wird die Bedeutung eines sinnstiftenden Mathematikunterrichts für die Schülerinnen und Schüler betont.
Assessment	Die Beurteilung und Bewertung der Leistungen von Schülerinnen und Schülern basiert darauf, dass Verstehensprozesse und nicht primär Faktenwissen geprüft werden.
Technology	Für den Unterricht ist der Einsatz digitaler Hilfsmittel mittlerweile unentbehrlich, doch es steht grundsätzlich die verantwortungs- und sinnvolle Nutzung von Technik im Zentrum.

Zu den getesteten Konzepten zählen zum Beispiel Prozesse, Algorithmen, Systeme, Netzwerke, Hardware, Software, Kommunikation und Daten. Es stellte sich heraus, dass vor allem die Konzepte Algorithmen, Computer, Daten, Informationen und Tests eine sehr wichtige Rolle im Informatikunterricht spielen sollten. Sie haben in allen vier Bewertungskriterien sehr hohe Werte erhalten. Sie sind in vielen Bereichen der Informatik anwendbar, lassen sich in jeder Klassenstufe vermitteln, können gut mit dem Alltag verknüpft werden und haben eine langfristige Relevanz. [13]

Das Konzept *Algorithmen* sollte somit eine zentrale Rolle im Informatikunterricht spielen. Diese Erkenntnis stützt die Entwicklung der Webanwendung für den Informatikunterricht, da diese grundsätzliche Funktionsweisen von Programmiersprachen und das Verständnis einfacher Algorithmen vermittelt.

Im Zeitkriterium schnitten die Konzepte alle gut ab, so dass wir an einem Punkt angelangt sind, an dem die vermittelten Konzepte konstant und langfristig sind. Im Sinnkriterium gibt es hingegen deutliche Unterschiede. Obwohl die Bewertung bei den Algorithmen sehr gut ausfiel, muss sie letztendlich auch richtig umgesetzt werden. [13]

Durch den Arbeitskreis „Bildungsstandards“ der *Gesellschaft für Informatik e.V.* wurden im Jahr 2008 Bildungsstandards für den Informatikunterricht in der Sekundarstufe I veröffentlicht [9]. In den Empfehlungen mit dem Titel „Grundsätze und Standards für die Informatik in der Schule“ werden Mindeststandards an Kompetenzen definiert, die jede Schülerin und jeder Schüler am Ende der 10.Klasse besitzen sollte.

Die von NCTM (*National Council of Teachers of Mathematics* [5]) formulierten Grundsätze für den Mathematikunterricht wurden in den Bildungsstandards auf das Fach Informatik angepasst (siehe Tabelle 2.1).

Aufgrund der unterschiedlichen technischen Voraussetzungen unter den SchülerInnen ist es wichtig, plattformunabhängige Software zu verwenden [9]. Dies ermöglicht eine Chancengleichheit (siehe Grundsatz „Equity“) beim Lernprozess der SchülerInnen. Das spricht für Webanwendungen, die auf jedem System und in jedem Browser ausgeführt werden können. Auf den Einsatz eines Webservers kann durch die ausschließliche Nutzung von HTML, CSS und Javascript verzichtet werden und ermöglicht somit einen noch leichteren Zugang ohne Installation.

Die Autoren beschreiben in „Grundsätze und Standards für die Informatik in der Schule“ [9] verschiedene Prozess- und Inhaltsbereiche, die in der Sekundarstufe I gelehrt werden sollen:

Prozessbereiche

- Modellieren und Implementieren
- Begründen und Bewerten
- Strukturieren und Vernetzen
- Kommunizieren und Kooperieren
- Darstellen und Interpretieren

Inhaltsbereiche

- Information und Daten
- Algorithmen
- Sprachen und Automaten
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Im Bezug auf die in dieser Arbeit entstehende Webanwendung sind vor allem die Inhaltsbereiche „Informationen und Daten“ und „Algorithmen“ von Bedeutung. Im Abschnitt für die Klassenstufen 8 bis 10 werden verschiedene Empfehlungen im Inhaltsbereich *Daten* festgelegt. Zum einen wird die Notwendigkeit der Datentypen genannt, die bereits beim ersten Umgang mit Programmiersprachen eine wichtige Rolle spielen. Dabei sollte sich der Unterricht zu Beginn dieses Inhaltsbereiches jedoch auf die Basistypen konzentrieren. Auch die Operationen, die auf die Daten ausgeführt werden können, werden in diesem Zusammenhang genannt. Für die entstehende Webanwendung spielt diese Empfehlung insofern eine Rolle, dass ein Modul sowohl für Wertzuweisungen mit unterschiedlichen Datentypen, als auch für Operatoren (arithmetisch & logisch) implementiert werden sollte.

In den Bildungsstandards [9] werden weiterhin Empfehlungen für den Umgang mit Algorithmen im Informatikunterricht gegeben. In den Klassenstufen 5 bis 7 sollten möglichst umgangssprachliche Formulierungen

verwendet werden, um Algorithmen zu beschreiben (z.B. wenn ... dann ...). Zu Beginn fällt es den SchülerInnen oft schwer, Abläufe so zu beschreiben, dass auch Maschinen sie verstehen. Daher wird für diesen Lernprozess viel Zeit benötigt. In den Klassenstufen 8 bis 10 sollte der Übergang zu formalen Beschreibungsformen stattfinden. Die Programmiersprache wird nach der Komplexität, der Anschaulichkeit und den Einsatzmöglichkeiten gewählt. Für den Anfang bietet sich hier eine Pseudosprache an. Einfache Beispiele und gute Visualisierungen unterstützen dabei den Lernprozess. Hier besteht ein guter Ansatzpunkt für die entstehende Webanwendung.

Nachdem im Jahr 2012 Bildungsstandards für die Allgemeine Hochschulreife für die Fächer Deutsch, Mathematik und fortgeführte Fremdsprachen vorgelegt wurden, entschied sich die Gesellschaft für Informatik e.V. einen Arbeitskreis einzurichten, der für die Entwicklung der Bildungsstandards im Informatikunterricht in der Sekundarstufe II zuständig war. Diese Standards wurden Anfang des Jahres 2016 letztendlich veröffentlicht [10].

Wie in „Grundsätzen und Standards für die Informatik in der Schule“ [9] basiert die Struktur der Bildungsstandards für die Sekundarstufe II ebenfalls auf den Prozess- und Inhaltsbereichen. Zusätzlich wurden jedoch Anforderungsbereiche festgelegt. Diese beschreiben die unterschiedlichen kognitiven Ansprüche an informatische Aktivitäten und schlüsseln die Prozessbereiche tiefgründiger auf [10].

Anforderungsbereiche

- Reproduktion
- Reorganisation und Transfer
- Reflexion und Problemlösung

Die Anforderungen an die SchülerInnen werden im Inhaltsbereich *Algorithmen* in der Sekundarstufe II deutlich angehoben. Algorithmen sollen von den SchülerInnen selbst entworfen und durch Programmiersprachen implementiert werden können. Dabei ist auch auf die Verwendung von Softwarebibliotheken zu achten. Zusätzlich sollen gegebene Programme und Problemstellungen analysiert und bewertet werden. Das in dieser Arbeit entstehende Tool soll lediglich die Grundlagen vermitteln und kann ab der 11. Klasse nur für die Wiederholung einfacher Algorithmen dienen. Eventuell kann die Webanwendung in Zukunft jedoch so erweitert werden, dass sie den SchülerInnen auch komplexere Themen vermitteln kann.

Die Informationstechnik ist unter den SchülerInnen heutzutage so weit verbreitet wie nie zu vor. Man könnte deshalb meinen, dass die Kinder heute ein gutes technisches Grundwissen haben. Ein Zitat aus einer Arbeit von Helmut Witten widerspricht dieser These jedoch:

> *„Aber Technik zu verwenden und sie zu verstehen ist nicht dasselbe. Die Computertechnik, zum Beispiel, ist heute zwar leichter zu benutzen als vor 20 Jahren, doch schwerer zu begreifen. Die Benutzeroberflächen versiegeln die Technik gegen den Zugriff des Laien.“*
> (H. Witten [12])

Wie man an der ubiquitären Softwareentwicklung erkennen kann, werden die heutigen Informationstechniken so gebaut, dass der Benutzer von der Funktionsweise wenig mitbekommt. Man muss das Gerät kaum noch verstehen, um es nutzen zu können. An dieser Stelle setzt der Informatikunterricht ein. Alltägliche Themen bieten sich als Schnittstelle dafür an. Auch fächerübergreifender Unterricht, zum Beispiel bei der Bildbearbeitung oder der Verschlüsselung im Zusammenhang mit der Mathematik, ist hierbei möglich [12].

2.3 Lehrmaterialien

In den letzten Abschnitten ging es um die Inhalte des Informatikunterrichtes. Im Anschluss daran stellt sich die Frage, auf welche Weise diese Inhalte den SchülerInnen vermittelt werden können. In der Vergangenheit wurden viele Lehrmaterialien veröffentlicht, welche die LehrerInnen im Informatikunterricht unterstützen sollen. Selten kommt es jedoch dazu, dass die Materialien letztendlich genutzt werden. Diethelm et al. nennen in ihrer Arbeit als möglichen Grund die fehlende Lehrerperspektive [1]. Bei der Erstellung der Lehrmaterialien wird oft davon ausgegangen, dass die LehrerInnen die Einsatzmöglichkeiten selbst erkennen. Neben der Lehrerperspektive gibt es jedoch noch weitere Perspektiven, nach denen man sich bei der Erstellung von Lehrmaterialien richten sollte [1]:

Gesellschaftliche Ansprüche
Hierbei geht es um Ansprüche durch die Gesellschaft, wie zum Beispiel Medienkompetenzen. Es sollen allgemeinbildende Informationen gelehrt werden.

Auswahl informatischer Systeme
Der Einfluss von Informationstechnologien wird immer größer, die dahinter stehenden Informatiksysteme bleiben jedoch unsichtbar. Es könnte Software erzeugt werden, um Systeme verstehen zu können, welche die SchülerInnen selbst verwenden.

> *„Unter einem Informatischen Phänomen verstehen wir ein Ereignis, das durch automatisierte Informationsverarbeitung verursacht wird und im realen oder mentalen Handlungsumfeld der Schülerinnen und Schüler stattfindet.“*
> (I. Diethelm [1])

Fachliche Klärung
In welche fachliche Tiefe soll das Material gehen und welche Modelle können dafür verwendet werden?

Schülervorstellungen
Auch die Vorstellungen der SchülerInnen spielen eine wichtige Rolle. Wie stellen sie sich die Lehrmaterialien vor und welche fachlichen Voraussetzungen bringen sie bereits mit? Die Beispiele, mit welchen gearbeitet wird, sollten dabei möglichst aus ihrem Alltag stammen.

Lehrerperspektiven
Die Lehrmaterialien sollten sich möglichst an den Erklärungsmustern der LehrerInnen orientieren. Umfangreiche Beschreibungen für die Einsatzmöglichkeiten und Unterrichtsziele sollten ebenfalls gegeben sein, so dass die Akzeptanz durch die LehrerInnen gesteigert werden kann.

2.4 Visualisierungen

In diesem Abschnitt soll es um den Einsatz von Visualisierungen im Informatikunterricht gehen. Für die entstehende Webanwendung sind die Visualisierungen von Algorithmen und Programmiergrundlagen ein Kernbestandteil. Doch in welcher Form unterstützen visuelle Darstellungen den Lernprozess von SchülerInnen? Mit dieser Fragestellung hat sich Nils Faltin in seiner Dissertation auseinandergesetzt [2].

Bei Visualisierungen ist es wichtig, dass sie einfach und verständlich gehalten werden. Es gibt verschiedene Visualisierungsformen wie Bilder, Bilderfolgen, Filme oder Animationen. Die Animationen können im Vergleich zu Filmen interaktiv gesteuert werden. Herr Faltin untersuchte die Auswirkungen der verschiedenen Visualisierungsformen auf den Lernerfolg und kam zu dem Schluss, dass es keine signifikanten Unterschiede gibt. Man kann den SchülerInnen somit auch mit statischen Bildern informationstechnische Prozesse gut erklären. [2]

Ein wesentlich wichtigerer Faktor ist das entdeckende Lernen. Herr Faltin stellte hierbei einen signifikanten Unterschied bei der Effektivität des Lernprozesses fest. SchülerInnen, die direkt am Lernprozess beteiligt werden, verstehen komplexe Zusammenhänge wesentlich schneller, als wenn sie nur etwas präsentiert bekommen.

> *„Das entdeckende Lernen appelliert an die Neugier des Lerners und an den Wunsch, sich als kompetent zu erweisen. In diesem Sinne fördert es die intrinsische Motivation, d.h. ein Lernen aus Interesse an der Sache und nicht aus äußerem Druck.“*
> (N. Faltin [2])

Die SchülerInnen kann man am einfachsten bei der Visualisierung mit Animationen einbeziehen. Dabei sollte jedoch darauf geachtet werden, dass sie nicht in eine Sackgasse geraten. Unterstützungsmöglichkeiten, wie zum Beispiel eine Musterlösung, müssen angeboten werden, um bei den SchülerInnen keine Resignation hervorzurufen.

3 Analyse von Werkzeugen

3.1 Aktuelle Werkzeuge

Um die Werkzeugvorlage als Webanwendung zu implementieren, sollen in diesem Kapitel aktuelle Werkzeuge vorgestellt und miteinander verglichen werden. Zuerst gab es die Überlegung, Werkzeuge aus unterschiedlichen Kategorien (Mobil, Desktop, Webanwendung, ...) zu vergleichen. Da jedoch jede Kategorie andere Bewertungsmaßstäbe enthält, fiel die Entscheidung darauf, nur webbasierte Werkzeuge zu testen und miteinander zu vergleichen. Diese können zusätzlich besser als Vorlage für die entstehende Anwendung genutzt werden.

Um eine einheitliche Bewertungsgrundlage für die ausgewählten Werkzeuge zu schaffen, wird zur Bewertung der Teil 110 (ehemals Teil 10) der Normenreihe DIN EN ISO 9241 verwendet. Diese geht in die abschließende Bewertung mit 50% ein. Damit jedoch nicht nur die Dialoggestaltung, sondern auch der Einsatz im Informatikunterricht bewertet werden kann, wurden zusätzliche Bewertungskriterien geschaffen. Diese ergeben die restlichen 50% der Endbewertung.

3.1.1 DIN EN ISO 9241 - Teil 110

Die Norm EN ISO 9241 wird auch „Ergonomie der Mensch-System-Interaktion“ genannt und beschreibt die Anforderungen an die Arbeitsumgebung, die Hardware und die Software. Der Teil 110 dieser Norm stellt die Grundsätze der Dialoggestaltung auf und beschreibt damit die ergonomische Gestaltung von interaktiven Systemen. Als Dialog wird die *„Interaktion zwischen einem Benutzer und einem interaktiven System als Folge von Handlungen des Benutzers (Eingaben) und Antworten des interaktiven Systems (Ausgaben), um ein Ziel zu erreichen“* verstanden. Die Grundsätze sind unabhängig von der Dialogtechnik, der Arbeitsumgebung und der

Anwendung und sollen den Benutzer vor den folgenden Problemen schützen [6]:

- unnötigen Schritten, die nicht als Teil der Arbeitsaufgabe erforderlich sind
- irreführenden Informationen und Darstellungen
- unerwarteten Reaktionen des Systems
- Einschränkungen bei der Navigation durch das System
- ineffizienten Behebungen von Fehlern

Um diese Probleme zu vermeiden, wurden die folgenden sieben Grundsätze aufgestellt:

A. Aufgabenangemessenheit

> *Ein interaktives System ist aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h., wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie.* [6]

Bewertungskriterien:

- Fernhaltung von internen Aufgaben
- Aufgabenorientierte Informationseingabe und Informationsausgabe
- Verwendung von Makros, Voreinstellungen und Shortcuts
- Speichern von ausgefüllten Daten

B. Selbstbeschreibungsfähigkeit

> *Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für den Benutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle im Dialog er sich befindet, welche Handlungen unternommen werden können und wie diese ausgeführt werden können.* [6]

Bewertungskriterien:

- Erklärung von Schritten
- Eindeutige Bezeichnungen
- Adaptive Hilfe
- Rückmeldung zum Bearbeitungsstand
- Überblickinformationen

C. Erwartungskonformität

Ein Dialog ist erwartungskonform, wenn er den aus dem Nutzungskontext heraus vorhersehbaren Benutzerbelangen sowie allgemein anerkannten Konventionen entspricht. [6]

Bewertungskriterien:

- Einheitliche Dialogverhalten
- Einheitliche Darstellung gleicher Bedienelemente

D. Lernförderlichkeit

Ein Dialog ist lernförderlich, wenn er den Benutzer beim Erlernen der Nutzung des interaktiven Systems unterstützt und anleitet. [6]

Bewertungskriterien:

- Unterstützung des Nutzers
- Konsistenz

E. Steuerbarkeit

Ein Dialog ist steuerbar, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist. [6]

Bewertungskriterien:

- Steuerung der Richtung durch den Benutzer
- Steuerung der Geschwindigkeit durch den Benutzer
- Schritte können rückgängig gemacht werden

F. Fehlertoleranz

Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Benutzers erreicht werden kann. [6]

Bewertungskriterien:

- Abfangen von Fehlern
- Korrekturhinweise
- Angebotene Hilfe

G. Individualisierbarkeit

> *Ein Dialog ist individualisierbar, wenn Benutzer die Mensch-System-Interaktion und die Darstellung von Informationen ändern können, um diese an ihre individuellen Fähigkeiten und Bedürfnisse anzupassen.* [6]

Bewertungskriterien:

- Anpassung vom Umfang der Erläuterungen
- Anpassung der Darstellung

3.1.2 Aufgabenspezifische Bewertungskriterien

Bei dem Einsatz einer Lernsoftware im Informatikunterricht muss auf spezielle Bewertungskriterien eingegangen werden. Zum einen ist die Anwendung für SchülerInnen gedacht, die mit den Grundlagen der Programmierung noch nicht in Kontakt gekommen sind. Daher muss der **Anspruch** an die Benutzer des zu testenden Werkzeugs möglichst gering sein.

Ein weiterer wichtiger Punkt ist das **Anwendungsgebiet**. Dabei ist zu beachten, dass den SchülerInnen keine speziellen Programmiersprachen beigebracht, sondern die Grundlagen der Programmierung vermittelt werden sollen. Dies wird durch einfachen Pseudo-Code ermöglicht und bildet die Basis für die Vertiefung in die Programmierung.

Zuletzt spielt auch die **Registrierung** bei der Anwendung eine Rolle. Für den Informatikunterricht ist es von Vorteil, wenn keine Registrierung notwendig ist bzw. vorausgesetzt wird. Auch die preislichen Aspekte und die technischen Voraussetzungen werden in dieser Kategorie berücksichtigt.

3.1.3 Test der Werkzeuge

Um die zu erstellende Webanwendung auf aktuelle und fundierte Entwicklungen zu stützen, werden in diesem Abschnitt acht verschiedene Lernplattformen getestet. Die Bewertung der Werkzeuge richtet sich dabei nach

den Bedingungen dieser Arbeit. Es geht dabei nicht darum, das Werkzeug allgemeingültig zu bewerten. Der Test bezieht sich auf die mögliche Verwendung im Informatikunterricht, um den SchülerInnen die Grundlagen der Programmierung zu vermitteln. Dabei wurde hauptsächlich darauf geachtet, welche Strategien das Werkzeug verwendet, um Lerninhalte effizient zu vermitteln. Der Umfang des Angebots hat bei der Bewertung daher eine niedrigere Priorität.

A. Codecademy

Das Unternehmen *Codecademy* [20] wurde im August 2011 von Zach Sims und Ryan Bubinski gegründet. Die beiden damaligen Studenten setzten sich zum Ziel, eine Plattform zu entwickeln, auf der Menschen gewünschte Fähigkeiten mithilfe digitaler Techniken erlernen können.

Die Registrierung erfolgt durch eine E-Mail Adresse, Facebook oder Google und ist kostenlos. Es steht jedoch ein kostenpflichtiges Premium-Model zur Verfügung, welches die Funktionalitäten erweitert. Zusätzlich bietet *Codecademy* verschiedene Ressourcen für LehrerInnen. Zum einen können Stundenpläne für den Informatikunterricht heruntergeladen werden, die das effektive Lernen mithilfe von *Codecademy* beschreiben. Zum anderen können LehrerInnen ihre SchülerInnen hinzufügen und erhalten somit eine Übersicht vom Wissensstand ihrer gesamten Klasse. Dadurch können LehrerInnen schnell die Probleme der SchülerInnen erkennen und sie somit rechtzeitig unterstützen.

Codecademy bietet verschiedene Kurse zur Webentwicklung und Programmierung an. Diese sind sehr spezifisch einer Programmiersprache oder einer Technologie zugeordnet. Es gibt bisher keine Kurse für grundlegende Algorithmen bei der Programmierung. Die angebotenen Kurse teilen sich in die Kategorien „Web Developer Skills“, „Language Skills“ und „Goals“. In der Tabelle 3.1 werden Beispielkurse für die einzelnen Kategorien aufgezählt.

Tabelle 3.1: Kurse bei Codecademy

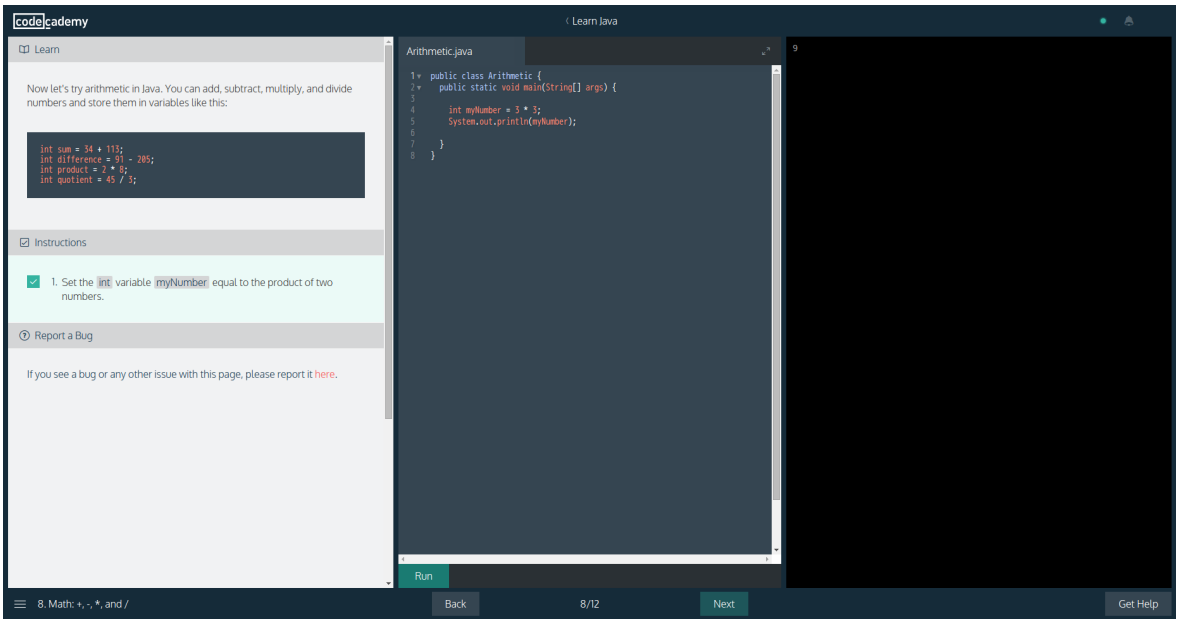
Web Developer Skills	Language Skills	Goals
• Learn Git • Learn Java • Learn SQL • Make a Website • ...	• HTML & CSS • JavaScript • PHP • Python • ...	• Animate your Name • Make a website all about you • Build your own Galaxy

Die Kurse bestehen aus verschiedenen Lerneinheiten. Diese können wiederum Aufgaben (Lessons), Quiz und Projekte enthalten. Ohne Premium-Zugang stehen einem jedoch nur die Aufgaben zur Verfügung. Daher wird

im folgenden speziell auf diese eingegangen.

Eine Aufgabe besteht aus mehreren Schritten. Jeder Schritt enthält verschiedene Teilaufgaben, die der Nutzer lösen muss, um zum nächsten Schritt zu gelangen.

Abbildung 3.1: Codecademy - Ansicht einer Aufgabe / Lesson



Wie in der Abbildung 3.1 dargestellt ist, besteht die Oberfläche einer Aufgabe aus vier Teilen. Auf der linken Seite wird der aktuelle Schritt erklärt und die Teilaufgaben aufgelistet. Dabei kann man sehen, welche Aufgaben man bereits abgeschlossen hat. In der mittleren Spalte befindet sich ein Editor, in den man seine Lösungen einträgt. Somit muss der Benutzer von Beginn an mit dem Quellcode arbeiten, was vor allem für Anfänger schwierig ist. Unter dem Editor kann der eingegebenen Code anschließend ausgeführt werden. Die Ausgabe erfolgt auf der rechten Seite der Oberfläche. Der Code wird dabei komplett ausgewertet und kann nicht schrittweise analysiert werden. Fehler in der Lösung werden vom System erkannt und dem Benutzer angezeigt. Syntaxfehler werden jedoch durch Compiler-Ausgaben in der Konsole ausgegeben, die fortgeschrittenes Verständnis benötigen.

An der unteren Seite der Oberfläche befindet sich eine Navigationsleiste. Dort wird die aktuelle Position in der Aufgabe dargestellt und man kann zwischen den einzelnen Schritten wechseln. Zum nächsten Schritt kann man jedoch erst gehen, wenn man die aktuellen Teilaufgaben vollständig abgeschlossen hat. Über die Navigation gelangt man außerdem zur Übersicht der aktuellen Schritte und der Hilfe. Die gelösten Teilaufgaben werden gespeichert und können zurückgesetzt werden.

In der Tabelle 3.2 sind die Vor- und Nachteile von *Codecademy* zusammengefasst. Bei der Gesamtbewertung von 70% (Tabelle 3.12) sind vor allem die aufgabenspezifischen Kriterien hervorzuheben, da sich *Codecademy*

nur eingeschränkt als Einstieg in die Programmierung im Informatikunterricht eignet. Durch die einfache Bedienung und Gestaltung der Kurse kann *Codecademy* gut in der Kategorie Erwartungskonformität abschließen.

Tabelle 3.2: Codecademy: Vor- und Nachteile

Vorteile	Nachteile
• Eingegebene Lösungen werden dauerhaft gespeichert • Gute Übersicht von angefangenen und abgeschlossenen Aufgaben • Aufgabenschritte werden gut erklärt • Kurse sind gleich aufgebaut, so dass man sich nach kurzer Eingewöhnungszeit schnell zurechtfindet • Größe der Arbeitsflächen kann angepasst werden • Verwaltung von SchülerInnen durch LehrerInnen möglich	• Das Verstehen von Quellcode wird vorausgesetzt • Es werden nur Kurse zu speziellen Sprachen und Techniken angeboten • Compiler-Fehler werden dem Nutzer durch komplizierte Konsolenausgaben präsentiert • Funktion „Schritte zurückzusetzen“ ist in der Hilfe versteckt

B. Code

Code.org [18] ist eine Non-Profit Organisation aus den USA, die sich im Jahr 2013 gründete, um Menschen den Zugang zur Informatik zu erleichtern. Ihr Ziel ist es, jedem Schüler und jeder Schülerin Informatikunterricht anzubieten. Dabei geht es im Besonderen auch darum, junge Frauen für dieses Fach zu begeistern.

Bei *Code.org* kann man sich kostenlos per E-Mail, Google, Facebook und Microsoft registrieren und damit den gesamten Funktionsumfang nutzen. Bei der Registrierung muss angegeben werden, ob man die Plattform als LehrerIn oder als SchülerIn nutzen möchte. Als LehrerIn werden einem neben der Kursübersicht zusätzliche Informationen bereitgestellt. Dazu gehören unter anderem Stundenpläne, Lehrmaterialien und eine Übersicht von Konten der eigenen SchülerInnen und deren Fortschritt. Um *Code.org* im Informatikunterricht nutzen zu können, besteht für LehrerInnen die Möglichkeit, Bereiche mit unterschiedlichen Aufgaben und Kursen zu erstellen. Zu diesen Bereichen können sie anschließend SchülerInnen hinzufügen. Dabei legen die LehrerInnen selbst die Zugangsdaten fest, so dass sich die SchülerInnen nicht selbst registrieren müssen. In einer Zusammenfassung erfahren die zuständigen LehrerInnen den aktuellen Fortschritt ihrer SchülerInnen. Diese Funktion ist sehr übersichtlich gestaltet.

Die Kurse sind nach dem Schema der blockbasierten Programmierung aufgebaut. Somit müssen die Benutzer keinen Quellcode schreiben, sondern können die einzelnen Funktionen durch *Drag & Drop* zusammenset-

zen. Dadurch beginnen die Kurse bereits ab einem Alter von 4 Jahren. Bei jedem Kurs sind das empfohlene Alter und die Voraussetzungen gekennzeichnet. Für fortgeschrittene Benutzer gibt es die Möglichkeit, einen vereinfachten Quellcode vom Lösungsvorschlag anzeigen zu lassen. Dabei werden nicht die Funktionsinhalte, sondern nur die Funktionsaufrufe und Schleifen in JavaScript dargestellt. In der Tabelle 3.3 werden ein Teil der angebotenen Kurse aufgelistet.

Tabelle 3.3: Kurse bei Code.org

Beginner	**Hour of Code**
• Kurs 1 - ab 4 Jahre (Vorschulkinder) - Der Benutzer erlernt hier das zusammensetzen von Blöcken, als Voraussetzung für die nächsten Kurse • Kurs 2 - ab 6 Jahren (Lesen erforderlich) • Kurs 3 - ab 8 Jahren • ...	• Star Wars - Lerne Droiden zu programmieren und erschaffe dein eigenes Star Wars Spiel • Minecraft - Entdecke die Welt von Minecraft • Künstler - Zeichne coole Bilder • ...

Die angebotenen Tutorien ermöglichen den sehr frühen Zugang zum Erlernen von einfachen Algorithmen. Dabei werden bekannte Filme und Spiele genutzt, um die Motivation bei den SchülerInnen anzuregen. Jeder Kurs wird durch Videomaterial unterstützt, welches bisher jedoch nur auf Englisch verfügbar ist. Ein Kurs besteht aus mehreren Aufgaben, die der Benutzer in beliebiger Reihenfolge abarbeiten kann.

In der Abbildung 3.2 ist eine Aufgabe aus dem Kurs „Minecraft“ dargestellt. Auf der linken Seite befindet sich die Ansicht für das Ausführungsergebnis. Dort kann die Ausführung ebenfalls gestartet und zurückgesetzt werden. Das manuelle Wechseln zwischen den Teilschritten ist nicht möglich. In der mittleren Spalte befindet sich eine Auflistung aller Funktionsblöcke, die für diese Aufgabe relevant sind. Dabei wird zwischen statischen (z.B. „vorwärts bewegen“), dynamischen (z.B. „Um $X°$ nach links drehen“) und Schleifen-Blöcken („Wiederhole X mal“) unterschieden. Auf der rechten Seite der Oberfläche befindet sich der Arbeitsbereich, auf dem man beliebige Blöcke zusammensetzen kann. Alle Funktionsblöcke, die an dem Block „beim Ausführen“ angeschlossen sind, werden beim Klick auf „Ausführen“ in der Vorschau abgespielt. Somit kann man auf eine einfache Weise komplexe Anwendungen erschaffen.

Code.org erhält mit 82% eine gute Bewertung (Tabelle 3.12), welche insbesondere durch den niedrigen Anspruch und dem offenen Anwendungsgebiet entsteht. Zusätzlich bietet diese Plattform gute Features, wie die Auswahl der Schwierigkeitsstufe oder das Lehrkraft-Portal mit vielen zusätzlichen Informationen. Die Anzeige des Codes ist für Fortgeschrittene etwas zu simpel gehalten. An dieser Stelle könnten Funktionen genauer beschrieben werden. Alle Vor- und Nachteile sind in der Tabelle 3.4 zusammengefasst.

Abbildung 3.2: Code - Ansicht einer Aufgabe

Tabelle 3.4: Code: Vor- und Nachteile

Vorteile	Nachteile
• Anmeldung als SchülerIn oder LehrerIn möglich • Kurse bereits ab einem Alter von 4 Jahren • Keine konkrete Programmiersprache, sondern blockbasierte Programmierung • Aufgaben können jederzeit neu gestartet werden • Gute Hilfestellungen bei falschen Angaben • Zusätzliche Informationen durch Videos • Nutzer kann teilweise selbst die Schwierigkeiten in den Aufgaben festlegen	• Geschwindigkeit der Ausführung kann nur in wenigen Kursen gesteuert werden • Bedienelemente sind teilweise den einzelnen Tutorien angepasst, was zu Verständnisproblemen führen kann • Genauere Erläuterungen des Codes für Fortgeschrittene nicht vorhanden • Hilfe nur bei Fehlerhaften Angaben

C. Learn X

Die Anwendung *Learn X* besteht aus interaktiven Tutorials zu unterschiedlichen Programmiersprachen. Da jedes Tutorial einen eigenen Namen hat, wurde der Name „Learn X“ für diese Masterarbeit ausgewählt, um

alle verschiedenen Tutorien zusammenzufassen. Das X im Namen kann somit durch die folgenden Programmiersprachen ersetzt werden: Python [33], Java [30], C [28], JavaScript [31], PHP [32], Shell [34] und C# [29].

Um *Learn X* zu verwenden ist keine Registrierung notwendig. Dadurch werden jedoch auch keine Lösungen gespeichert und man erhält keine Übersicht, welche Aufgaben man bereits abgeschlossen hat. Die Tutorials für die einzelnen Programmiersprachen besitzen einen ähnlichen Aufbau, der in der Tabelle 3.5 auszugsweise dargestellt ist.

Tabelle 3.5: Beispielhafte Lerninhalte in den Tutorials bei Learn X

Learn the Basics	**Advanced Tutorials**
• Hello, World! • Variablen • Arrays • Operationen • Schleifen • ...	• Vererbung • Exceptions • Interfaces • Collections • ...

Ein Tutorial beschreibt anhand von Texten und Codezeilen bestimmte Funktionen einer Programmiersprache (siehe Abbildung 3.3). Zwischen den verschiedenen Tutorials kann beliebig gewechselt werden. Am unteren Rand der Oberfläche befindet sich ein Code- und Ausgabebereich, der zur besseren Übersicht auch minimiert werden kann. Dort kann man einen beliebigen Code ausführen und die Ausgaben betrachten. Unter jedem Codeblock im Tutorial befindet sich eine Schaltfläche zum Ausführen des Codes. Dies ermöglicht das schnelle Testen von Beispielen. Am Ende jedes Tutorials befindet sich eine Aufgabe, die ebenfalls im Editor eingegeben werden muss. Wenn die korrekte Ausgabe erscheint, wird eine Erfolgsmeldung angezeigt und man kann direkt zum nächsten Tutorial springen. Da lediglich die Ausgabe und nicht der Quellcode überprüft wird, kann es jedoch zu Problemen kommen, die exakte Lösung zu finden. Man kann sich jedoch einen Lösungsvorschlag und die erwartete Ausgabe anzeigen lassen. Hinweise zu fehlerhaften Eingaben werden nicht angezeigt. Compiler-Fehler werden zwar in der Ausgabe dargestellt, können jedoch durch Anfänger nur schwer behoben werden.

Learn X landet mit einer Endbewertung von 67% auf dem vorletzten Platz der bewerteten Anwendungen (siehe Tabelle 3.12). Dies wird vor allem durch die mangelhaften Hilfestellungen während der Bearbeitung der Tutorien verursacht. Zusätzlich ist *Learn X* nicht für Anfänger geeignet, da das Verstehen von Quelltext der entsprechenden Sprachen vorausgesetzt wird. In der Tabelle 3.6 werden alle Vor- und Nachteile der Anwendung aufgelistet.

Abbildung 3.3: Learn X - Ansicht eines Tutorials (Learn JS [31])

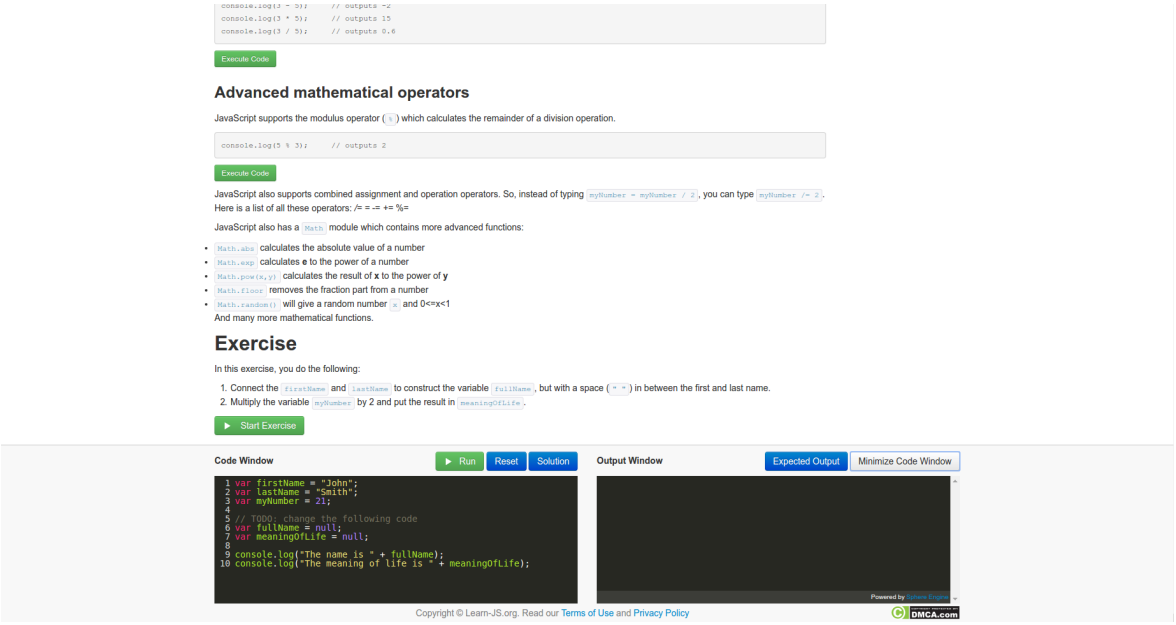


Tabelle 3.6: Learn X: Vor- und Nachteile

Vorteile	Nachteile
• Keine Registrierung nötig • Die Tutorials sind einheitlich aufgebaut und einfach beschrieben • Aufgaben können während der Bearbeitung zurückgesetzt werden	• Das Verstehen von Quellcode wird vorausgesetzt • Nur spezielle Programmiersprachen werden unterstützt • Eingegebene Lösungen bleiben nicht erhalten • Das Ausführen des Quellcodes dauert sehr lange • Kein Überblick über absolvierte Tutorials • Unzureichende Hilfe

D. Coding Dojo

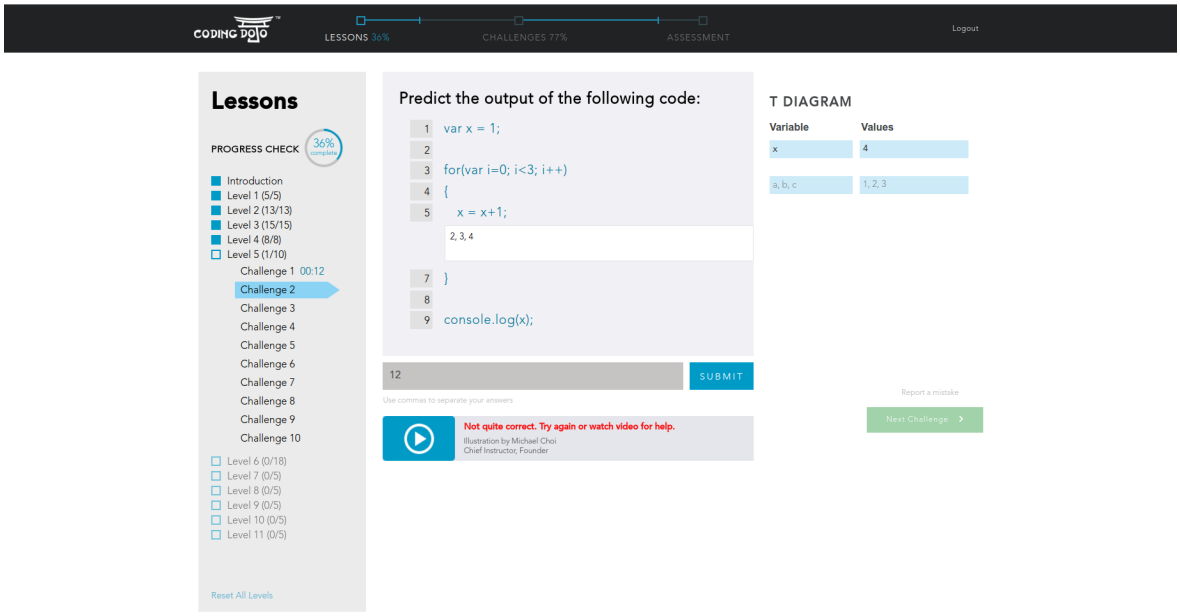
Bei „Coding Dojos“ handelt es sich um Veranstaltungen, bei denen es darum geht, typische Problemstellungen mithilfe der Programmierung in festgelegten Zeiträumen zu lösen. Die Webseite von *Coding Dojo* [21] veranstaltet dafür seit 2012 regelmäßige „Bootcamps“, in denen verschiedene Teams gegeneinander antreten.

Auf ihrer Webseite bieten sie zusätzlich eine Algorithmus-Plattform an [22], auf der Benutzer verschiedene Aufgaben zu Algorithmen lösen können.

Die Registrierung auf der Plattform erfolgt lediglich über die Eingabe einer E-Mail Adresse. Da diese jedoch nicht bestätigt werden muss und es auch keine Passwortvergabe gibt, kann sich anschließend jeder mit dieser E-Mail Adresse anmelden und die Aufgaben weiter bearbeiten. Auf der Plattform wird zwischen Lessons, Challenges und Assessment unterschieden. Diese Bereiche sollen in den folgenden Abschnitten näher erläutert werden.

Die **Lessons** sind in verschiedene Level aufgeteilt, welche wiederum aus mehreren Aufgaben bestehen. Die nächste Aufgabe kann erst erledigt werden, wenn die vorherigen abgeschlossen wurden. In der Abbildung 3.4 ist die Ansicht einer Lesson dargestellt. Die Aufgaben bestehen darin, die Ausgaben, die durch einen vorgegebenen Quellcode erzeugt werden, vorherzusagen. Dabei werden verschiedene Algorithmen wie Array-Operationen, Schleifen und Bedingungen betrachtet. Um den Überblick nicht zu verlieren, steht dem Benutzer unter jeder Zeile des Quellcodes ein Textfeld zur Verfügung, in dem er eigene Notizen schreiben kann. Zusätzlich wird auf der rechten Seite eine Tabelle bereitgestellt, in der die Werte von gegebenen Variablen notiert werden können. Wenn der Benutzer seine Antwort abschickt, erscheint ein Hinweis, ob diese korrekt oder falsch ist. Zusätzlich wird ein Video mit einem Lösungsvorschlag angezeigt. Sonstige Hilfestellungen stehen nicht zur Verfügung. Dadurch sind die Aufgaben für Anfänger ungeeignet. An dieser Stelle wird deutlich, dass die „Algorithm Platform“ nicht dazu gedacht ist die Grundlagen der Programmierung zu lernen, sondern seine bestehenden Fähigkeiten zu testen und mit anderen zu messen.

Abbildung 3.4: Coding Dojo - Algorithm Platform - Ansicht einer Aufgabe



In den **Challenges** geht es darum, verschiedene Probleme mithilfe der Programmierung zu lösen. Dafür steht dem Benutzer ein Editor zu Verfügung, in dem er eine vorgegebene Funktion so erweitern muss, dass sie die korrekte Rückgabe erzeugt. Die Ausgabe kann jederzeit überprüft werden. Die Aufgaben bestehen zum Beispiel darin, das Maximum innerhalb eines gegebenen Arrays zu finden. Der Lösungsweg ist dem Nutzer dabei freigestellt, für das Ergebnis zählt nur der Rückgabewert der Funktion. Auch bei den Challenges stehen lediglich die Lösungsvideos zur Verfügung.

Im **Assessment**-Abschnitt erhält der Benutzer abschließend eine Bewertung seiner Fähigkeiten. Dafür werden ihm acht verschiedene Aufgaben präsentiert, die sowohl aus Lessons, als auch aus Challenges bestehen. Dieser abschließende Test kann jederzeit gestartet und wiederholt werden. Die Lösungsvideos stehen dem Benutzer hierbei nicht zur Verfügung.

Tabelle 3.7: Coding Dojo: Vor- und Nachteile

Vorteile	Nachteile
• Allgemeines Verständnis von Programmierung wird durch die Verwendung von JavaScript erzeugt • Eingegebene Lösungen werden dauerhaft gespeichert • Gute Übersicht über Bearbeitungsstand • Lösungsvorschlag durch Video • Gute Möglichkeiten für eigene Notizen	• Anmeldung nur mit E-Mail ohne Passwort • Keine Erklärungen zu Aufgaben, sehr schwer für Anfänger • Keine Anpassung der Darstellung möglich • Wenig Hilfemöglichkeiten gegeben

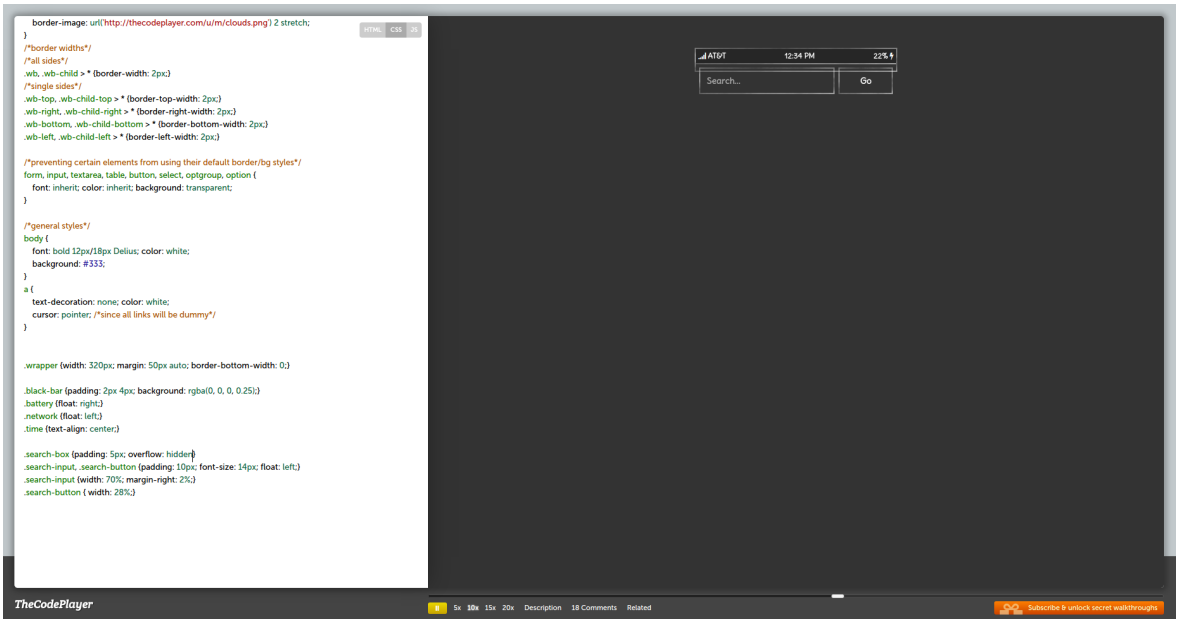
In der Tabelle 3.7 sind die Vor- und Nachteile von Coding Dojo [22] aufgelistet. Da keine Beschreibungen für Anfänger vorhanden sind und die Plattform nicht für das Lehren von Grundlagen der Programmierung konzipiert wurde, erhält diese in der Kategorie „Aufgabenspezifische Kriterien“ nur 53%. Dies wirkt sich stark auf die Endbewertung von 68% aus (siehe Tabelle 3.12).

E. TheCodePlayer

TheCodePlayer [37] ist eine seit 2012 bestehende Plattform, die das Ziel hat, dem Benutzer das Erlernen von Frontend-Technologien zu ermöglichen und zu vereinfachen. Auf der Startseite gibt es dafür die Möglichkeit, zwischen verschiedenen Video-Walkthroughs zu wählen. Die meisten sind ohne Registrierung zugänglich.

In einem Walkthrough kann man beobachten, wie jemand nacheinander ein bestimmtes Feature mithilfe von CSS, JavaScript und HTML entwickelt (siehe Abbildung 3.5). Die Geschwindigkeit kann dabei vom Benutzer am unteren Rand mit vier verschiedenen Stufen geregelt werden. Zusätzlich kann der Walkthrough jederzeit

Abbildung 3.5: TheCodePlayer - Ansicht eines Walkthroughs



pausiert werden, um die aktuellen Änderungen nachvollziehen zu können. Auf der rechten Seite der Oberfläche wird die Vorschau des zu entwickelnden Features angezeigt.

Bei dem Walkthrough handelt es sich nicht um ein einfaches Video, da der Quellcode vom Benutzer jederzeit bearbeitet werden kann. Die einzelnen Bereiche für den Quellcode von HTML, CSS und JavaScript können ein- und ausgeblendet werden. Die Vor- und Nachteile von *TheCodePlayer* sind in der Tabelle 3.8 aufgelistet. Ein großer Nachteil der Plattform ist, dass der Benutzer bereits Kenntnisse im Bereich der Webentwicklung benötigt, um die Walkthroughs nachvollziehen zu können. Es werden keine Erklärungen und Hilfestellungen angeboten. Dadurch erreicht *TheCodePlayer* nur eine Gesamtbewertung von 58% und ist damit auf dem letzten Platz der getesteten Werkzeuge (siehe Tabelle 3.12).

F. Khan Academy

Khan Academy [27] ist eine im Jahr 2009 gegründete Non-Profit Plattform. Sie hat die Absicht, jedem Menschen das Erlernen von verschiedenen Fachgebieten von zu Hause aus zu ermöglichen. Dazu zählen unter anderem die Bereiche der Mathematik, der Naturwissenschaften, der Kunst, der Informatik und der Wirtschaft.

Um das Portal nutzen zu können ist keine Registrierung notwendig. Damit die Fortschritte jedoch gespeichert werden, kann man sich mit E-Mail, Facebook oder Google kostenlos anmelden. Zusätzlich kann bei der Registrierung ausgewählt werden, ob man Lehrkraft oder Elternteil ist. Dadurch kann man neben seinen eigenen Account auch seine SchülerInnen bzw. Kinder verwalten und deren Fortschritt überprüfen.

Tabelle 3.8: TheCodePlayer: Vor- und Nachteile

Vorteile	Nachteile
• Keine Registrierung notwendig • Geschwindigkeit kann in verschiedenen Stufen reguliert werden • Walkthrough kann pausiert werden • Auswahl der Ansichten für CSS, HTML und JS möglich • Änderungen im Code können durch die Vorschau gut nachvollzogen werden	• CSS, HTML und JS müssen vom Benutzer bereits beherrscht werden, um dem Walkthrough folgen zu können • Text kann während des Walkthroughs verändert werden, was falsche Ausgaben erzeugt • Schritte werden nicht erläutert • Es wird keine Hilfe angeboten

Das Fachgebiet „Computing“ teilt sich bei *Khan Academy* in drei Teile. Der erste Teil **Computer programming** bietet verschiedene Kurse zu den Sprachen HTML, CSS, JavaScript und SQL an. Im Bereich **Computer Science** gibt es zum Beispiel Informationen zu Algorithmen oder Kryptographie. Im dritten Teil, dem **Hour of Code** wird dem Benutzer eine Einführung in die Programmierung mithilfe von JavaScript angeboten.

Die einzelnen Kurse enthalten wiederum verschiedenen Module. Dazu gehören Einführungen, Quick Tips, Challenges, Quiz und Projects. Jeder Kurs startet mit einer Einführung in das entsprechende Thema. Diese werden in Textform oder als Video angeboten. *Quick Tips* erklären mit einfachen Animationen den Umgang mit der Plattform, zum Beispiel die Benutzung der Farbauswahl im Editor. Eine Challenge besteht aus mehreren Teilaufgaben und ist in verschiedene Bereiche eingeteilt (siehe Abbildung 3.6). Im ersten Bereich wird die Teilaufgabe dieser Challenge beschrieben. Zusätzlich wird an dieser Stelle ein Hinweis angezeigt, um dem Nutzer den Einstieg zu erleichtern. Darunter befindet sich ein Editor, in den der Benutzer seine Lösung eingeben kann. Die Eingaben werden unmittelbar ausgewertet und in einem Ausgabebereich dargestellt. Bei Fehlern reagiert das System sofort und gibt dem Benutzer sinnvolle Korrekturhinweise. Wurde eine Teilaufgabe richtig gelöst, kann zum nächsten Schritt übergegangen werden.

Im Modul Quiz werden dem Benutzer Fragen gestellt, die er durch Auswahl einer vorgegebenen Antwortmöglichkeit beantworten kann. Verschiedene Hinweise können nacheinander eingeblendet werden. Der letzte Hinweis gibt die Lösungsantwort an. Das Modul „Project“ ist wie eine Challenge aufgebaut. Der Nutzer ist in der Gestaltung seiner Lösung bei Projekten jedoch sehr frei. Er kann selbst entscheiden, wann er seinen Lösungsvorschlag einreichen möchte. Anschließend werden ihm verschiedene Fragen gestellt, ob seine Lösungen mit dem gewünschten Ziel des Projektes zusammenpassen.

Bei der Auflistung der Vor- und Nachteile von der *Khan Academy* (siehe Tabelle 3.6) wird deutlich, dass es nur wenige Punkte gibt, die negativ auffallen. Die Plattform bietet einen sehr gelungenen Einstieg in die Grundlagen der Programmierung (Variablen, Schleifen, If-Else-Bedigungen etc.). Aber auch fortgeschrittene Algorith-

Abbildung 3.6: Khan Academy - Ansicht einer Challenge

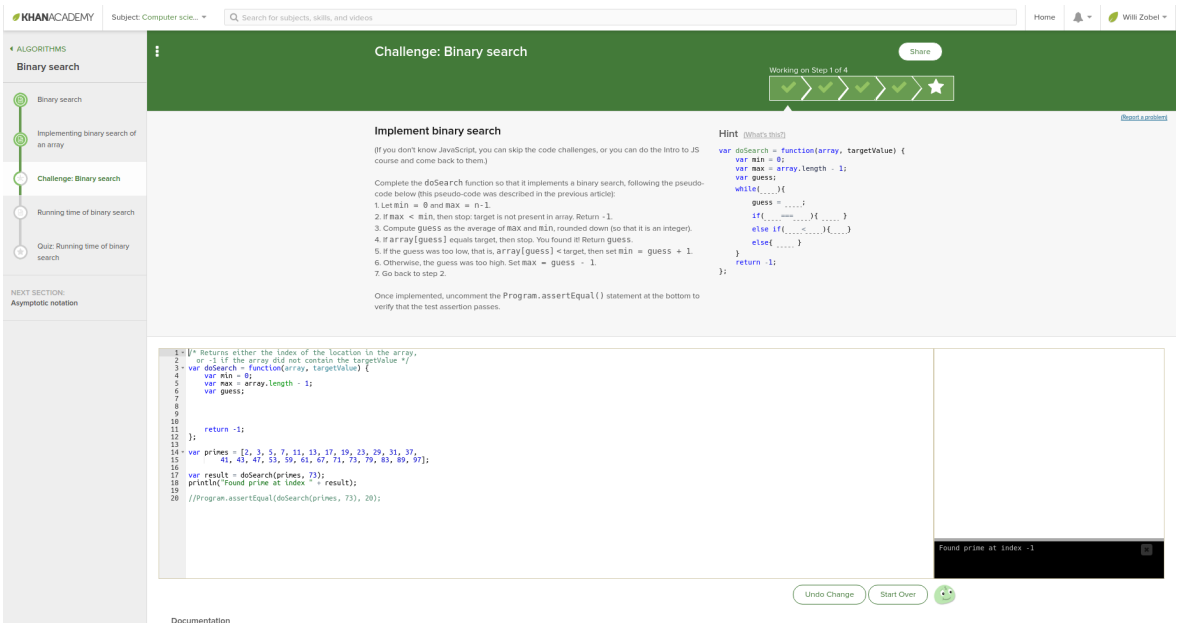
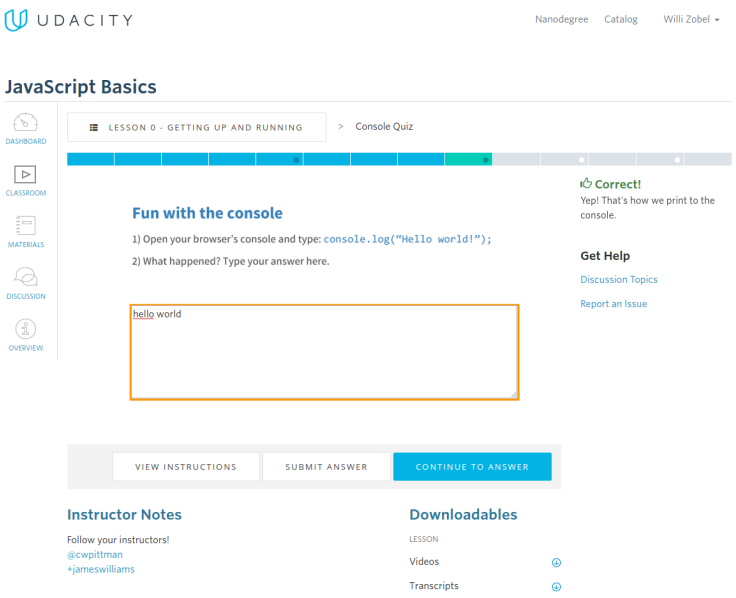


Tabelle 3.9: Khan Academy: Vor- und Nachteile

Vorteile	Nachteile
• Registrierung nicht notwendig • Sehr niedriger Anspruch durch gute Erklärungen und Videos • Viele verschiedene Themengebiete • Abwechslungsreiche Module • Sehr gute Hilfestellungen	• Durch großen Umfang etwas unübersichtlich • Kein Pseudo-Code

men, wie die binäre Suche oder Sortierungen, werden dem Benutzer durch verschiedene Module beigebracht. Die Hilfestellungen sind sehr vielseitig und reichen von einfachen Hinweisen bis zum Austausch mit anderen Mitgliedern. Damit erreicht die *Khan Academy* mit 94% die beste Bewertung der getesteten Werkzeuge (siehe Tabelle 3.12).

Abbildung 3.7: Udacity - Ansicht eines Kurses



G. Udacity

Das Anliegen von *Udacity* [38] ist eine Welt zu schaffen, in der Bildung effizient, bezahlbar und jedem zugänglich gemacht wird. Dafür bieten sie kostenpflichtige und kostenlose Kurse zum Thema Programmierung an. Die freien Kurse können ohne Registrierung betrachtet werden, nach einigen Schritten wird man jedoch zur Anmeldung aufgefordert. In den folgenden Abschnitten werden nur die freien Kurse von *Udacity* betrachtet, was die Gesamtbewertung des Portals einschränkt.

Ein Kurs ist in verschiedene Kapitel eingeteilt, die in einem Dashboard aufgelistet sind. Wenn ein Kapitel geöffnet wird, gelangt man in den sogenannten Classroom. Dieser besteht wiederum aus mehreren Teilschritten. In jedem Schritt befindet sich ein Video mit dem entsprechenden Tutorial. Am Ende einiger Schritte befindet sich eine Fragestellung, die der Benutzer beantworten kann (siehe Abbildung 3.7). Hilfestellungen gibt es dabei nicht. Die Video-Tutorials sind gut beschrieben. Die Quiz beinhalten jedoch immer nur eine Frage und sind damit nicht gut geeignet, um das gelernte Wissen abzufragen. In einem Diskussionsforum kann sich der Benutzer mit anderen Personen über die Aufgaben und Kurse austauschen. Im Abschnitt „Materials“ bekommt der Lernende zusätzliche Informationen zu den Kursinhalten. In Tabelle 3.10 sind alle Vor- und Nachteile von *Udacity* aufgelistet. Bei der Bewertung ergibt sich ein durchschnittliches Ergebnis von 74% (siehe Tabelle 3.12). Positiv hervorzuheben ist, dass die Kurse in verschiedene Schwierigkeitsstufen eingeteilt werden. Jedoch wird der Benutzer bei den Tutorials nur wenig durch komplexere Aufgaben gefordert.

Tabelle 3.10: Udacity: Vor- und Nachteile

Vorteile	Nachteile
• Freie Kurse können ohne Registrierung betrachtet werden • Kurse sind mit unterschiedlichen Ansprüchen gekennzeichnet • Diskussionsforum vorhanden • Videos sehr verständlich gestaltet	• Nur spezielle Programmiersprachen möglichen • Lösungen können nicht zurückgesetzt werden • Keine Korrekturhinweise • Aufgaben und Erklärungen hauptsächlich in Video-Form

H. Learneroo

„You don't learn skills like programming by just listening to a lecture, you learn programming by programming.“ [35]. So lautet ein Prinzip von *Learneroo* und möchte die Benutzer damit animieren, selbst zu forschen und zu entdecken. Dafür werden viele interaktive Kurse zu Themen wie Java, Webentwicklung, Algorithmen, Python und auch der Mathematik angeboten. Es gibt sowohl kostenlose, als auch kostenpflichtige Tutorien. Die Bewertung bezieht sich in dieser Arbeit jedoch nur auf die freien Kurse.

Die Kurse können auch ohne Registrierung betrachtet werden, jedoch kann der Benutzer die Challenges in diesem Fall nicht bearbeiten. Jeder Kurs besteht aus mehreren Schritten, in denen jeweils bestimmte Sprachelemente beschrieben werden. Ein Schritt kann wiederum Challenges enthalten. Davon gibt es zwei verschiedene Typen. Beim ersten Typ wird dem Benutzer eine Frage gestellt, die er beantworten muss. Beim zweiten Typ muss der Benutzer selbst eine Funktion implementieren, dessen Rückgabewert anschließend überprüft wird (siehe Abbildung 3.8). Dafür wird ein integrierter Editor zur Verfügung gestellt. Dieser ist sehr anpassungsfähig. Sogar eine Kollaboration mit anderen Personen ist möglich. In einigen Kursen des Moduls „Algorithmen“ kann dort auch die Programmiersprache ausgewählt werden. Neben dem Editor wird die Ausgabe mit verschieden Eingabewerten getestet und angezeigt. Der Benutzer erhält durch eine „Quick Reference“ und kleinen Hinweisen zusätzliche Hilfe bei der Bearbeitung. Ein Nachteil ist jedoch, dass bei falschen Eingaben nur ungenügende Korrekturhinweise angezeigt werden. Alle Vor- und Nachteile von *Learneroo* werden in der Tabelle 3.11 aufgezeigt.

In der Bewertung erhält die Plattform *Learneroo* 83% (siehe Tabelle 3.12). Dieses Ergebnis wird vor allem durch die umfangreichen Hilfestellungen bei der Bearbeitung beeinflusst. Zusätzlich ist die Anpassungsfähigkeit der Plattform deutlich höher als bei den anderen getesteten Werkzeugen. Der Benutzer kann bei der Bearbeitung im Editor zum Beispiel ein ein- oder zweispaltiges Layout wählen, was die Verwendung deutlich erleichtert.

Abbildung 3.8: Learneroo - Ansicht einer Challenge in einem Kurs

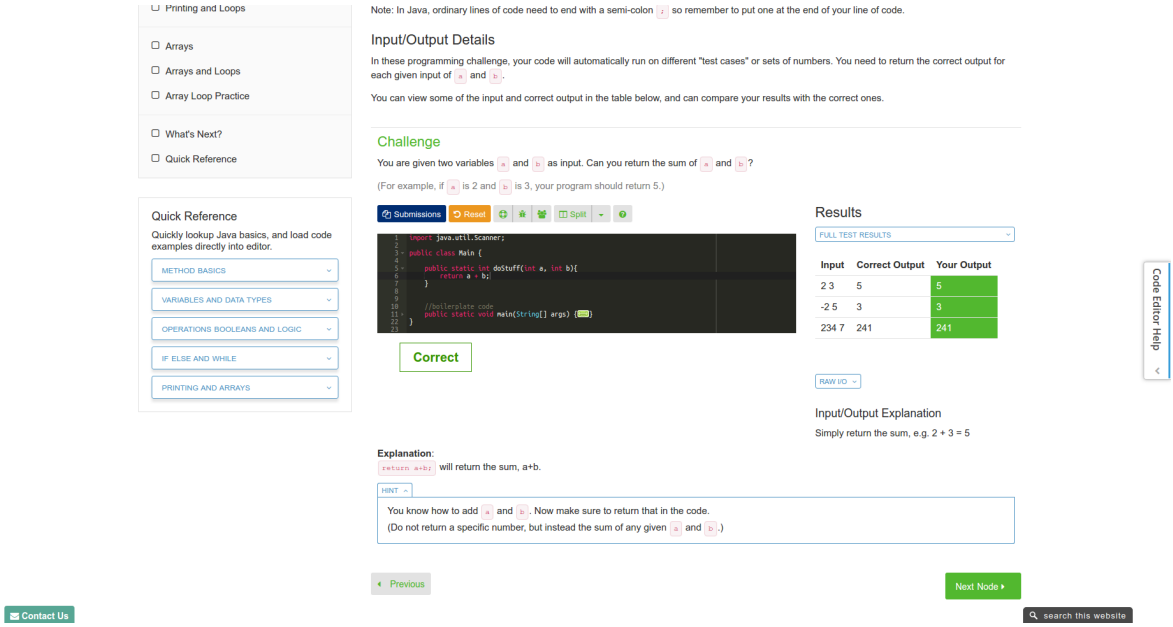


Tabelle 3.11: Learneroo: Vor- und Nachteile

Vorteile	Nachteile
• Programmiersprache kann teilweise selbst ausgewählt werden • Hinweise können eingeblendet werden • Persönliche Fähigkeit pro Kurs wird angezeigt • Quick Reference für weitere Informationen • Oberfläche kann angepasst werden	• Registrierung für Challenges notwendig • Unterschiedliche Voraussetzungen für Kurse nicht gekennzeichnet • Ungenügende Korrekturhinweise • Einzelne Schritte beim Ausführen können nicht gesteuert werden

3.1.4 Fazit

Die Bewertung der Werkzeuge hat viele Erkenntnisse erzeugt, die für die Entwicklung der Webanwendung in dieser Arbeit maßgeblich sind. Auf einigen der getesteten Plattformen wird das Verstehen von Quellcode vorausgesetzt. Eine frühe Gewöhnung daran ist zwar sinnvoll, jedoch sollten die einzelnen Codezeilen gut erläutert werden. Um die Motivation der SchülerInnen zu steigern, ist es zudem empfehlenswert, die gelösten Aufgaben zu kennzeichnen. Somit kann der Fortschritt auch von der zuständigen Lehrkraft überprüft werden.

Tabelle 3.12: Bewertungen aktueller Werkzeuge

	TheCodePlayer	**Khan Academy**	**Udacity**	**Learneroo**
Aufgabenspezifische Kriterien	53%	93%	67%	73%
Aufgabenangemessenheit	65%	100%	100%	90%
Selbstbeschreibungsfähigkeit	64%	88%	84%	96%
Lernförderlichkeit	90%	100%	80%	100%
Steuerbarkeit	93%	100%	80%	80%
Erwartungskonformität	60%	90%	100%	100%
Individualisierbarkeit	60%	90%	50%	100%
Fehlertoleranz	0%	100%	73%	80%
Gesamt	**58%**	**94%**	**74%**	**83%**
	Codecademy	**Code**	**Learn X**	**Coding Dojo**
Aufgabenspezifische Kriterien	60%	80%	73%	53%
Aufgabenangemessenheit	95%	95%	75%	95%
Selbstbeschreibungsfähigkeit	88%	92%	68%	88%
Lernförderlichkeit	90%	100%	80%	90%
Steuerbarkeit	67%	67%	80%	80%
Erwartungskonformität	100%	80%	80%	100%
Individualisierbarkeit	50%	70%	40%	50%
Fehlertoleranz	73%	80%	7%	73%
Gesamt	**70%**	**82%**	**67%**	**68%**

Durch die Plattform **Code** [18] ist die Idee entstanden, die blockbasierte Programmierung mit in dem zu erstellenden Werkzeug zu integrieren, da diese einen einfachen Einstieg ermöglicht. Jedoch sollte dabei darauf geachtet werden, dass die einzelnen Blöcke schrittweise ausgeführt werden können.

Die Plattform **Khan Academy** [27] erhält mit 94% die beste Bewertung. Vor allem die abwechslungsreichen Module zum vermitteln der Lerninhalte hinterlassen einen positiven Eindruck. Trotzdem ist dieses Werkzeug im Unterricht nur eingeschränkt zu verwenden, da der große Umfang die Plattform unübersichtlich erscheinen lässt. Dieses Problem haben auch viele andere der getesteten Werkzeuge.

Bei **Learneroo** [35] ist hervorzuheben, dass eine Anpassung der Oberfläche deutliche Vorteile bei der Bearbeitung von Aufgaben bringt. Auch wenn kein Pseudo-Code verwendet wird, um die Lerninhalte zu vermitteln, ist die Auswahl der Programmiersprache eine gute Alternative.

Neben den positiven Beispielen haben jedoch auch Plattformen wie **TheCodePlayer** [37] oder **Udacity** [38] gezeigt, dass das Vermitteln von Lerninhalten allein über Videos nicht sinnvoll ist. Vor allem nicht im Unterricht, da diese meist die Aufgabe der LehrerInnen ersetzen. Generell sollte bei einem Werkzeug im Informatikunterricht möglichst auf Tonausgaben verzichtet werden, um andere SchülerInnen nicht zu stören.

In Anhang A werden die genauen Bewertungen der einzelnen Werkzeuge dargestellt.

3.2 Werkzeugvorlage

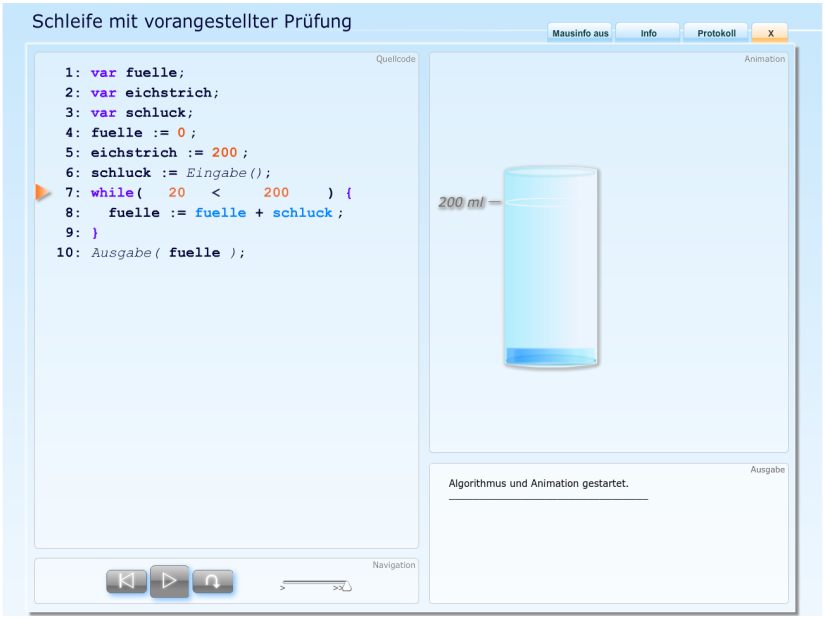
Neben den acht aktuellen Werkzeugen, wurde auch das Werkzeug getestet, dass dieser Arbeit als Vorlage dient und aus der Diplomarbeit von Claudia Schindler [11] entstand. Um die Bewertung mit den aktuellen Lernplattformen vergleichen zu können, wurden hierbei die gleichen Bewertungsmaßstäbe gewählt. In der Tabelle 3.13 sind die Bewertungen der einzelnen Kriterien aufgelistet. An jedes Kriterium wurden dabei 0 bis 5 Punkte gegeben, wobei 0 die schlechteste und 5 die beste Bewertung ist. Innerhalb jeder Kategorie wurden anschließend die prozentualen Anteile an der maximalen Punktezahl berechnet. Danach wurden die Bewertungen der Kategorien aus der ISO-Norm 9241-110 (siehe Abschnitt 3.1.1) gemittelt. Zum Schluss wurde der Mittelwert aus den Bewertungen der ISO-Norm (82%) und der „Aufgabenspezifischen Bewertungskriterien“ (100%) gebildet, der die abschließende Gesamtbewertung darstellt (91%).

Im Rahmen einer wissenschaftlichen Arbeit wurde die Werkzeugvorlage um einige Funktionen erweitert [3]. Die folgenden Bewertungen beziehen sich auf die überarbeitete Version.

3.2.1 Aufgabenspezifische Bewertungskriterien

Um das Werkzeug nutzen zu können, ist keine Registrierung notwendig. Dadurch kann der Nutzer gleich in die angebotenen Kurse einsteigen. Der angegebene Quellcode kann schrittweise ausgeführt werden. Durch die

Abbildung 3.9: Werkzeugvorlage - Ansicht eines Moduls



gleichzeitige Visualisierung kann der Benutzer den Code gut nachvollziehen. Die Kurse enthalten einfache, praktische Beispiele und können so gut von Anfängern verwendet werden. Zusätzliche Hinweise werden als Tooltip über den entsprechenden Elemente angezeigt. Der Quellcode ist in einer Pseudo-Sprache verfasst, so dass dem Benutzer keine spezielle Programmiersprache beigebracht wird (siehe Abbildung 3.9). Dadurch erhält der Nutzer eine allgemeine Einführung in die Grundlagen der Programmierung.

Die aufgabenspezifischen Kriterien konnten somit voll erfüllt werden, weshalb das Werkzeug in dieser Kategorie eine Bewertung von 100% erhält.

3.2.2 Aufgabenangemessenheit

Grundlegend unterstützt das Werkzeug den Benutzer dabei, die Grundlagen von einfachen Algorithmen zu erlernen. Dem Benutzer werden nur die Informationen bereitgestellt, die für die Aufgabe relevant sind. Es gibt jedoch kleine Abstriche bei der Informationsein- und -ausgabe. Die Benutzereingaben, mit der einzelne Variablen im Programmcode geändert werden können, werden nur im Protokoll und in der Visualisierung vermerkt. In einigen Modulen (wie zum Beispiel bei den Schleifen) werden die Eingaben auch nicht in der Visualisierung vermerkt (siehe Abbildung 3.9). Das Belegungsprotokoll ist zudem für Anfänger etwas unverständlich aufgebaut und bedarf einer Eingewöhnung.

Für die Bedienung des Werkzeugs sind keine Shortcuts vorhanden. Diese könnten zum Beispiel dazu dienen, die einzelnen Schritte und die Geschwindigkeit des Ablaufs zu steuern. Zusätzlich wäre es für den Benutzer

praktisch, wenn persönliche Einstellungen gespeichert und bei der nächsten Verwendung automatisch geladen werden würden. Dies könnte zum Beispiel bei der Ansicht des Belegungsprotokolls oder der Geschwindigkeitseinstellung erfolgen, wenn der Benutzer eine bestimmte Ansicht bzw. Einstellung präferiert.

Aufgrund der kleinen Mängel erhält das Werkzeug in der Kategorie „Aufgabenangemessenheit“ eine Bewertung von 80%.

3.2.3 Selbstbeschreibungsfähigkeit

Positiv hervorzuheben ist, dass das Werkzeug mit einem Tooltip dem Benutzer jede Codezeile erläutert. Durch die zusätzliche Visualisierung ist es einfach die einzelnen Schritte nachzuvollziehen. Die Schaltflächen und Hinweise sind eindeutig beschrieben. Wie in Abbildung 3.9 zu sehen, sind die Symbole für „Wiedergabe“, „Pause“, „Nächster Schritt“, „Reset“ und die Geschwindigkeit nachvollziehbar. Lediglich die Pseudo-Sprache bedarf weniger Überarbeitungen, um sie dem Benutzer noch einfacher zu präsentieren. In den Switch-Konstrukten können die „break;“-Anweisungen vernachlässigt werden. Die Schleifen können vereinfacht und bekannten Programmiersprachen nachempfunden werden, um den Umstieg nicht unnötig zu behindern.

Der Benutzer wird jederzeit darüber informiert, in welchem Modul er sich gerade befindet. Der Aufruf der Modul-Übersicht ist von jedem Punkt aus möglich. In der Übersicht werden die verschiedenen Module gruppiert und übersichtlich dargestellt. Auch innerhalb des Moduls wird die aktuelle Position durch einen auffälligen Pfeil gekennzeichnet. Einen negativen Punkt gibt es bei der Rückmeldung des Bearbeitungsstands der Module zu vermerken. Absolvierte Algorithmen könnten hervorgehoben werden, um den Benutzer darüber zu informieren, welche Module er noch nicht betrachtet hat.

Trotz der dargestellten Abzüge, erhält das Werkzeug in der Kategorie „Selbstbeschreibungsfähigkeit“ eine Bewertung von 96%, da die Gewichtung der bestehenden Probleme sehr gering ausfällt.

3.2.4 Lernförderlichkeit

In dieser Kategorie stellt sich die Frage, ob die Anwendung den Benutzer dabei unterstützt, das Werkzeug und die dargestellten Informationen zu erlernen. Die Konsistenz ist dabei ein entscheidender Punkt. Dieser wird sehr gut umgesetzt, da alle Module gleich aufgebaut sind und der Benutzer sich somit nicht an verschiedene Oberflächendarstellungen gewöhnen muss. Lediglich die Navigation ist etwas umständlich gelöst. Beim öffnen eines Moduls aus der Übersicht heraus, öffnet sich eine Art neues Fenster. Zurück zur Übersicht gelangt man, in dem das aktuelle Modulfenster geschlossen wird. Zwischen den Ansichten kann nicht direkt gewechselt werden. Diese Darstellungsweise kann zu Irritationen führen, da die Schaltfläche für das Schließen eines Moduls assoziiert, dass die gesamte Anwendung beendet wird. Auch der Wechsel zwischen Animation und Belegungsprotokoll ist nicht eindeutig gelöst. Dies kann den Benutzer bei der Verwendung zu Beginn bei der Lernförderlichkeit behindern.

Aufgrund der dargelegten Navigationsprobleme erhält das Werkzeug in der Kategorie „Lernförderlichkeit“ eine Bewertung von 80%.

3.2.5 Steuerbarkeit

Die Steuerbarkeit der Anwendung ist sehr gut. Die Codezeilen können automatisch nacheinander abgespielt werden. Dafür kann der Benutzer die Geschwindigkeit der Ausführung in neun verschiedenen Stufen steuern. Auch das Pausieren ist möglich, um sich den aktuellen Zustand genauer anzuschauen. Die Einzelschritte können auch von Hand gesteuert werden. Die gesamte Ausführung kann zwar zurückgesetzt werden, einzelne Schritte kann der Benutzer jedoch nicht zurückgehen. Aufgrund dieser fehlenden Funktionalität erhält das Werkzeug in der Kategorie „Steuerung“ eine Bewertung von 93%.

3.2.6 Erwartungskonformität

Wie bereits in der Kategorie „Lernförderlichkeit“ beschrieben, ist die Navigationsstruktur der Werkzeugvorlage nicht perfekt umgesetzt. Dem Benutzer wird nicht verdeutlicht, in welchem Fall sich ein neues Fenster öffnet und wann sich der aktuelle Inhalt ändert, wenn auf eine Schaltfläche gedrückt wird. Dies entspricht nicht den Erwartungen des Nutzers. Eingabefenster für Variablenwerte können zudem nur geschlossen werden, indem ein gültiger Wert eingegeben wird. Die Darstellung der Bedienelemente wurde jedoch einheitlich gestaltet und erzeugt damit einen Wiedererkennungswert.

In den Modulen zum Thema Schleifen wird bei der Visualisierung ein Gefäß schrittweise mit Wasser gefüllt. Wenn der Rand des Gefäßes erreicht wird, bleibt der Wasserstand bestehen und läuft nicht drüber hinaus. Dies führt zu falschen Eindrücken vom Ergebnis. Damit erreicht die Werkzeugvorlage eine Bewertung von 80% in der Kategorie „Erwartungskonformität“.

3.2.7 Individualisierbarkeit

Bei der Individualisierbarkeit geht es darum, dem Benutzer zu ermöglichen, das System nach seinen Bedürfnissen anpassen zu können. Dies gelingt der Anwendung nur begrenzt. Zwar können die Mausinfos (zusätzliche Informationen in Tooltip bei Mouseover) ein- und ausgeschaltet werden, die Größen und das Layout der Ansichten können jedoch nicht angepasst werden. Auf kleinen Bildschirmen werden daher die Bereiche und Schriften viel zu klein dargestellt. Aufgrund dieser geringen Individualisierbarkeit erhält die Anwendung in dieser Kategorie nur 70%.

3.2.8 Fehlertoleranz

Die einzigen Eingaben die der Benutzer beeinflussen kann, sind Variablenwerte. Teilweise werden Zahlenbereiche genannt, in denen die Eingabe liegen muss. In einigen Fällen, wie zum Beispiel bei den Schleifen, werden jedoch keine Bereiche vorgegeben, so dass der Nutzer selbst testen muss, ob seine Eingabe korrekt ist. Auch in der Fehlermeldung sind meisten keine eindeutigen Korrekturhinweise gegeben. Es wird lediglich angezeigt, dass die Eingabe nicht korrekt war. Hilfestellungen zu den Codeblöcken erhält der Benutzer nur über die bereits erwähnten Tooltips. Es ist kein weiterer Hilfebereich definiert.

Die unausgereiften Korrekturhinweise begründen die Bewertung von 80% in der Kategorie „Fehlertolerant“.

3.2.9 Evaluationsergebnisse

Im Rahmen der Diplomarbeit von Claudia Schindler [11] wurde eine Evaluation durchgeführt. Dafür wurde das Werkzeug und ein Fragebogen an LehrerInnen, Studierende des Lehramts Informatik und Studierende der Informatik gesendet. Die Bewertung der Anwendung erfolgte mithilfe von Schulnoten und ergab am Ende eine Gesamtnote von 1,82 (Gut). In diesem Abschnitt soll es darum gehen, welche Probleme und Hinweise bei der Evaluation auftraten. Diese können anschließend für die Gestaltung des Werkzeugs genutzt werden, das in dieser Arbeit entsteht. Die Evaluationsergebnisse beziehen sich auf die ursprüngliche Version des Werkzeugs, nicht auf die erweiterte Variante von Sebastian Lehmann [3].

Das **Aussehen** des Werkzeugs erhielt mit einem Durchschnitt von 1,71 eine gute Bewertung. Kommentare konnten zu diesem Bereich nicht abgegeben werden, wodurch sich keine genauen Begründungen für die Bewertungen finden lassen. Die Wechselansicht von Animation und Belegungsprotokoll wurde von einer Testperson mit *Mangelhaft* bewertet, was sich auf die in der Kategorie „Lernförderlichkeit“ und „Erwartungskonformität“ angesprochene Navigationsstruktur beziehen könnte.

Im Bereich **Inhalt** der Evaluation geht es um die Bewertung der einzelnen Module und den dazugehörigen Metaphern bei der Visualisierung. Dabei konnten zusätzlich Kommentare, Begründungen und Verbesserungsvorschläge angegeben werden. Grundsätzlich wurde auf die geringe Schriftgröße innerhalb der Module hingewiesen. Die Boxen-Metapher in den Modulen der Wertzuweisungen wurde als verständlich und einfach bewertet. Für komplexere Szenarien wurde sich jedoch eine Darstellung mit Speicherbelegung und unterschiedlichen Wertetypen gewünscht. Die Anzeige eines Speichers könnte mithilfe einer Regal-Metapher umgesetzt werden, in der die einzelnen Fächer mit Adressen beschriftet sind. Die Typen der Werte könnten durch verschiedene Formen der Boxen dargestellt werden. Ein wichtiger Diskussionspunkt in den Modulen für Verzweigungen ist die Zuweisung *leer* für die Variable „empfinden“. Einige der Testpersonen fanden diese Zuweisung nicht gelungen. Jedoch wurde auch die Deklaration von leeren Strings für Anfänger als schwierig erachtet. Alternativen für die Zuweisung konnten nicht genannt werden. Die Anpassung der Metapher wäre daher an dieser Stelle sinnvoll. Diese sollten sich mehr an dem Alltag der SchülerInnen orientieren, wie zum Beispiel der Akkustand

vom Smartphone. Weiterhin wurde darauf hingewiesen, den Pseudocode mehr an die Programmiersprache *Pascal* anzugleichen, da diese im Schulalltag häufiger zur Anwendung kommt als zum Beispiel C. Vor allem die Verwendung der Switch-Anweisung wurde kritisiert. An dieser Stelle wäre es sinnvoll, dem Benutzer die Auswahl von verschiedenen Programmiersprachen zu ermöglichen. Die Wasserglas-Metapher in den Modulen für Schleifen wurde am meisten gelobt. Jedoch gibt es dabei Probleme mit der Umsetzung. Wie bereits im Abschnitt „Erwartungskonformität“ beschrieben, wird auch bei der Evaluation kritisiert, dass das Wasserglas in der Visualisierung nicht überlauft. Zusätzlich wurde darauf hingewiesen, dass die Variable *Schluck* in der Visualisierung dargestellt werden sollte. Am Ende dieses Evaluationsbereiches wurde nach Erweiterungen um Module gefragt. Hierbei wurden konkrete Programmbeispiele (z.B. der größte gemeinsame Teiler) und weitere Programmiertechniken (z.B. Zeiger) genannt. Auch wurden sich Einführungen in die Programmiersprache und eine Hilfeseite gewünscht.

Die schlechteste Bewertung gab es im Bereich **Steuerung**. Mit einer durchschnittlichen Note von 2,65 bewerteten die Testperson die Navigation innerhalb eines Moduls. Die Autorin [11] vermutet die ungewohnten Schaltflächen für *Abspielen*, *Pause*, *Reset* und *Nächster Schritt*. Aufgrund fehlender Kommentare kann diese Bedeutung jedoch nicht klar begründet werden. Diese Navigationsstruktur wurde im Rahmen der wissenschaftlichen Arbeit von Sebastian Lehmann [3] überarbeitet.

Im Bereich **Wirkung** erhielt die Fragestellung nach der Kombination von Quellcode und visueller Darstellung ein *Sehr gut* (1,38). Diese Darstellungsform sollte daher beibehalten werden. Für das Verständnis des Programmablaufs, wurde die Anzeige des Belegungsprotokolls minimal besser bewertet (1,81), als die Darstellung der Animation (1,88).

Im letzten Bereich **Allgemeines** wurde danach gefragt, was die Testpersonen an dem Werkzeug verbessern würden. Dabei wurde zum Beispiel die gleichzeitige Anzeige von Animation und Belegungsprotokoll genannt. Dadurch könnte der Benutzer die aktuellen Werte der Variablen einfach nachvollziehen, ohne die Visualisierung verlassen zu müssen. Zusätzlich wurde mehrfach angemerkt, dass die Modulbeschreibungen verständlicher umschrieben werden könnten, um den SchülerInnen einfach und schnell zu erklären, worum es im Modul geht.

3.2.10 Erweiterungen

Im Jahre 2009 wurde die Werkzeugvorlage von Claudia Schindler [11] im Rahmen einer wissenschaftlichen Arbeit von Sebastian Lehmann erweitert [3]. Dabei wurden vier weitere Module zum Thema „Unterprogramme“ hinzugefügt:

1. Funktion mit Werteparameter (call by value)
2. Prozedur mit Werteparameter (call by value)
3. Funktion mit Variablenparameter (call by reference)
4. Prozedur mit Variablenparameter (call by reference)

Neben den zusätzlichen Modulen wurden weitere Änderungen am Werkzeug vorgenommen. Die wichtigsten werden in der folgenden Liste aufgezählt:

- Module wurden in Kategorien eingeordnet
- Geschwindigkeit kann geregelt werden
- Symbole für Steuerung der Module wurde ausgetauscht
- Tooltips wurden ausgebaut
- Wasserglasanimation wurde durch neues Design ersetzt

Bei der Evaluation der Erweiterungen wurden weitere Probleme und Verbesserungsvorschläge aufgezählt. Eine Testperson fand den Quelltext für den Einsatz des Werkzeugs auf Mittelschulen zu kompliziert. Zusätzlich wurde sich hier, wie bereits bei der Evaluation in der Diplomarbeit von Claudia Schindler [11], ein animiertes Struktogramm zur besseren Übersicht gewünscht.

3.2.11 Fazit

Die Umsetzung mit Adobe Flash ist nicht mehr zeitgemäß. Auf mobilen Plattformen, die zunehmend auch im Unterricht eine immer größere Rolle spielen [8], werden Flash-Inhalte oft nicht korrekt wiedergegeben. Unter dem Betriebssystem iOS ist die Darstellung gar nicht erst möglich. Auch wird für die Verwendung von Flash-Inhalten ein zusätzliches Browser-Plugin benötigt. Da bei Adobe Flash häufig neue Sicherheitslücken erscheinen, ist es wichtig die entsprechenden Plugins immer aktuell zu halten. Aufgrund dessen ist die Entscheidung gefallen, das Werkzeug als Webanwendung zu erstellen.

Um aus den Problemen und Erfahrungen zu lernen, wurde in diesem Kapitel die Werkzeugvorlage analysiert. Zum einen hat sich herausgestellt, dass die Beispiele und vor allem der Quelltext zu Beginn für viele unverständlich ist. Es gilt daher den Code besser zu erläutern und eventuell ein Struktogramm für ein besseres Verständnis zu implementieren. Außerdem sollte das Belegungsprotokoll nicht durch die Animation ersetzt, sondern gleichzeitig präsentiert werden. Auch die Metaphern der einzelnen Module und die dazugehörigen Animationen müssen überdacht werden. Die gesamten Ergebnisse der Bewertung werden in Anhang B dargestellt.

Tabelle 3.13: Bewertung der Werkzeugvorlage [11]

Kategorie	Kriterium	Bewertung	
Aufgabenspezifische Bewertungskriterien	Registrierung	5	100%
	Anspruch	5	
	Anwendungsgebiet	5	
Aufgabenangemessenheit	Fernhaltung von internen Aufgaben	5	80%
	Aufgabenorientierte Informationsein- und ausgabe	3	
	Verwendung von Makros, Voreinstellungen und Shortcuts	4	
	Speichern von ausgefüllten Daten	4	
Selbstbeschreibungsfähigkeit	Erklärung von Schritten	5	96%
	Eindeutige Bezeichnungen	5	
	Adaptive Hilfe	5	
	Rückmeldung zum Bearbeitungsstand	4	
	Überblickinformationen	5	
Lernförderlichkeit	Unterstützung des Nutzers	3	80%
	Konsistenz	5	
Steuerbarkeit	Steuerung der Richtung durch Nutzer	4	93%
	Steuerung der Geschwindigkeit durch Nutzer	5	
	Schritte können rückgängig gemacht werden	5	
Erwartungskonformität	Einheitliche Dialogverhalten	5	80%
	Einheitliche Darstellung gleicher Bedienelemente	3	
Individualisierbarkeit	Anpassung vom Umfang der Erläuterungen	4	70%
	Anpassung der Darstellung	3	
Fehlertoleranz	Abfangen von Fehlern	5	80%
	Korrekturhinweise	4	
	Hilfe	3	
Gesamtbewertung			**91%**

4 Implementierung

Nachdem in den letzten Kapiteln eine Zielgruppen- und Konkurrenzanalyse durchgeführt wurde, erfolgt nun die Konzeption der eigenen Webanwendung. Dafür werden als erstes die Anforderungen an die Anwendung erstellt und nach der Priorität geordnet. Anschließend erfolgt die Erstellung von Szenarien. Diese stellen den Ablauf und die Inhalte der verschiedenen Module dar. Daraus entstehen der Aufbau der Anwendung und die entsprechenden Mockups. Abschließend wird die Webanwendung anhand der Entwurfsvorlagen implementiert.

4.1 Entwurf

4.1.1 Anforderungen

Die Anforderungen an die Webanwendungen werden in drei Gruppen aufgeteilt. Die Muss-Anforderungen haben die höchste Priorität und sind essentiell für die Anwendung. Die Soll-Anforderungen sollten umgesetzt werden. Wenn die Zeit zu knapp wird, können diese jedoch auch weggelassen werden. Die Kann-Anforderungen haben die geringste Priorität und können implementiert werden, wenn noch ausreichend Zeit vorhanden ist.

Muss-Anforderungen

- Module implementieren
    - Variablen
        - Wertzuweisungen
        - Operatoren

    - Verzweigungen
        - Einseitige Verzweigungen
        - Zweiseitige Verzweigungen
        - Mehrseitige Verzweigungen
    - Schleifen
        - Kopfgesteuerte Schleifen (While-Do)
        - Fußgesteuerte Schleifen (Do-While)
        - Zählschleifen (For)
    - Unterprogramme
        - Funktionen
        - Prozeduren mit Werteparametern (Call by Value)
        - Prozeduren mit Referenzparametern (Call by Reference)
    - Datenstrukturen
        - Objekte
        - Verkettete Listen (Einfügen)
- Modulinhalte erstellen
    - Beschreibung / Einleitung
    - Block-Based Programming Abschnitt
    - Steuerung
        - Start / Wiedergabe
        - Nächster Schritt
    - Werte der Variablen (Tabelle)
    - Visualisierung

Soll-Anforderungen

- Module implementieren
    - Datenstrukturen
        - Verkettete Listen (Löschen)
        - Verkettete Listen (Suchen)
- Modulinhalte erstellen
    - Block-Based Programming Abschnitt

        - Hilfestellung bei Hover
        - Einzelne Werte können geändert werden
    - Steuerung
        - Reset
        - Geschwindigkeit festlegen

Kann-Anforderungen

- Module implementieren
    - Variablen
        - Variablentypen
    - Datenstrukturen
        - Verkettete Listen (Sortieren)
        - Arrays
- Modulinhalte erstellen
    - Block-Based Programming Abschnitt
        - Einzelne Schritte können aktiviert werden
        - Animation beim Abspielen (Progress Bar)
        - Bei bestimmten Schritten nach Werten fragen
    - Steuerung
        - Schritt zurück
    - Visualisierung
        - Alternative: Flussdiagramm
    - Auswahl der Programmiersprache (z.B. Umgangssprachlich, Pascal, JavaScript, Java, ...)
- Eingaben speichern
    - Abgeschlossene Module in Menü anzeigen
    - Daten löschen
    - Beim Aufruf von Modulen alte Daten laden
- Tutorial zur Einführung in die Anwendung
- Layout & Design
    - Alternatives Farbdesign (helle/dunkle Variante)
    - Komponenten in Modulen können vergrößert/verkleinert werden
    - Ansicht der Komponenten können beliebig ausgewählt und ausgetauscht werden

4.1.2 Szenarien

Die Webanwendung soll verschiedene Module mit dem gleichen Grundaufbau enthalten. Jedes Modul besteht aus einer Beschreibung, die dem Nutzer eine Einführung in das entsprechende Themengebiet geben soll. Außerdem wird ein Code-Beispiel dargestellt, dass der Nutzer sortieren muss und anschließend ausführen kann. Bei der Ausführung des Codes findet eine gleichzeitige Visualisierung statt, wodurch der Benutzer die einzelnen Schritte im Code gut nachvollziehen kann. Der genaue Ablauf wird im folgenden Abschnitt erläutert. Anschließend werden die Inhalte der einzelnen Module beschrieben.

Genereller Modulablauf

1. Der Nutzer liest sich die Einleitung über das Thema durch.
2. Der Nutzer erhält eine Aufgabenstellung und sortiert anschließend die Codeblöcke so, dass sie der Aufgabenstellung entsprechen und valide sind.
3. Die Sortierung wird überprüft.
   (a) Wenn die Sortierung richtig ist, kann der Nutzer den Code ausführen, pausieren und in Einzelschritten durchgehen. Die Animation zeigt parallel eine Visualisierung des Codes an.
   (b) Ist die Sortierung falsch, wird ein Hinweis angezeigt und der Nutzer muss die Sortierung korrigieren. Springe zu Schritt 3.
4. Nach einem Reset können die Variablenwerte angepasst, die Codeblöcke umsortiert und das Programm erneut gestartet werden.

1. Modul: Wertzuweisungen

In diesem Modul geht es darum, den SchülerInnen eine Einführung in die Funktionsweise von Variablen zu geben. Dafür werden vorerst die Grundlagen für die Pseudo-Programmiersprache geklärt. Dazu gehören das Semikolon, um das Ende eines Kommandos zu kennzeichnen und das benötigte Schlüsselwort **var**, um Variablen zu definieren. Auf die Typisierung wird aufgrund der Komplexität verzichtet. Anschließend wird auf die Wertezuweisung eingegangen. Die Werte werden dabei jedoch vorerst auf ganze Zahlen und Zeichenketten begrenzt.

Aufgabe

Die Aufgabe besteht darin, verschiedene Variablen zu definieren und ihnen anschließend Werte zuzuordnen. Dabei müssen die SchülerInnen darauf achten, dass den Variablen erst Werte zugeordnet werden können, wenn sie mit dem Schlüsselwort **var** definiert wurden.

Visualisierung

Die Visualisierung orientiert sich größtenteils an der Variante aus der Werkzeugvorlage [11]. Dabei wurde jede Variable durch einen farbigen Kasten repräsentiert. Die Namen der Variablen entsprachen der jeweiligen Farbe und innerhalb der Box wurden die Werte dargestellt. Bei der neuen Darstellungsform sollen die Variablen von Regalfächern repräsentiert werden. Diese ermöglichen eine bessere Verknüpfung, um die Funktionsweise von Variablen verstehen zu können. Jedes Regalfach enthält den Namen und den Wert einer Variable. Da das Farbkonzept bei der Evaluation gut angenommen wurde [11], soll dies auch hier übernommen werden. Die Variablen werden somit nach Farben benannt und die dazugehörigen Regalfächer dementsprechend eingefärbt. Auf die Anzeige von Adressen wird verzichtet, da die Visualisierung möglichst einfach gehalten werden soll.

2. Modul: Operatoren

In den meisten Programmiersprachen gibt es viele Operatoren, die den SchülerInnen an dieser Stelle jedoch schnell überfordern können. Aus diesem Grund werden in dem Modul nur die wichtigsten Operatoren vorgestellt. Dazu gehören vorrangig die Rechenoperationen bzw. arithmetischen Operatoren (+, -, /, *), die anhand von Beispielen erläutert werden. Zusätzlich wird der Plus-Operator zur Konkatenation von Zeichenketten beschrieben. Bei der Darstellung von Beispielcode ist es wichtig, dass die Ergebnisse von den Operationen angezeigt werden, damit der Benutzer diese gut nachvollziehen kann.

Aufgabe

Wie auch im ersten Modul, müssen an dieser Stelle zuerst die Variablen definiert werden. Anschließend werden verschiedene arithmetische und Zeichenketten-Operationen angewandt. Hierbei soll es vor allem darum gehen, dem Benutzer die verschiedenen Änderungen von Variablenwerten deutlich zu machen.

Visualisierung

Die Visualisierung basiert auf der Regal-Metapher aus dem Modul der Wertzuweisungen. Hinzu kommt, dass die entsprechenden Operationen zwischen den Variablen sichtbar gemacht werden müssen. Dies geschieht innerhalb des Regalfaches, in dem das Ergebnis der Operation gespeichert werden soll.

3. Modul: Einseitige Verzweigungen

Mithilfe eines einfachen Beispieles wird dem Benutzer in diesem Modul eine Einführung in das Thema Verzweigungen geboten. Dafür werden als erstes Vergleichsoperatoren und das Schlüsselwort **if** vorgestellt. Außerdem wird der Begriff *Bedingung* und dessen Rolle bei Verzweigungen erläutert. Am Schluss wird zusätzlich erklärt, dass man mithilfe des Und- (and) und Oder-Operators (or) verschiedene Bedingungen verknüpfen kann.

Aufgabe

Wie in der Diplomarbeit von Claudia Schindler [11] mehrfach angesprochen wurde, wünschten sich einige der gefragten Testpersonen Programmbeispiele, die näher am Leben der SchülerInnen sind. Im Besonderen erhielt die Thermometer-Metapher bei der Erläuterung von Verzweigungen eine mittelmäßige Kritik. Aus diesem Grund wurde das Beispiel und die damit verbundene Aufgabe geändert. Als Metapher dient der Akku eines beliebigen Smartphones. Der Benutzer soll zuerst eine Variable mit einem Akkustand und einer Schwelle (jeweils in Prozent) festlegen. Anschließend wird geprüft, ob der Akkustand sich unterhalb der Schwelle befindet und eine entsprechende Meldung ausgegeben werden muss.

Visualisierung

Wie bereits in der Aufgabe beschrieben, wird der Akkustand eines Smartphones überprüft. Dafür muss in der Visualisierung ein Smartphone angezeigt werden. In der Mitte des Smartphones befindet sich eine große Batterie, die sowohl den Wert des Akkustandes, als auch den Wert der Schwelle wiedergibt. Am oberen Rand des Smartphones befindet sich zusätzlich eine Benachrichtigungsleiste, in der ein Hinweis dargestellt wird, wenn der Akkustand unter der Schwelle liegt.

4. Modul: Zweiseitige Verzweigungen

Bei der zweiseitigen Verzweigung wird das Schlüsselwort **else** vorgestellt und anhand eines Beispiels erläutert.

Aufgabe

Die Aufgabe entspricht der Aufgabenstellung von einseitigen Verzweigungen. Lediglich wurde eine Alternativbedingung hinzugefügt. Als erstes überprüft der Benutzer, ob der Akkustand über der Schwelle ist. Wenn dies wahr ist, wird eine Meldung ausgegeben, dass der Akku noch nicht geladen werden muss. Wenn die Bedingung hingegen falsch ist, wird die Meldung angezeigt, dass der Akku geladen werden muss.

Visualisierung

Auch bei der Visualisierung verändert sich nichts Wesentliches im Vergleich zu den einseitigen Verzweigungen. In der Benachrichtigungsleiste wird der entsprechende Hinweis angezeigt, ob der Akku geladen werden muss oder nicht.

5. Modul: Mehrseitige Verzweigungen

Als erstes werden in diesem Modul die zusammengehörigen Schlüsselwörter **switch** und **case** vorgestellt, die mehrseitige Verzweigungen ermöglichen. Auf die Else-If-Anweisungen wurde an dieser Stelle verzichtet, da

Switch-Case-Anweisungen in vielen Programmiersprachen Verwendung finden und somit eine weitere Art der Verzweigung vorgestellt werden kann.

Am Beispiel einer Testbewertung wird die Funktionsweise von mehrseitigen Verzweigungen erläutert. Anhand einer gegebenen Punktezahl wird mithilfe von einer Switch-Case-Anweisung die Note festgelegt. Wichtig an dieser Stelle ist, dem Benutzer deutlich zu machen, dass die restlichen Bedingungen nicht mehr überprüft werden, sobald eine Bedingung wahr ist.

Aufgabe

Die Aufgabe von zweiseitigen Verzweigungen wird an dieser Stelle wiederum erweitert. Wenn der Akkustand zwischen 100% und der angegebenen Schwelle liegt, soll der Akkustand in der Meldung ausgegeben werden. Wenn der Akkustand die Schwelle erreicht, soll die Meldung „Akku sollte demnächst geladen werden!“ ausgegeben werden. In allen anderen Fällen (der Akkustand ist kleiner als die Schwelle) soll angegeben werden, dass der Akku geladen werden muss.

Visualisierung

In der Visualisierung können nun drei verschiedene Benachrichtigungen angezeigt werden, je nach dem wie groß der Akkustand und die Schwelle gewählt wurden.

6. Modul: Kopfgesteuerte Schleifen

Bei den kopfgesteuerten Schleifen geht es im Speziellen um While-Do-Schleifen. Zu Beginn wird die generelle Funktionsweise von Schleifen erläutert. Anschließend wird anhand eines Beispiels die While-Do-Schleife erklärt.

Aufgabe

Wie bereits in der Bewertung der Werkzeugvorlage angemerkt wurde, bekam die Wasserglas-Metapher eine sehr gute Bewertung. Aus diesem Grund wird diese auch in der entstehenden Webanwendung genutzt. Die Werte sollen jedoch nur soweit anpassbar sein, dass das Wasserglas nicht überlaufen kann. Die Aufgabe besteht darin, das Glas in jedem Schleifendurchlauf mit einem Schluck Wasser zu befüllen, bis der Inhalt den Füllstrich von *200 ml* überschreitet.

Visualisierung

Die Darstellung besteht am Anfang aus einem leeren Glas. Dieses wird in jedem Schritt mit etwas Wasser befüllt. Am oberen Rand des Glases wird ein Füllstrich mit dem Wert *200 ml* angezeigt. Das Glas kann aufgrund der ausgewählten Variablen nicht überlaufen.

7. Modul: Fußgesteuerte Schleifen

In diesem Modul soll dem Nutzer der Unterschied zwischen kopf- und fußgesteuerten Schleifen deutlich gemacht werden. Fußgesteuerte Schleifen werden auch nachprüfende oder Do-While-Schleifen genannt.

Aufgabe und Visualisierung

Die Aufgabe und Visualisierung sind die gleichen wie im Modul „Kopfgesteuerte Schleifen“.

8. Modul: Zählschleifen

Bei der Zählschleife wird die in vielen Programmiersprachen bekannte For-Schleife vorgestellt. Diese wird jedoch in einer vereinfachten Form dargestellt.

```
for ( var index = 1 to 10 ) { ... }
```

Zuerst wird die Schleife in einem Beispiel nur dafür verwendet, den gleichen Codeblock mehrfach auszuführen. Anschließend wird jedoch auch die Index-Variable mit in Berechnungen einbezogen, um die Mächtigkeit dieser Schleifenvariante deutlich zu machen.

Aufgabe

Im Gegensatz zu den While-Schleifen geht es bei der Aufgabe hier darum, das Wasserglas zu leeren. Der Benutzer definiert dafür die Schluck-Größe und die Anzahl der Durchläufe. Anschließend wird der Inhalt dementsprechend oft um einen Schluck verringert. Die Variablen können dabei nur so ausgewählt werden, dass der Inhalt niemals kleiner als *0 ml* werden kann.

Visualisierung

Wie bei der Visualisierungen der While-Schleifen, wird auch hier ein Wasserglas dargestellt. Zu Beginn ist dieses jedoch bis zu dem Füllstrich von *200 ml* gefüllt. In jedem Schleifendurchlauf wird der Wasserstand verringert.

9. Modul: Funktionen

In diesem Modul werden den SchülerInnen die Definition und Verwendung von Funktionen erklärt. Dies erfolgt anhand eines einfachen Beispieles, in dem eine Funktion zwei Werte addiert und zurück gibt. Der Vorteil, dass Funktionen in verschiedenen Situationen verwendet werden können und somit viele Codezeilen ersparen

können, wird hervorgehoben. Um die Reihenfolge des Codes für den Benutzer verständlich zu machen, müssen auch Funktionen definiert werden, bevor sie verwendet werden können.

Aufgabe

Die Module zu Unterprogrammen wurden in der wissenschaftlichen Arbeit von Sebastian Lehmann [3] ergänzt. Als Visualisierung wählte er dafür, wie im Modul für Wertzuweisungen, die farbigen Kästen aus. Diese Darstellung bietet sich in diesem Fall jedoch nur bedingt an. Zum einen wurde dieses Beispiel schon in anderen Modulen verwendet. Für die SchülerInnen ist es jedoch einfacher, wenn sie Modulen unterschiedliche Beispiele zuordnen können. Dadurch können sie zum Beispiel das Wasserglas sofort mit den Schleifen verknüpfen und das gelernte Wissen somit schneller abrufen. Des Weiteren werden die Informationen über die Variablen bereits in einer Tabelle (in diesem Fall dem Protokoll) angezeigt und müssen nicht zusätzlich noch einmal dargestellt werden. In der Aufgabe in diesem Modul wird es somit um die Berechnung des Flächeninhalts eines Rechtecks gehen. Zuerst muss dafür die entsprechende Funktion deklariert werden. Anschließend wird diese mit den Werten für Breite und Länge aufgerufen und der Rückgabewert in einer Variable gespeichert.

Visualisierung

In der Visualisierung wird ein Rechteck dargestellt. Am Rand stehen die Angaben für die Länge und die Breite des Rechtecks. Wenn die Berechnung erfolgt, wird der Flächeninhalt auf dem Rechteck dargestellt.

10. Modul: Prozeduren mit Werteparameter

Zu Beginn wird der Unterschied von Prozeduren zu Funktionen erklärt: Es gibt keinen Rückgabewert und das Schlüsselwort zur Deklaration lautet **procedure** und nicht **function**. Das Beispiel an dem Prozeduren anschließend weiter erläutert werden, orientiert sich an dem Beispiel aus dem Modul für Funktionen. Der Hauptunterschied besteht darin, dass innerhalb der Prozedur die Variable *summe* definiert wird, um sie anschließend zu berechnen und auszugeben. Weiterhin wird hier bereits auf den Unterschied zwischen Werte- und Referenzparametern hingewiesen.

Aufgabe

Die Aufgabe besteht wie im Modul für Funktionen darin, den Flächeninhalt eines Rechtecks zu berechnen. Dieses mal muss der Benutzer jedoch darauf achten, dass die Variable für den Flächeninhalt innerhalb der Prozedur definiert und ausgegeben werden muss, da es keinen Rückgabewert gibt.

Visualisierung

Die Visualisierung ist identisch zu der Darstellung vom Modul für Funktionen.

11. Modul: Prozeduren mit Referenzparameter

In diesem Modul wird noch einmal vertieft auf den Unterschied zwischen Werte- und Referenzparametern eingegangen. Die Referenz kann man in der hier verwendeten Pseudosprache einer Funktion bzw. Prozedur übergeben, in dem der entsprechenden Variable ein Sternchen (*) vorangestellt wird. Die Summe in dem Beispiel muss somit nicht mehr innerhalb der Prozedur ausgegeben, sondern kann einfach als Referenzparameter übergeben werden. Da dieser Vorteil vor allem bei der Verwendung von Prozeduren deutlich wird, wurde das Modul „Funktionen mit Referenzparametern“ nicht implementiert. Es geht bei diesem Modul hauptsächlich sowieso darum, dem Benutzer den Unterschied von Werte- und Referenzparameter zu verdeutlichen.

Aufgabe

Die Aufgabe ist es diesmal, den Flächeninhalt von zwei verschiedenen Rechtecken zu berechnen und den gesamten Flächeninhalt in einer Variable zu speichern. Diese soll dabei als Referenzparameter der Prozedur übergeben werden.

Visualisierung

Bei der Visualisierung werden in diesem Modul zwei verschiedene Rechtecke dargestellt. Die restliche Darstellung entspricht derjenigen in den beiden vorherigen Modulen.

12. Modul: Objekte

Das Modul für Objekte dient hauptsächlich der Vorbereitung auf das Modul für verkettete Listen. Objekte werden wie Variablen mit dem Schlüsselwort **var** definiert, können jedoch mehrere Eigenschaften mit dazugehörigen Werten beinhalten. Diese Datenstruktur wird dem Benutzer anhand eines Auto-Objektes in einem Beispiel deutlich gemacht. Ein Auto kann verschiedene Eigenschaften wie Farbe oder Typ besitzen. Anschließend wird erklärt, wie Werte von Eigenschaften geändert werden können. Um die Voraussetzungen für das 13. Modul zu schaffen, wird außerdem der leere Wert **null** erläutert.

Aufgabe

Die Aufgabe besteht darin, ein Schuh-Objekt mit einer bestimmten Größe und einer Farbe zu erstellen. Anschließend sollen die Werte geändert werden.

Visualisierung

Es wird ein Schuh dargestellt, der die entsprechenden Eigenschaften annimmt, wenn sie im Programmcode geändert werden.

13. Modul: Verkettete Listen erstellen

Zu Beginn werden dem Benutzer verschiedene Einsatzmöglichkeiten von verketteten Listen vorgestellt. Anhand des Beispiels einer Aufgabenliste wird anschließend die Verwendung erläutert. Die einzelnen Elemente werden durch Objekte dargestellt. Es wurde bewusst auf Datenstrukturen wie **struct** oder **record** verzichtet, um den SchülerInnen nicht mit zu vielen unterschiedlichen Konstruktionen zu überfordern. Anschließend wird die Bedeutung von Zeigern erläutert und in einem Beispiel eine Aufgabenliste mit einem Element angelegt.

Aufgabe

Die Aufgabe besteht darin, eine Wiedergabeliste eines Musikspieler zu erstellen und anschließend mit Songs zu füllen. Dafür muss zuerst ein Musikspieler-Objekt angelegt werden, das immer auf das erste Element der Liste zeigt. Anschließend wird das erste Song-Objekt erstellt und der Zeiger des Musikspielers dementsprechend angepasst. Am Ende soll ein weiterer Song hinzugefügt werden, der jedoch nicht an das Ende, sondern an den Anfang der Liste gesetzt wird.

Visualisierung

Es wird ein Smartphone mit einem gestarteten Musikspieler dargestellt. In einer Wiedergabeliste werden die aktuellen Songtitel aufgelistet. Mit einem Pfeil wird der Titel angezeigt, der aktuell abgespielt wird (der oberste Titel).

14. Modul: Verkettete Listen durchsuchen

Anhand eines Beispiels wird dem Benutzer erläutert, wie sich mithilfe von Verzweigungen und Schleifen eine verkettete Liste durchsuchen lässt. Diese Suche wird gleichzeitig dafür genutzt, um zu erklären, wie Elemente mit einer bestimmten Eigenschaft aus der Liste entfernt werden können.

Aufgabe

Die Aufgabe besteht darin, eine existierende Wiedergabeliste nach einem Song mit einem bestimmten Titel zu durchsuchen und ihn anschließend aus der Liste zu entfernen. Die Schwierigkeit besteht darin, die Klammern, Schleifen und Verzweigungen richtig anzuordnen.

Visualisierung

Es wird ein Smartphone mit einem gestarteten Musikspieler dargestellt. In einer Wiedergabeliste werden die aktuellen Songtitel aufgelistet. Mit einem Pfeil wird der Titel angezeigt, der aktuell abgespielt wird. Wenn die Suche gestartet wird, wird der zu überprüfende Song unterstrichen. Gelöschte Songs werden semi-transparent dargestellt.

4.1.3 Aufbau der Anwendung

Die Webanwendung soll aus drei verschiedenen Bestandteilen bestehen: Der Startanimation, dem Menü und der Modulansicht. Die Startanimation wird nur beim Aufruf der Anwendung angezeigt und kann nicht über die Navigationsstruktur erneut gestartet werden.

Nach der Startanimation kann das Menü am oberen Rand geöffnet werden. Der entsprechende Entwurf ist in Abbildung 4.1 zu sehen. Das Menü besteht aus fünf Kategorien: Variablen, Verzweigungen, Schleifen, Unterprogramme und Datenstrukturen. Unterhalb der jeweiligen Kategorie sind die dazugehörigen Module aufgelistet. Das Menü ist von jedem Punkt der Anwendung erreichbar. Somit kann jederzeit mit maximal zwei Mausklicks zu einem gewünschten Modul gewechselt werden.

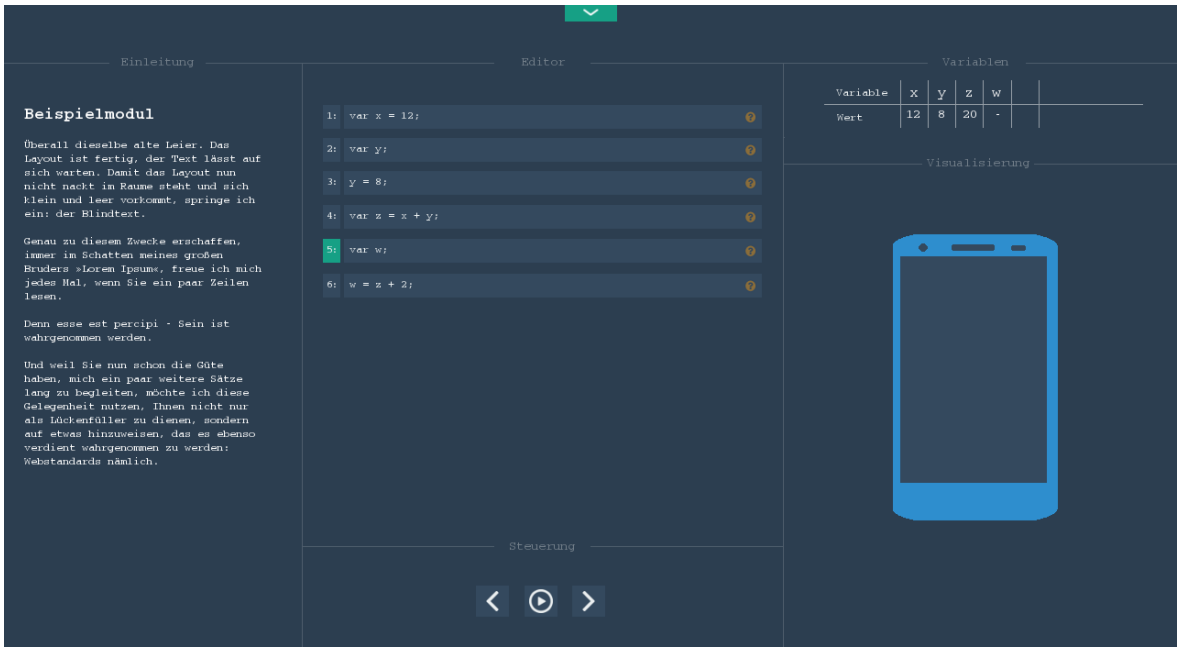
Abbildung 4.1: Entwurf - Navigation / Menü

Die Module haben alle den gleichen Grundaufbau. In Abbildung 4.2 wird ein Beispielmodul dargestellt. Auf der linken Seite befindet sich die Einleitung in das Modul. Darin wird das vorausgesetzte Wissen vermittelt, das für die Bearbeitung des Moduls notwendig ist. Zusätzlich werden hier nützliche Informationen gegeben und die Aufgabenstellung des Moduls erläutert.

Im mittleren Abschnitt der Ansicht befindet sich der Editor. Dort werden verschiedene Codeblöcke angezeigt, die der Benutzer mittels *Drag & Drop* sortieren kann. Darunter befindet sich die Steuerung. Bei der richtigen Sortierung der Codeblöcke kann das Programm dort gestartet werden. Die Codezeilen werden anschließend in der angegeben Reihenfolge automatisch abgespielt. Der Benutzer hat jedoch die Möglichkeit, die Ausführung zu pausieren und die Schritte manuell zu aktivieren. Die aktive Codezeile wird farbig hervorgehoben.

In der oberen rechten Ecke befindet sich der Abschnitt für die Darstellung der Variablen und den dazugehörigen Werten. Dadurch kann der Benutzer in jedem Schritt nachvollziehen, welche Variablen gerade definiert sind und welche Werte sie besitzen. Darunter befindet sich die Visualisierung. Diese wird ebenfalls dem jeweiligen aktiven Schritt angepasst.

Abbildung 4.2: Entwurf - Ansicht eines Beispielmoduls



4.2 Prototyp

Bei der entstandenen Webanwendung **EduCode** handelt es sich um einen stark ausgereiften Prototypen. Die implementierten Module können vollständig ausgeführt werden. Durch die Evaluation werden Fehler und Verbesserungsvorschläge herausgestellt, durch die der Prototypen letztendlich zu einer vollwertigen Anwendung wird.

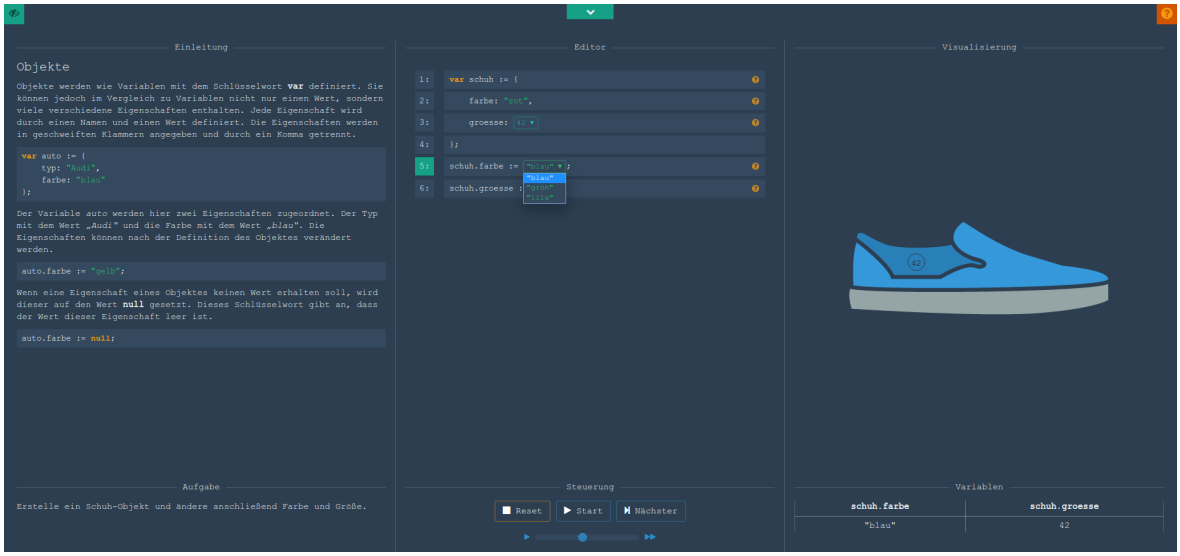
EduCode wurde als Webanwendung mit den Sprachen HTML, CSS und Javascript implementiert. Auf serverseitige Ausführungen mittels PHP oder Ajax wurde verzichtet. Dadurch kann die Anwendung lokal im Browser aufgerufen werden. Eine plattformunabhängige Verwendung ist somit möglich.

Ein weiterer Anspruch an die Anwendung war die mögliche Einbettung in E-Learning Plattformen. Durch den Einsatz einfacher und clientseitiger Techniken wird diese Integration ermöglicht. Getestet wurde die Einbettung von EduCode in die Bildungsplattform Opal als SCORM-Lerninhalt.

4.2.1 Umsetzung der Anforderungen

Die Muss- und Soll-Anforderungen an die Anwendung wurden alle umgesetzt. Die Module zum Durchsuchen und Löschen von verketteten Listen wurden zusammengefasst, da das Löschen eines Elements aus der Liste die Suche beinhaltet. Im Editor-Abschnitt, in der die einzelnen Codeblöcke dargestellt werden, kann der Benutzer ausgewählte Variablenwerte mithilfe eines Auswahlfeldes verändern. Dadurch können verschiedene Abläufe des Programms hervorgerufen und die dargestellte Problemstellung verdeutlicht werden.
Die Ansicht eines Moduls mit Auswahlfeld wird in Abbildung 4.3 dargestellt.

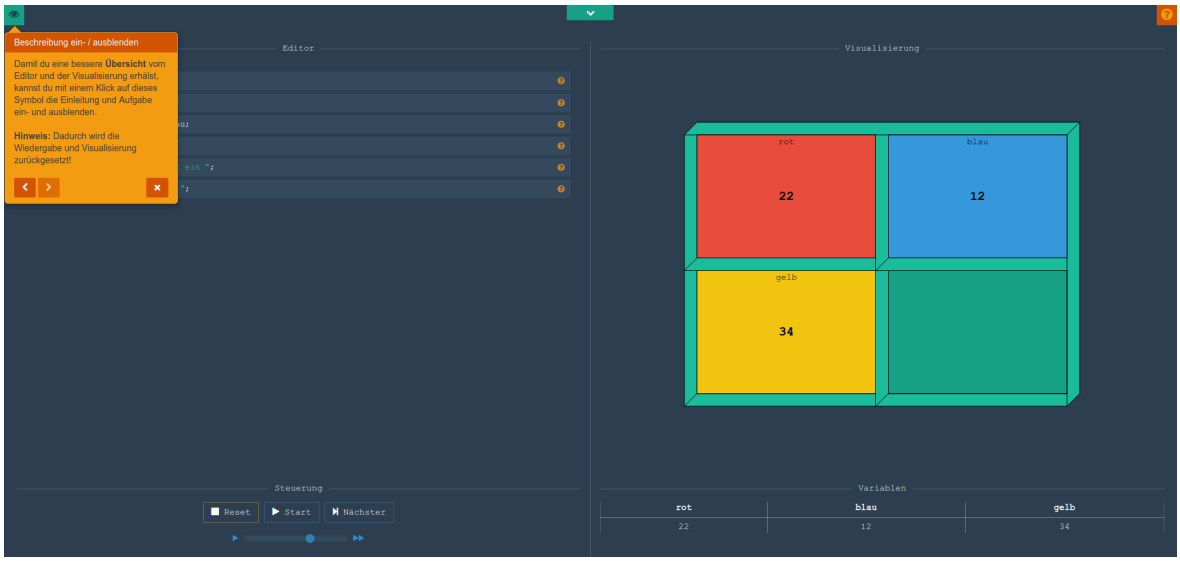
Abbildung 4.3: EduCode - Ansicht des Moduls **Objekte** mit Auswahlfeld



Das Programm kann in den jeweiligen Modulen automatisch oder manuell abgespielt werden. Die Anpassung der Wiedergabegeschwindigkeit ist jederzeit möglich. Diese beeinflusst auch die Animationsgeschwindigkeit bei der manuellen Wiedergabe, bei der jeder Codeblock mit einem Klick auf *Nächster* einzeln ausgeführt wird. Die Kann-Anforderung eines Zurück-Buttons wurde nicht umgesetzt, da die Aufzeichnung eines Ausführungsverlaufs erheblichen Mehraufwand bedeutet.

Neben den Muss- und Soll-Anforderungen wurden weitere Features implementiert. Dazu gehört ein Tutorial, das den Benutzer beim ersten Aufruf der Anwendung durch die einzelnen Bestandteile von Educode führt. Das Tutorial kann jederzeit unterbrochen und erneut gestartet werden. Durch die Verwendung von Cookies wird das Tutorial beim zweiten Aufruf der Anwendung nicht noch einmal angezeigt. Um die Übersichtlichkeit eines Moduls zu gewährleisten, kann der Bereich **Einleitung** und **Aufgabe** ausgeblendet werden. Dies ermöglicht zusätzlich den Einsatz auf geringer aufgelösten Monitoren. Die restlichen Kann-Anforderungen dienen als Vorlage für zukünftige Erweiterungen von EduCode.
Das Tutorial und die ausgeblendeten Bereiche werden in Abbildung 4.4 dargestellt.

Abbildung 4.4: EduCode - Ansicht des Moduls **Operatoren** mit Tutorial und ausgeblendeter Einleitung und Aufgabenstellung



4.2.2 Technische Realisierung

Jedes Modul von EduCode besteht aus einer HTML-Datei, die die Modulstruktur, Einleitung und Aufgabe beinhaltet. Alle weiteren Bestandteile werden dynamisch über eine spezifische Javascript-Datei erstellt. Jedes Modul besitzt eine eigene Javascript-Klasse, die jedoch alle von einem Basismodul erben. Dadurch können grundlegende Funktionen, die in allen Modulen verwendet werden, in dem Basismodul definiert werden. Dies trifft zum Beispiel auf die Steuerung zu. Die Visualisierung hingegen unterscheidet sich in jedem Modul und wird daher in der eigenen Javascript-Klasse implementiert. Durch diese Struktur können weitere Module auf einfache Art hinzugefügt oder entfernt werden.

Um die Visualisierung, das Tutorial oder auch das *Drag & Drop* zu realisieren, wurden verschiedene Bibliotheken verwendet, die in Anhang D.3 aufgelistet werden.

Um die Formatierungen der Module zu gestalten, wurde bei der Implementierung SCSS (Sassy CSS) verwendet. Dies ermöglicht im Vergleich zu reinem CSS eine übersichtlichere und schnellere Implementierung. Ein SASS-Compiler minimiert die SCSS-Dateien anschließend zu einer einzelnen CSS-Datei. Dadurch wird die Ladezeit der Anwendung verkürzt.

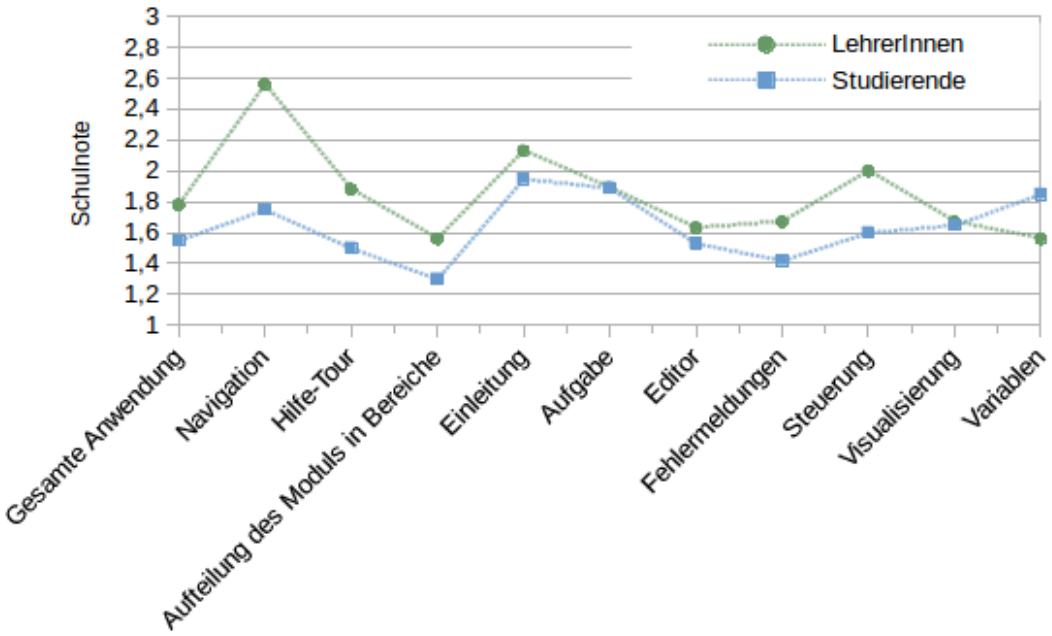
5 Evaluation

Um existierende Fehler zu erkennen und Verbesserungsvorschläge zu sammeln, wurde nach der Implementierung eine Evaluation durchgeführt. Dafür wurde ein Fragebogen erstellt, mit dem das Erscheinungsbild, die Funktionen und die einzelnen Module bewertet werden konnten. Die Bewertung erfolgt dabei mithilfe der Schulnoten von 1 (Sehr gut) bis 6 (Ungenügend). Zusätzlich steht das Feld „*Weiß nicht*“ zur Auswahl, um eine Unentschlossenheit oder Unsicherheit beim Verständnis ausdrücken zu können. Neben dieser Bewertungsskala enthält der Fragebogen jedoch auch Ja-/Nein-Fragen (z.B. ob die Testperson das Tool im Unterricht einsetzen würde) und Freitextfelder. Durch diese freien Anmerkungen erhält man bei der Auswertung spezifische Hinweise und Verbesserungsvorschläge. Am Ende der Umfrage wird nach der aktuellen Tätigkeit und dem Alter der TeilnehmerInnen gefragt. Der gesamte Fragebogen ist im Anhang C.1 zu finden.

Der Fragebogen wurde auf der Webseite *Umfrage Online*[1] erstellt, um sie auf eine einfache Weise an potentielle TeilnehmerInnen versenden zu können. Da eine Beurteilung die vorherige Verwendung von Educode voraussetzt, wurde das Tool bei *bplaced*[2] im *World Wide Web* gehostet. Der Link zum Online-Fragebogen und zum Tool wurde an LehrerInnen der Informatik und an Studierende des Lehramts Informatik an der TU Dresden per E-Mail geschickt. Zusätzlich wurde im Rahmen einer Übung die Umfrage mit angehenden LehrerInnen der Informatik durchgeführt. Im Folgenden werden die Ergebnisse der Umfrage präsentiert.

[1] https://www.umfrageonline.com/
[2] http://www.bplaced.net/

Abbildung 5.1: Bewertung des Erscheinungsbilds

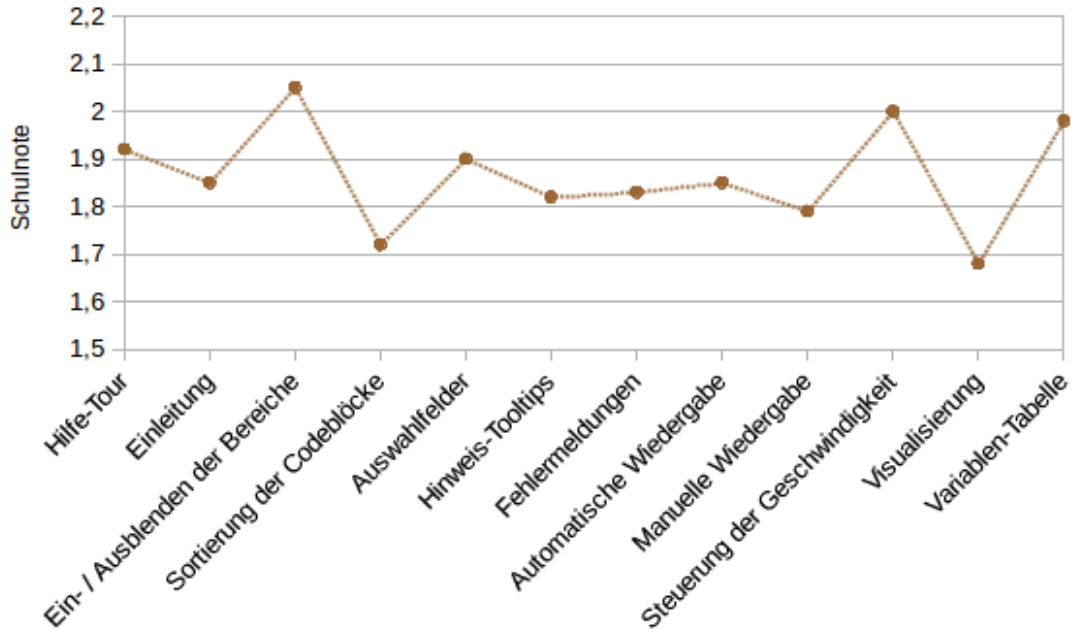


5.1 Auswertung

An der Umfrage haben 42 Personen teilgenommen. Darunter sind 9 LehrerInnen für Informatik und 20 Studierende des Lehramts Informatik. Ein Lehrer bzw. eine Lehrerin scheint die Umfrage an seinen/ihren Grundkurs der 11. Klasse weitergegeben zu haben, wodurch auch 9 SchülerInnen unter den TeilnehmerInnen vertreten sind. Alle weiteren TeilnehmerInnen (4) haben keine Auskunft über ihre derzeitige Tätigkeit gemacht. Die Umfrage ist aufgrund der Durchführungsart nicht statistisch repräsentativ und gibt daher lediglich eine Orientierung über die Bewertung des Tools.

Die Auswertung der einzelnen Bereiche des Tools erfolgt in den folgenden Abschnitten. Durchschnittlich erhält Educode eine Bewertung von 1,83 (Gut). Die LehrerInnen gaben eine durchschnittliche Bewertung von 1,88 und die Studierenden von 1,73 ab. Bei den SchülerInnen liegt dieser Wert hingegen bei 2,03. Durch auffällig lineare Bewertungen unter den SchülerInnen wird deutlich, dass einige die Bewertung wahrscheinlich nicht mit vollem Ernst ausgeführt haben. Zusätzlich war diese Umfrage ursprünglich nicht für SchülerInnen konzipiert. Aufgrund der zusätzlichen Perspektive, die auch die Einschätzung von SchülerInnen berücksichtigt, wird diese Bewertung trotzdem nicht ausgeschlossen. Die Priorität der Auswertung liegt somit bei der Bewertung von LehrerInnen und Studierenden. Die gesamten Umfrageergebnisse sind im Anhang C.2 dargestellt.

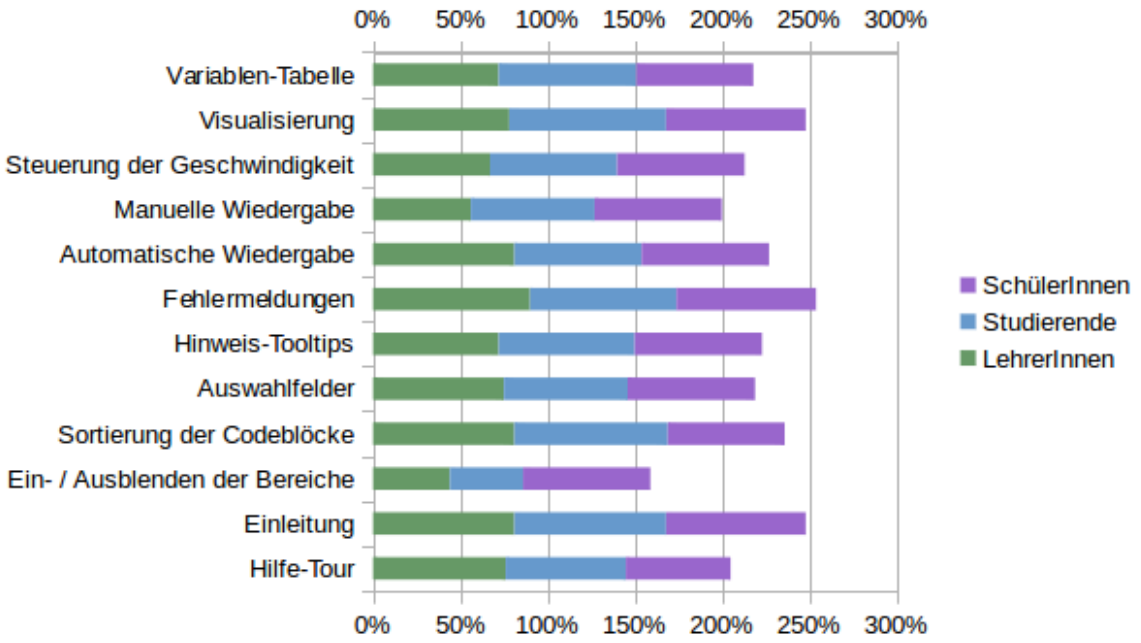
Abbildung 5.2: Bewertung der Funktionen



5.1.1 Erscheinungsbild

Im ersten Teil der Umfrage ging es um das Erscheinungsbild verschiedener Komponenten von Educode. Mit einer durchschnittlichen Note von 1,74 erhält das Aussehen des Tools eine gute Bewertung. In Abbildung 5.1 werden die Bewertungen von LehrerInnen und Studierenden zum Erscheinungsbild der Anwendung visualisiert. Die Navigation wird von den LehrerInnen (2,56) deutlich schlechter bewertet, als von den Studierenden (1,75). Jedoch ist auch unter den LehrerInnen eine große Uneinigkeit festzustellen, da die Standardabweichung mit einem Wert von 1,33 sehr hoch ist. In den Anmerkungen wurde mehrfach erwähnt, dass das Navigationsmenü zu unscheinbar wäre. Auf das Menü sollte somit beim Start der Anwendung verdeutlicht hingewiesen werden. Bei den Studierenden kam bei der Bewertung des Erscheinungsbilds die Einleitung im Modul am schlechtesten weg (1,95). Als Begründung dafür wurde genannt, dass die Einleitung sehr textlastig ist. Dabei wurden auch verschiedene Verbesserungsvorschläge beschrieben. Zum einen könnte die Einleitung Schritt für Schritt in einem Popup vorgestellt werden. Zum anderen wäre es möglich, nur wichtige Anweisungen darzustellen und erst durch einen Klick auf eine Schaltfläche mehr Informationen einzublenden (*Details on Demand*). In den zusätzlichen Anmerkungen wurde weiterhin erwähnt, dass die Aufgabe und die Variablen-Tabelle besser hervorgehoben werden sollten. Die Variablen dürfen dabei jedoch nicht zu sehr in den Mittelpunkt rücken. Sehr gute Ergebnisse erzielt bei den Studierenden die Aufteilung der Module in Bereiche (1,3). Mit einem Wert von 0,47 ist die Standardabweichung unter den studierenden TeilnehmerInnen sehr gering. Auch die LehrerInnen vergeben in dieser Kategorie ihre beste Durchschnittsnote (1,56).

Abbildung 5.3: Gewichtung der Funktionen



5.1.2 Funktionen

Der zweite Teil des Fragebogens beschäftigt sich mit den Funktionen von Educode. Dazu zählen zum Beispiel die Hilfe-Tour, die Hinweis-Tooltips und auch die Visualisierung. In Abbildung 5.2 sind die durchschnittlichen Noten für die einzelnen Funktionen abgebildet. Die besten Bewertungen erhält die Funktion, die das sortieren der Codeblöcke ermöglicht (1,72) und die Visualisierung (1,68). Die Möglichkeit der Sortierung wird im Besonderen von den Studierenden gut bewertet (1,53). Die LehrerInnen unterstützen hingegen die manuelle Wiedergabe des Programms (1,56). Weniger gut kommt das Ein- und Ausblenden des Einleitungs- und Aufgabenbereichs bei der Bewertung weg (2,05). Durch die Anmerkungen lässt sich dies hauptsächlich dadurch begründen, dass die Aufgabe laut eines Teilnehmenden nicht ausgeblendet werden dürfe. Weiterhin wird die Steuerung der Geschwindigkeit mit 2,0 bewertet. In den Freitextfeldern wurden zur Geschwindigkeitssteuerung jedoch keine Aussagen getroffen. Eventuell sollte diese Funktion intuitiver gestaltet werden, damit der Nutzer die Auswirkungen des Schiebereglers besser nachvollziehen kann. Mit einer Durchschnittsnote von 1,98 landet die Variablen-Tabelle auf den hinteren Plätzen der Funktionen. Dies ist laut der Anmerkungen, wie bereits im vorigen Abschnitt, auf die unscheinbare Platzierung zurückzuführen. Außerdem wird der Vorschlag gemacht, die Tabelle aufgrund der besseren Übersichtlichkeit vertikal aufzubauen. Viele der aufgeführten Anmerkungen beziehen sich auf Funktionen, die in der anfänglichen Hilfe-Tour erläutert werden. Einige der Testpersonen klicken die Tour jedoch vorzeitig weg, da diese für den Anfang zu umfangreich ist. Dadurch war diesen Personen auch die Aufgabe der Sortierung nicht bewusst. Besser wäre es zum Beispiel, ein Modul mithilfe der Tour beispielhaft zu durchlaufen, um die Möglichkeiten und Funktionen deutlich zu machen. Ein weiterer Punkt

richtet sich an die Navigation nach dem Abschluss eines Moduls. TeilnehmerInnen schlagen vor, dass eine Schaltfläche nach der erfolgreichen Absolvierung eines Moduls erscheint, mit der sie zum nächsten Modul übergehen können.

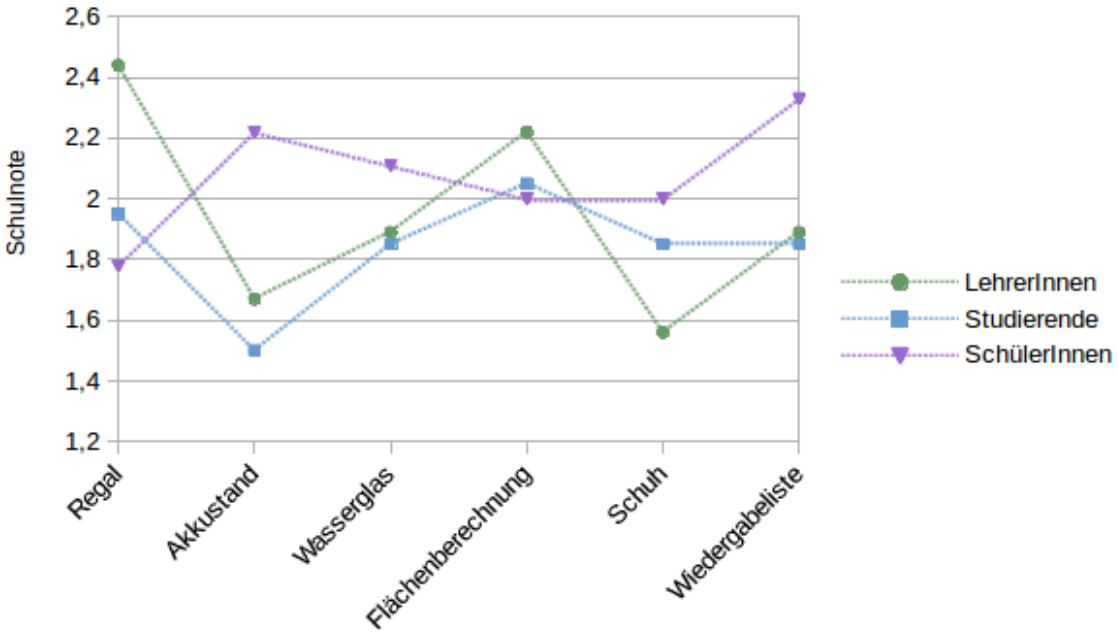
Neben der Bewertung der einzelnen Funktionen, wurde auch nach der Wichtigkeit dieser gefragt. Die Ergebnisse werden dabei in Prozent angegeben, wobei 0% *unwichtig* und 100% *sehr wichtig* bedeutet. In Abbildung 5.3 werden die Ergebnisse dieser Frage visualisiert, wobei zwischen LehrerInnen, Studierenden und SchülerInnen unterschieden wird. Die einzelnen Gewichtungen werden aufsummiert dargestellt, womit der Wert 300% die höchste Wichtigkeit repräsentiert. Hierbei wird deutlich, dass die *Einleitung*, *Fehlermeldungen* und *Visualisierung* von den drei Gruppen als die wichtigsten Funktionen des Tools eingestuft werden. Wie sich bereits aus den Bewertungen erkennen lässt, wird auch hier deutlich, dass das Ausblenden des Einleitungs- und Aufgabenbereichs keine große Bedeutung für die Testpersonen hat. Vor allem die LehrerInnen (49%) und die Studierenden (44%) beeinflussen dieses Ergebnis maßgeblich. Diese Funktion wurde jedoch hauptsächlich für kleinere Bildschirme konzipiert, um eine bessere Übersicht zu gewährleisten. Es ist durchaus möglich, dass die TeilnehmerInnen mit hohen Bildschirm-Auflösungen diese Funktion für überflüssig halten.

5.1.3 Module

Im dritten Abschnitt der Umfrage ging es um die Bewertung der einzelnen Module und deren Metaphern, die den Programmcode und die Visualisierung bestimmen. In Abbildung 5.4 werden die Bewertungen der einzelnen Personengruppen für die Metaphern dargestellt. Das Beispiel mit dem Akkustand in den Modulen zu den Verzweigungen erhält dabei die beste Bewertung (1,72), obwohl die SchülerInnen, im Gegensatz zu den anderen beiden Gruppen, die Metapher nur mit 2,22 bewerten. Dies wird vor allem durch die gute Bewertungen der Studierenden hervorgerufen (1,5). Das beste Beispiel für die SchülerInnen ist hingegen das Regal in den Modulen für Wertzuweisungen und Operatoren (1,78), da es wohl am verständlichsten ist. Die LehrerInnen sind da jedoch einer anderen Meinung und bewerten diese Metapher mit 2,44. Kisten oder Schuhkartons wären laut der Anmerkungen bessere Metaphern, da zum einen der Speicherplatzbedarf durch unterschiedliche Größen dargestellt werden könnte und zum anderen die Variablen in der Praxis nicht physisch zusammenhängen. Die Gruppe der LehrerInnen vergibt ihre beste Note an die Schuh-Metapher im Objekt-Modul (1,56).

Neben den Bewertungen der Metaphern, konnten die TeilnehmerInnen zusätzliche Anmerkungen zu den Modulen machen. Wie bereits bei der Evaluation in der Arbeit von Claudia Schindler [11], wurde sich auch hier der Einsatz von Variablentypen gewünscht. Dies ist ein gutes Thema für ein zusätzliches Modul in der Modulgruppe *Variablen*. In dem Modul *Wertzuweisungen* wäre dies jedoch für den Einstieg zu umfangreich für Anfänger. Weiterhin wurde in dieser Modulgruppe angemerkt, dass die Variablennamen (blau, gelb, rot) bereits die Werte der Variablen suggerieren. Als eine Alternative wird die Verwendung von Briefkästen mit Namensschildern als Metapher vorgeschlagen. Es besteht jedoch auch die Möglichkeit, die Variablennamen in der Einleitung umzubenennen, um den SchülerInnen zu verdeutlichen, dass dieser nicht den Wert bestimmt.

Abbildung 5.4: Bewertung der Modulmetaphern



Die Module zu den Verzweigungen werden in den Anmerkungen, bis auf die Farbauswahl bei der Visualisierung des Smartphones und den teilweise irreführenden Fehlermeldungen, hauptsächlich gelobt. Eine Testperson kritisiert den ständigen Wechsel der Metaphern in den verschiedenen Modulen. Vor allem bei den Verzweigungen und den Schleifen wünscht sich die Person die gleiche Metapher. Die Entscheidung, verschiedene Metaphern zu verwenden, wurde bereits in der Entwurfsphase begründet. Die SchülerInnen können so zum Beispiel durch das Wasserglas direkt eine Verbindung zu den Schleifen aufbauen und sich diese Technik somit besser merken. Bei den Zählschleifen wird sich weiterhin eine Anzeige des Zählers in der Animation gewünscht. Dies könnte zum Beispiel dadurch geschehen, dass die einzelnen Schlucke bereits beim Start am Glas markiert werden. Auf eine Anzeige des Zählers in der Visualisierung wurde bisher verzichtet, da sie ja bereits in der Variablen-Tabelle dargestellt wird. Eine Testperson wünschte sich außerdem zusätzliche Werte bei der Auswahl des Zähler-Wertes. Dadurch könnte deutlich gemacht werden, in welchen Fällen die Schleife nur einmal oder gar nicht durchlaufen wird.

Bei der Flächeninhaltsberechnung in der Modulgruppe *Unterprogramme* wird sich ein Beispiel aus der Lebenswelt der SchülerInnen gewünscht, ähnlich wie mit dem Akku des Smartphones bei den Verzweigungen. Eine weitere Möglichkeit wäre die Erweiterung um Funktionen zur Berechnung verschiedener Werte. In den Anmerkungen wird von einer Testperson darauf hingewiesen, dass der Aufruf des Unterprogramms von dem Unterprogramm selbst getrennt dargestellt werden sollte. Dies erfordert jedoch aufgrund des Modulaufbaus einen deutlichen Mehraufwand.

In dem Objekt-Modul wird die Größe des Schuhs allein durch ein Label gekennzeichnet. Von einer Testperson wird sich jedoch gewünscht, dass dadurch auch die Größe des Schuhs angepasst wird, wenn sich der dementsprechende Wert ändert. Außerdem fragen sich einige, warum die Größe in dem Schuh-Objekt nach der Farbe zugeordnet werden muss. Diese Reihenfolge wurde aus dem Grund festgelegt, weil die erste Eigenschaft im Objekt mit einem Komma abgeschlossen wird. Es wäre jedoch möglich, das Komma bei der Sortierung automatisch der ersten Eigenschaft zuzuordnen, so dass die Farbe auch nach der Größe festgelegt werden kann.

5.1.4 Allgemein

Im vierten Teil der Umfrage wurden allgemeine Fragen zu Educode gestellt. Die erste Frage sollte beantworten, ob die Testpersonen die Umsetzung des Tools als Webanwendung als sinnvoll erachten. Dies wurde von allen TeilnehmerInnen mit *Ja* beantwortet. Bei der zweiten Frage ging es darum, ob die Testpersonen das Tool im Unterricht einsetzen würde. Damit soll geklärt werden, inwiefern Educode bereits für den Einsatz gewappnet ist. Zwei der 39 TeilnehmerInnen beantworteten diese Frage mit *Nein*. Diese Personen wurden im Nachhinein dazu aufgefordert ihre Antwort zu begründen. Bei der ersten Person handelt es sich um einen Schüler bzw. eine Schülerin, die ihre Antwort auch mit der aktuellen Tätigkeit erklärt. Die zweite Person gibt an, dass sie die Anwendung nur als Ergänzung für Zuhause einsetzen würde, solange nicht die Codebeispiele in der Einleitung oder das Programm ohne Sortierung ausgeführt werden kann.

Weiterhin wurden die TeilnehmerInnen nach Verbesserungsvorschlägen für die Anwendung befragt. Dabei wurde ein Expertenmodus vorgeschlagen, in denen der Nutzer nicht nur Codezeilen bzw. -blöcke, sondern auch feinere Fragmente anordnen muss. Auch wünschten sich TeilnehmerInnen komplexere Aufgaben für den Abschluss. Educode ist zwar hauptsächlich für den Einstieg in die Thematik der Programmierung gedacht, kann jedoch auf eine einfache Weise um weitere Module ergänzt werden. Eine Funktion, die auch bereits in der Anforderungsanalyse erwähnt wurde, ist die mögliche Auswahl der Programmiersprache. Da wären für den Einsatz in Schulen im Besonderen Java und Pascal zu nennen. Die Implementierung dieser Funktion ist jedoch mit einem sehr hohen Aufwand verbunden, da die Wahl einer Programmiersprache den Quellcode einiger Module stark verändern würde. Eine Testperson vermisst die Funktion für LehrerInnen, die Lösungen für die Module anzeigen zu lassen. Eine weitere Person wünscht sich prominentere Platzierung wichtiger Hinweise, wie zum Beispiel zur Sortierung der Codezeilen per *Drag & Drop*. Weitere Hilfestellungen können optional angeboten werden, wie dies mit der Hilfetour bereits geschieht.

5.2 Bewertung der Ergebnisse

Neben Fehlerbehebungen werden auch noch einige Verbesserungsvorschläge in dieser Arbeit umgesetzt, die aus der Evaluation resultieren. In der folgenden Liste werden diese Verbesserungen dargestellt:

- Schaltfläche „Zum nächsten Modul“ hinzufügen, wenn Modul abgeschlossen wurde
- Auf wichtige Funktionen zu Beginn in einem Popup hinweisen (z.B. Drag & Drop), alle weiteren Informationen in existierendem Hilfe-Tutorial darstellen
- Bei Klick auf *Start* nicht mit erstem Modul starten, sondern Navigation öffnen
- Möglichkeit erstellen, um das Umsortieren der Codeblöcke zu deaktivieren
- Die Augen-Schaltfläche zum Ausblenden des Einleitungs-Bereichs besser platzieren, um eine bessere Zuordnung zu garantieren
- Die ausgeblendete Einleitung auch beim Wechsel des Moduls beibehalten
- Zählschleife: Bei der Auswahl der Variable *anzahl* den Wert 0 hinzufügen

Neben diesen, gab es noch viele weitere Verbesserungsvorschläge, die jedoch eine erneute Analyse- und Entwurfsphase durchlaufen müssten und somit einen erheblichen Mehraufwand bedeuten. Sie dienen jedoch als Grundlage für zukünftige Arbeiten, die sich das Ziel setzen, Educode weiter auszubauen und zu verbessern. Voraussichtlich im August 2016 wird dieses Tool zudem auf dem sächsischen Bildungsserver hoch geladen, um es allen LehrerInnen für Informatik in Sachsen für ihren Unterricht zur Verfügung zu stellen. Bei dem Einsatz mit SchülerInnen können dadurch weitere Informationen und Bewertungen des Tools gesammelt werden.

6 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Implementierung einer Webanwendung, die SchülerInnen im Informatikunterricht dabei unterstützt, Programmiergrundlagen und Algorithmen mithilfe einfacher Beispiele und Visualisierungen kennenzulernen und zu verstehen. Vor der eigentlichen Implementierung wurde jedoch eine Untersuchung des Informatikunterrichts durchgeführt. Als erstes wurde dafür die aktuelle Situation analysiert. Seit 2007 gelten einheitliche Prüfungsanforderungen im Fach Informatik in allen deutschen Bundesländern. Diese geben Auskunft über das Fachwissen und die Kompetenzen, die von den SchülerInnen in der Abiturprüfung erwartet werden [4]. Weiterhin wurde auch der sächsische Lehrplan für das Fach Informatik am Gymnasium untersucht [7]. Dadurch konnte der Einsatz der Webanwendung zeitlich eingeordnet werden. Ab der 8. Klassenstufe beschäftigen sich die SchülerInnen intensiver mit dem Algorithmusbegriff. In der 9. Klasse erfolgt die formale Beschreibung von Algorithmen und der erste Kontakt mit Programmiersprachen. Nach der aktuellen Situation wurde die Frage untersucht, welche Inhalte im Informatikunterricht vermittelt werden sollten. Der Unterricht sollte sich nicht an kurzfristigen, sondern an konstanten und alltäglichen Entwicklungen orientieren [13]. In einer Untersuchung fanden Zendler et al. heraus, dass das Konzept *Algorithmen* eine zentrale Rolle im Informatikunterricht einnehmen sollte. Dabei schnitt es in allen untersuchten Kriterien (Horizontalkriterium, Vertikalkriterium, Sinnkriterium und Zeitkriterium) sehr gut ab [13]. In den im Jahr 2008 veröffentlichten Bildungsstandards der *Gesellschaft für Informatik e.V.* wurden weitere Empfehlungen für den Informatikunterricht festgelegt [9]. Durch diese wurde die Idee der Entwicklung des Tools als Webanwendung gestützt. Durch die plattformunabhängige Verwendbarkeit wird die Chancengleichheit unter den SchülerInnen vergrößert, da die Anwendung auf fast jedem System ausgeführt werden kann. In den Bildungsstandards wurde außerdem eine Empfehlung bezüglich des Umgangs mit Algorithmen ausgesprochen. In den Klassenstufen 5 bis 8 sollten die SchülerInnen lernen, Algorithmen umgangssprachlich zu beschreiben. Ab der 8. Klasse werden Algorithmen mit formalen Sprachen beschrieben, so dass auch Maschinen diese verstehen können. An diesem Punkt kommt das Tool zum Einsatz. Im dritten Teil der Untersuchung des Informatikunterrichts ging es um das Lehrmaterial. Dabei geht es um die Fragestellung, auf welche Weise den SchülerInnen die Lehrinhalte vermittelt werden

sollen. In den letzten Jahren sind viele Lehrmaterialen entstanden, die SchülerInnen beim Lernprozess unterstützen können. Laut Diethelm et al. kommt es aufgrund einer fehlenden Lehrerperspektive häufig nicht zum Unterrichtseinsatz dieser Materialien [1]. Die LehrerInnen müssen zum Beispiel erfahren, wann sie ein Tool einsetzen können, welches Wissen dabei vermittelt wird und wie sie damit umzugehen haben. Im letzten Teil der Untersuchung des Informatikunterrichts wurde der Einsatz von Visualisierungen untersucht. Nils Faltin kam in seiner Dissertation zu dem Schluss, dass es nicht darum geht welche Visualisierungsform verwendet wird, sondern ob die Visualisierung einfach und verständlich ist. Jedoch können interaktive Visualisierungen den Lernprozess fördern, da die Motivation durch entdeckendes Lernen erhöht wird.

Im Anschluss an die Untersuchung des Informatikunterrichts ging es um die Bewertung aktueller Werkzeuge, die das Ziel verfolgen, Algorithmen und Programmiersprachen zu vermitteln. Die Bewertungsgrundlage setzt sich dabei zum einen aus dem Teil 110 der Norm EN ISO 9241 [6] und zum anderen aus der Einsatzmöglichkeit im Informatikunterricht zusammen. Beide Beurteilungen gingen zu 50% in die Gesamtbewertung ein. Damit geben die Ergebnisse nicht nur darüber Auskunft wie gut sie sind, sondern auch ob sie SchülerInnen in der 7.-9. Klassenstufe bei der Vermittlung von Programmiergrundlagen im Informatikunterricht unterstützen können. Die Testergebnisse zeigen, dass viele Tools Grundlagen der Programmierung voraussetzen. Der Quellcode wird oft nicht weiter erläutert. Eine interessante Funktion lieferte die Plattform *Code* [18], die mithilfe blockbasierter Programmierung Kindern ermöglicht, einfache Algorithmen spielerisch zu verstehen. Hervorzuheben ist weiterhin die Plattform *Khan Academy* [27], die sich mit abwechslungsreichen Modulen und Methoden die beste Bewertung sichert. Einige der getesteten Tools setzen auf Lernvideos. Diese sind für den Informatikunterricht jedoch meist ungeeignet.

Als Vorlage für die Webanwendung diente eine Flash-Anwendung, die im Rahmen der Diplomarbeit von Claudia Schindler entstand [11]. Im zweiten Teil der Analyse wurde diese Werkzeugvorlage bewertet. Dies erfolgte mit den gleichen Bewertungskriterien, wie für die aktuellen Werkzeuge. Am besten schnitt das Tool bei den aufgabenspezifischen Bewertungskriterien ab, da es speziell für den Einsatz im Informatikunterricht entwickelt wurde. Im nächsten Schritt wurde die Evaluation aus der Diplomarbeit von Frau Schindler ausgewertet. Dabei wurde festgestellt, dass der Quelltext zu Beginn sehr unverständlich ist. Außerdem wurde kritisiert, dass das Belegungsprotokoll und die Visualisierung nicht gleichzeitig angezeigt werden kann. Die Informationen über die aktuell existierenden Variablen sind jedoch für den Ablauf des Programms von großer Bedeutung. Weiterhin wurde festgestellt, dass die Metaphern in den Modulen zum Teil nicht relevant für den Alltag der SchülerInnen sind. Diese fördern jedoch ihre Motivation.

Im Anschluss an die Analyse wurden in der Entwurfsphase die Anforderungen an die Webanwendung definiert. Diese wurden anschließend in Muss-, Soll- und Kann-Kriterien unterteilt. Die Muss-Kriterien sind dabei so gewählt, dass die Anwendung lauffähig und einsatzbereit ist. Die Soll-Kriterien erweitern das Tool unter anderem um weitere Module und Steuerungsmöglichkeiten. Optionale Ideen wurden hingegen in den Kann-Kriterien festgehalten. Nach der Anforderungsanalyse wurden Szenarien und Inhalte für die einzelnen Module erstellt. Im Anschluss erfolgte die Erstellung von Mockups, die das Layout und das Design der Webanwendung darstellen. Der Prototyp wurde als Webanwendung mit den Sprachen HTML, CSS und Javascript implementiert. Durch den Verzicht auf serverseitige Techniken kann das Tool ohne Webserver im Browser aufgerufen

werden. Dies ermöglicht eine einfache Einbettung in E-Learning Plattformen als Lerninhalt.

Zum Schluss wurde eine Evaluation durchgeführt, bei der LehrerInnen, Studierende und auch SchülerInnen zu Ihrer Meinung bezüglich Educode befragt wurden. Dadurch konnte festgestellt werden, dass das Tool mit einer durchschnittlichen Bewertung von 1,83 einen guten Anklang findet und von den meisten auch im Unterricht eingesetzt werden würde. Zusätzlich wurden durch die Evaluation auch einige Verbesserungsvorschläge gesammelt, die zum Teil im Anschluss umgesetzt wurden.

Algorithmen zu verstehen und so zu formulieren, dass auch Maschinen sie richtig interpretieren, stellt SchülerInnen und LehrerInnen im Informatikunterricht auch in Zukunft vor eine große Aufgabe. Anwendungen wie Educode können die SchülerInnen bei diesem Schritt stark unterstützen. Im Besonderen sind Visualisierungen und alltagsnahe Beispiele dabei eine große Hilfe. Solche Tools sollten somit eine stärkere Einbindung in den Informatikunterricht erfahren. Dabei gibt es jedoch einige Dinge zu beachten. Die Anwendungen sollten sich nicht an kurz- sondern langfristigen Techniken orientieren. Außerdem ist es wichtig, dass auch SchülerInnen und LehrerInnen an der Entwicklung beteiligt werden, um das bestmögliche Ergebnis für alle zu erreichen. Vor allem die Lehrerperspektive wird oftmals nicht berücksichtigt. Die Tools werden nicht ausreichend beschrieben und auch der Einsatzzeitpunkt ist häufig unklar.

Das in dieser Arbeit entstandene Tool *Educode* kann zu diesem Zeitpunkt schon vollständig genutzt werden. Daher wird die Webanwendung voraussichtlich im August 2016 auf den sächsischen Bildungsserver geladen, um LehrerInnen und SchülerInnen aus Sachsen den Zugriff auf dieses Tool zu gewähren. Trotzdem gibt es noch einige Anpassungen, die Educode in Zukunft weiter ausbauen und verbessern können. Neue Module können implementiert werden, wie zum Beispiel für den Umgang mit Variablentypen. Aber auch das sortieren von verketteten Listen und die Einführung in Arrays wären zwei mögliche Module für die Modulgruppe *Datenstrukturen*. Dabei sollte jedoch darauf geachtet werden, dass das Tool derzeit für den Einstieg in die Programmiergrundlagen gedacht ist und die Module daher nicht zu komplex werden sollten. Da das dunkle Farbkonzept sehr ungewöhnlich ist, kann wahlweise eine hellere Variante angeboten werden. Auch die Anpassung der Schriftgröße ist eine gute Idee, um die Darstellung auf Bildschirmen mit unterschiedlichen Auflösungen zu verbessern. Zusätzlich können dem Benutzer noch mehr Freiheiten eingeräumt werden, in dem man ihm die Gestaltung der Oberfläche selbst überlässt. Der Nutzer kann so die benötigten Modulbereiche einblenden und an eine beliebige Position schieben. Dabei könnte die Anwendung auch besser an mobile Geräte angepasst werden.

Bei der Evaluation wurde von einigen Testpersonen die textlastige Einleitung kritisiert. Durch die Erstellung eines Popups, der die SchülerInnen zu Beginn eines Moduls Schritt für Schritt durch die Einleitung führt, könnte auf den Bereich *Einleitung* im Anschluss verzichtet werden. Außerdem können erfahrenere Personen diesen Teil überspringen. Ein wichtiger Punkt ist die Auswahl einer Programmiersprache im Editor, wodurch die LehrerInnen einen besseren Bezug für ihren persönlichen Unterricht herstellen können. Vor allem die Programmiersprachen *Java* und *Pascal* sind in diesem Zusammenhang zu nennen. Für fortgeschrittene SchülerInnen könnte der Editor weiterhin so angepasst werden, dass einzelne Abschnitte selbst eingetippt werden müssen. Um die Inhalte der Module noch verständlicher zu vermitteln, ist die Darstellung eines Flussdiagramms als

Alternative zur Visualisierung möglich. Neben den inhaltlichen Faktoren sollte jedoch auch die Motivation beachtet werden, die den Lernprozess der SchülerInnen maßgeblich verbessert. Dies kann zum Beispiel dadurch erreicht werden, dass abgeschlossene Module in der Navigation gekennzeichnet werden, so dass die SchülerInnen ihren Fortschritt besser nachvollziehen können. Gleichzeitig können dadurch auch die LehrerInnen den Lernfortschritt ihrer SchülerInnen überprüfen.

Literaturverzeichnis

[1] I. Diethelm, C. Dörge, A.-M. Mesaros, und M. Dünnebier (2011): Die Didaktische Rekonstruktion für den Informatikunterricht. In INFOS. Seiten 77–86. https://www.researchgate.net/profile/Ira_Diethelm/publication/221208850_Die_Didaktische_Rekonstruktion_fr_den_Informatikunterricht/links/02e7e5258414c49aff000000.pdf.

[2] N. Faltin (2002): Strukturiertes aktives Lernen von Algorithmen mit interaktiven Visualisierungen. Ph.D. thesis, Universität Oldenburg. http://oops.uni-oldenburg.de/269/1/299.pdf.

[3] S. Lehmann (2009): Visualisierung einfacher Algorithmen für den Informatikunterricht - Unterprogrammtechnik. Wissenschaftliche arbeit, Technische Universität Dresden, Fakultät Informatik, Inst. für Software- und Multimediatechnik, AG Didaktik der Informatik/Lehrerbildung.

[4] o. V. (1989 i. d. F. vom 05.02.2004). Einheitliche Prüfungsanforderungen - Informatik. Kultusministerium. http://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/1989/1989_12_01-EPA-Informatik.pdf.

[5] o. V. (2000). National Council of Teachers of Mathematics: Principles and Standards for School Mathematics. (NCTM). http://standards.nctm.org/.

[6] o. V. (2008). Ergonomie der Mensch-System-Interaktion - EN ISO 9241-110. Deutsches Institut für Normung e.v. http://xd-i.com/wp-content/uploads/2016/01/9241-110.pdf.

[7] o. V. (2011). Lehrplan Informatik - Gymnasium. Sächsisches Staatsministerium. http://www.schule.sachsen.de/lpdb/web/downloads/lp_gy_informatik_2011.pdf.

[8] o. V. (2014). Sachsen startet ins Klassenzimmer der Zukunft. Medienservice Sachsen. http://www.medienservice.sachsen.de/medien/news/190949.

[9] H. Puhlmann, T. Brinda, M. Fothe, S. Friedrich, B. Koerber, G. Röhner, und C. Schulte (2008). Grundsätze und Standards für die Informatik in der Schule. http://www.informatikstandards.de/docs/bildungsstandards_2008.pdf.

[10] G. Röhner, P. D. T. Brinda, V. Denke, D. L. Hellmig, T. Heußer, D. A. Pasternak, P. D. A. Schwill, und M. Seiffert (2016). Bildungsstandards Informatik für die Sekundarstufe II. http://www.informatikstandards.de/docs/Bildungsstandards_SII.pdf.

[11] C. Schindler (2008): Visualisierung einfacher Algorithmen für den Informatikunterricht. Diplomarbeit, Technische Universität Dresden, Fakultät Informatik, Inst. für Software- und Multimediatechnik, AG Didaktik der Informatik/Lehrerbildung.

[12] H. Witten (2003): Allgemeinbildender Informatikunterricht? Ein neuer Blick auf HW Heymanns Aufgaben allgemeinbildender Schulen. In INFOS. Seiten 59–75. http://cs.emis.de/LNI/Proceedings/Proceedings32/GI-Proceedings.32-7.pdf.

[13] A. Zendler und C. Spannagel (2006): Zentrale Konzepte im Informatikunterricht: eine empirische Grundlegung. *Notes on Educational Informatics - Section A: Concepts and Techniques*, Vol. 2 (Nr. 1):Seiten 1–21. http://www.ph-ludwigsburg.de/fileadmin/subsites/2e-imix-t-01/user_files/Journal_NEI_-_PDFs_fuer_Webauftritt/Section_A/Volume_2_No_1_2006/NEI_Section_A_Vol._2_No._1_2006_p._01-21_-_Zendler_Spannagel_-_Zentrale_Konzepte_01.pdf.

Verzeichnis der Webadressen

[14] Animate.css. https://daneden.github.io/animate.css/.

[15] Bootstrap. http://getbootstrap.com/.

[16] Bootstrap Slider. http://www.eyecon.ro/bootstrap-slider.

[17] Bootstrap Tour. http://bootstraptour.com/.

[18] Code. https://code.org/. (Zugriff: 2.3.2016).

[19] Code Prettify. https://github.com/google/code-prettify.

[20] Codecademy. https://www.codecademy.com/. (Zugriff: 2.3.2016).

[21] Coding Dojo. http://www.codingdojo.com/. (Zugriff: 2.3.2016).

[22] Coding Dojo Algorithm Platform. http://algorithm.codingdojo.com/. (Zugriff: 2.3.2016).

[23] Font Awesome. http://fontawesome.io/.

[24] jQuery. https://jquery.com/.

[25] jQuery Custom Scrollbar. http://manos.malihu.gr/jquery-custom-content-scroller/.

[26] jQuery Sortable. https://johnny.github.io/jquery-sortable/.

[27] Khanacademy. https://www.khanacademy.org/. (Zugriff: 3.3.2016).

[28] Learn C. http://www.learn-c.org/. (Zugriff: 2.3.2016).

[29] Learn C-Sharp. http://www.learncs.org/. (Zugriff: 2.3.2016).

[30] Learn Java. http://www.learnjavaonline.org/. (Zugriff: 2.3.2016).

[31] Learn JavaScript. http://www.learn-js.org/. (Zugriff: 2.3.2016).

[32] Learn PHP. http://www.learn-php.org/. (Zugriff: 2.3.2016).

[33] Learn Python. http://www.learnpython.org/. (Zugriff: 2.3.2016).

[34] Learn Shell. http://www.learnshell.org/. (Zugriff: 2.3.2016).

[35] Learneroo. https://www.learneroo.com/. (Zugriff: 3.3.2016).

[36] Raphael.js. http://dmitrybaranovskiy.github.io/raphael/.

[37] TheCodePlayer. http://thecodeplayer.com/. (Zugriff: 3.3.2016).

[38] Udacity. https://www.udacity.com/. (Zugriff: 3.3.2016).

A Bewertung aktueller Werkzeuge

A.1 Codecademy

Kriterium	**Punkte** (0 - 5)	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (60%)		
Registrierung	4	· E-Mail / Google / Facebook · Kostenlos + Premium Modell · Extra Lehrermodell möglich
Anspruch	2	· Das Lesen von Code wird vorrausgesetzt
Anwendungsgebiet	3	· speziell · Ruby on Rails, AngularJS, Command Line, SQL, Java, Git, HTML, CSS, jQuery, PHP, Python, Ruby
Aufgabenangemessenheit (95%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	

Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Editor mit üblichen Shortcuts (Kopieren, Ausschneiden, Einfügen, . . .) · Sonst keine Shortcuts möglich · Keine Makros, aber auch nicht notwendig
Speichern von ausgefüllten Daten	5	· Eingegebene Lösungen werden dauerhaft gespeichert, können aber zurückgesetzt werden
Selbstbeschreibungsfähigkeit (88%)		
Erklärung von Schritten	5	· jede Aufgabe wird verständlich erklärt · Aktuelle Aufgaben sind immer präsent
Eindeutige Bezeichnungen	5	· Die Bezeichnungen von Schaltflächen sind sehr verständlich · Texte sind ebenfalls verständlich geschrieben
Adaptive Hilfe	3	· Code wird bei Verständnisproblemen vor allem für Anfänger nicht erklärt · Teilweise Tooltips vorhanden
Rückmeldung zum Bearbeitungsstand	4	· Aktuelle Position im Tutorial wird dargestellt · Wenn Code ausgeführt wird, wird zwar Ladeanimation angezeigt, aber nicht wie lange · Bearbeitungsstand der einzelnen Tutorials werden angezeigt
Überblickinformationen	5	· Übersicht der abgeschlossenen Aufgaben · Gute Übersicht über begonnene und abgeschlossene Tutorials
Lernförderlichkeit (90%)		
Unterstützung des Nutzers	4	
Konsistenz	5	
Steuerbarkeit (67%)		
Steuerung der Richtung durch Nutzer	4	· Nutzer kann zu Schritten springen, die er bereits erledigt hat · Auf spätere Schritte kann nicht zugegriffen werden
Steuerung der Geschwindigkeit durch Nutzer	3	· Nutzer kann allein bestimmen, wann er zum nächsten Aufgabe übergeht · Code-Schritte können nicht gesteuert werden
Schritte können rückgängig gemacht werden	3	· Schritte können rückgängig gemacht werden, dies ist aber nur über die Hilfe möglich (versteckt)
Erwartungskonformität (100%)		

Einheitliche Dialogverhalten	5	· Tutorials sind alle nach gleichem Schema aufgebaut
Einheitliche Darstellung gleicher Bedienelemente	5	· Bedienelemente sind immer als solche erkennbar
Individualisierbarkeit (50%)		
Anpassung vom Umfang der Erläuterungen	2	· Anpassung nicht möglich
Anpassung der Darstellung	3	· Größe der Ansichtsfelder können verändert werden
Fehlertoleranz (73%)		
Abfangen von Fehlern	5	· Fehler werden abgefangen und dem Nutzer angezeigt
Korrekturhinweise	3	· Typische Fehler werden dem Nutzer erläutert · Untypische Fehler werden nur durch Konsolenausgabe dem Nutzer präsentiert, dies erfordert zu viel Wissen
Hilfe	3	· Forum · Im Abo-Modell: Personen die dir persönlich helfen · Keine Hilfeseite

Tabelle A.1: Bewertungen - Codecademy [20]

A.2 Code.org

Kriterium	**Punkte** (0 - 5)	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (80%)		
Registrierung	4	· als LehrerIn oder SchülerIn · E-Mail / Google / Facebook / Microsoft
Anspruch	4	· Kurse ab 4 Jahren bis zu jedem Alter · Code wird nur auf Anfrage teilweise angezeigt
Anwendungsgebiet	4	· allgemein · Code wird in JavaScript dargestellt
Aufgabenangemessenheit (95%)		

Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	
Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Shortcuts für Kopieren, Ausschneiden, Einfügen sind vorhanden
Speichern von ausgefüllten Daten	5	· Eingegebene Lösungen werden dauerhaft gespeichert, können aber zurückgesetzt werden
Selbstbeschreibungsfähigkeit (92%)		
Erklärung von Schritten	5	· Schritte werden sehr einfach beschrieben
Eindeutige Bezeichnungen	5	· Bezeichnungen sind eindeutig
Adaptive Hilfe	5	· Bei falschen Eingaben wird Hilfe angeboten
Rückmeldung zum Bearbeitungsstand	4	· Aktuelle Position im Tutorial wird angezeigt · Ausführen von Programmen dauert etwas, dabei gibt es keine Rückmeldung
Überblickinformationen	4	· Übersicht über aktuelle und abgeschlossene Aufgaben vorhanden · Keine Übersicht über aktuellen Stand
Lernförderlichkeit (100%)		
Unterstützung des Nutzers	5	· Nutzer wird durch Videos, Animationen und Bildern unterstützt
Konsistenz	5	
Steuerbarkeit (73%)		
Steuerung der Richtung durch Nutzer	3	· Nutzer kann zu jeder beliebigen Aufgabe springen · Schritte innerhalb einer Aufgabe können nicht gesteuert werden
Steuerung der Geschwindigkeit durch Nutzer	3	· Geschwindigkeit der Ausführung kann nur in wenigen Tutorials gesteuert werden
Schritte können rückgängig gemacht werden	5	· Aufgaben können neugestartet werden
Erwartungskonformität (80%)		
Einheitliche Dialogverhalten	5	

Einheitliche Darstellung gleicher Bedienelemente	3	· Bedienelemente sind teilweise den einzelnen Tutorials angepasst, was zu Verständnisproblemen führen kann
Individualisierbarkeit (70%)		
Anpassung vom Umfang der Erläuterungen	4	· Hinweise können eingeblendet werden · Genauere Erläuterungen für Fortgeschrittene nicht vorhanden
Anpassung der Darstellung	3	· Größe der Ansichtsfelder können verändert werden
Fehlertoleranz (80%)		
Abfangen von Fehlern	5	· Fehler werden abgefangen und dem Nutzer angezeigt
Korrekturhinweise	5	· Fehler werden dem Nutzer angezeigt und erläutert · Wenn Hinweis nichts bringt, kann mehr Hilfe angefordert werden
Hilfe	2	· Hilfe nur bei fehlerhaften Eingaben

Tabelle A.2: Bewertungen - Code.org [18]

A.3 Learn X

Kriterium	**Punkte (0 - 5)**	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (73%)		
Registrierung	5	· Keine Registrierung nötig
Anspruch	5	· Umgang mit Code wird vorrausgesetzt, da er teilweise nicht erläutert wird
Anwendungsgebiet	3	· speziell · Python, Java, C, JavaScript, PHP, Shell, C#
Aufgabenangemessenheit (75%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	

Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Editor mit üblichen Shortcuts (Kopieren, Ausschneiden, Einfügen, . . .)
Speichern von ausgefüllten Daten	1	· Angegebene Daten werden nicht gespeichert
Selbstbeschreibungsfähigkeit (68%)		
Erklärung von Schritten	4	· Einzelne Schritte werden gut erläutert, jedoch nicht direkt am Beispiel
Eindeutige Bezeichnungen	5	
Adaptive Hilfe	3	· Es wird keine adaptive Hilfe angeboten
Rückmeldung zum Bearbeitungsstand	3	· Aktuelle Position im Tutorial wird angezeigt · Das Ausführen des Codes dauert sehr lange, keine Rückmeldung
Überblickinformationen	2	· Kein Überblick über absolvierte Tutorials
Lernförderlichkeit (80%)		
Unterstützung des Nutzers	3	· Komplizierter Umgang
Konsistenz	5	
Steuerbarkeit (80%)		
Steuerung der Richtung durch Nutzer	5	· Nutzer kann zu jeder beliebigen Aufgabe springen
Steuerung der Geschwindigkeit durch Nutzer	2	· Nutzer wird nach Beendigung automatisch zum nächsten Teil weitergeleitet
Schritte können rückgängig gemacht werden	5	· Aufgaben können zurückgesetzt werden
Erwartungskonformität (80%)		
Einheitliche Dialogverhalten	3	· Tutorials sind gleich aufgebaut, jedoch etwas unübersichtlich
Einheitliche Darstellung gleicher Bedienelemente	5	· Bedienelemente sind einheitlich
Individualisierbarkeit (40%)		
Anpassung vom Umfang der Erläuterungen	2	· Anpassung nicht möglich

Anpassung der Darstellung	2	· Anpassung nicht möglich
Fehlertoleranz (7%)		
Abfangen von Fehlern	0	· Auf fehlerhafte Eingaben wird nicht hingewiesen
Korrekturhinweise	1	· Keine Korrekturhinweise, bis auf Shell-Ausgabe
Hilfe	0	· Keine Hilfe vorhanden

Tabelle A.3: Bewertungen - Learn X [28, 29, 31, 30, 32, 33, 34]

A.4 Coding Dojo

Kriterium	**Punkte (0 - 5)**	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (53%)		
Registrierung	3	· E-Mail ohne Passwort, dadurch kann sich jeder mit deinem Account anmelden
Anspruch	1	· Tutorial beginnt mit einfachen Aufgaben, die jedoch nicht erläutert werden · werden dadurch später sehr schwer für Anfänger
Anwendungsgebiet	4	· alle Tutorials und Aufgaben basieren auf JavaScript · allgemeines Verständnis wird vermittelt
Aufgabenangemessenheit (95%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	
Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Editor mit üblichen Shortcuts (Kopieren, Ausschneiden, Einfügen, . . .)
Speichern von ausgefüllten Daten	5	· Eingegebene Lösungen werden dauerhaft gespeichert, können aber zurückgesetzt werden

Selbstbeschreibungsfähigkeit (88%)		
Erklärung von Schritten	4	· Schritte werden beschrieben, Lösungsweg jedoch nicht
Eindeutige Bezeichnungen	5	
Adaptive Hilfe	3	· Es wird keine adaptive Hilfe angeboten
Rückmeldung zum Bearbeitungsstand	5	· Über Bearbeitungsstand wird sehr gut informiert
Überblickinformationen	5	· Gut Übersicht über absolvierte und kommende Aufgaben
Lernförderlichkeit (90%)		
Unterstützung des Nutzers	4	
Konsistenz	5	
Steuerbarkeit (80%)		
Steuerung der Richtung durch Nutzer	4	· Nutzer kann zu Schritten springen, die er bereits erledigt hat · Auf spätere Schritte kann nicht zugegriffen werden
Steuerung der Geschwindigkeit durch Nutzer	3	· Nutzer kann allein bestimmen, wann er zum nächsten Aufgabe übergeht · Code-Schritte können nicht gesteuert werden
Schritte können rückgängig gemacht werden	5	· Aufgaben können zurückgesetzt werden
Erwartungskonformität (100%)		
Einheitliche Dialogverhalten	5	
Einheitliche Darstellung gleicher Bedienelemente	5	
Individualisierbarkeit (50%)		
Anpassung vom Umfang der Erläuterungen	2	· Anpassung nicht möglich
Anpassung der Darstellung	3	· Anpassung nicht möglich
Fehlertoleranz (73%)		
Abfangen von Fehlern	5	· Fehler werden abgefangen und dem Nutzer angezeigt

Korrekturhinweise	3	· Auf Syntaxfehler wird hingewiesen, jedoch nicht gezeigt an welcher Stelle · Video mit Lösungsweg, wenn eingegebene Lösung falsch ist
Hilfe	3	· Hilfe nur durch Video-Tutorial

Tabelle A.4: Bewertungen - Coding Dojo [22]

A.5 TheCodePlayer

Kriterium	**Punkte (0 - 5)**	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (53%)		
Registrierung	5	· Keine Registrierung
Anspruch	1	· Sprachen müssen bereits gekonnt werden, um sie zu verstehen
Anwendungsgebiet	2	· direkte Tutorials für bestimmte Funktionalitäten · CSS, HTML, Javascript
Aufgabenangemessenheit (65%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	3	· Text kann während des Code Playings geändert werden
Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Editor mit üblichen Shortcuts (Kopieren, Ausschneiden, Einfügen, . . .)
Speichern von ausgefüllten Daten	1	· Angegebene Daten werden nicht gespeichert
Selbstbeschreibungsfähigkeit (64%)		
Erklärung von Schritten	1	· Schritte werden nicht erklärt
Eindeutige Bezeichnungen	5	
Adaptive Hilfe	3	· Es wird keine adaptive Hilfe angeboten

Rückmeldung zum Bearbeitungsstand	3	· Position wird angegeben
Überblickinformationen	4	· Bereits abgespielte Programme werden nicht gekennzeichnet
Lernförderlichkeit (90%)		
Unterstützung des Nutzers	4	
Konsistenz	5	
Steuerbarkeit (93%)		
Steuerung der Richtung durch Nutzer	4	· Nutzer kann zu beliebiger Position springen · Rückwärtslauf nicht möglich
Steuerung der Geschwindigkeit durch Nutzer	5	· Geschwindigkeit kann in vier verschiedenen Stufen angegeben werden
Schritte können rückgängig gemacht werden	5	
Erwartungskonformität (60%)		
Einheitliche Dialogverhalten	3	· Es öffnen sich teilweise verschachtelte Dialoge, die zur Verwirrung führen können
Einheitliche Darstellung gleicher Bedienelemente	3	· Schaltflächen werden auf unterschiedliche Art dargestellt
Individualisierbarkeit (60%)		
Anpassung vom Umfang der Erläuterungen	2	· Anpassung nicht möglich
Anpassung der Darstellung	4	· Es kann ausgewählt werden, welche Module angezeigt werden sollen
Fehlertoleranz (0%)		
Abfangen von Fehlern	0	· Auf fehlerhafte Eingaben wird nicht hingewiesen
Korrekturhinweise	0	· Keine Korrekturhinweise
Hilfe	0	· Keine Hilfe vorhanden

Tabelle A.5: Bewertungen - TheCodePlayer [37]

A.6 Khanacademy

Kriterium	Punkte (0 - 5)	Anmerkungen
Aufgabenspezifische Bewertungskriterien (93%)		
Registrierung	5	· E-Mail, Facebook, Google · Als LehrerIn oder Elternteil · Registrierung nicht zwingend notwendig
Anspruch	5	· Sehr einfach mithilfe von Videos erklärt
Anwendungsgebiet	4	· Allmögliche Themengebiete · Computing: Computer Science (Algorithmen), Hour of Code (JavaScript, Allgemein), Web Programming (Javascript, CSS, HTML)
Aufgabenangemessenheit (100%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	
Verwendung von Makros, Voreinstellungen und Shortcuts	5	· Editor: Shortcuts für Kopieren, Ausschneiden, Einfügen sind vorhanden · Shortcuts für rückgängig
Speichern von ausgefüllten Daten	5	· Eingegebene Lösungen werden dauerhaft gespeichert, können aber zurückgesetzt werden
Selbstbeschreibungsfähigkeit (88%)		
Erklärung von Schritten	5	· Schritte werden durch Videos und Texte verständlich und umfangreich erklärt
Eindeutige Bezeichnungen	5	
Adaptive Hilfe	4	
Rückmeldung zum Bearbeitungsstand	5	· Aktueller Bearbeitungsstand wird sehr gut angezeigt

Überblickinformationen	3	· Überblick innerhalb einzelner Sektionen gut, über alle Sektionen nicht vorhanden
Lernförderlichkeit (100%)		
Unterstützung des Nutzers	5	
Konsistenz	5	
Steuerbarkeit (100%)		
Steuerung der Richtung durch Nutzer	5	· Nutzer kann zu jeder beliebigen Aufgabe springen · Videosteuerung
Steuerung der Geschwindigkeit durch Nutzer	5	· Nutzer kann Geschwindigkeit selbst bestimmen
Schritte können rückgängig gemacht werden	5	
Erwartungskonformität (90%)		
Einheitliche Dialogverhalten	5	
Einheitliche Darstellung gleicher Bedienelemente	4	· Schaltflächen werden teilweise unterschiedlich dargestellt
Individualisierbarkeit (90%)		
Anpassung vom Umfang der Erläuterungen	5	· Zusätzliche Informationen können angezeigt werden
Anpassung der Darstellung	4	· Menü kann ausgeblendet werden · Ansichts- und Codefenster können vergrößert und verkleinert werden
Fehlertoleranz (100%)		
Abfangen von Fehlern	5	· Fehler werden abgefangen und dem Nutzer angezeigt
Korrekturhinweise	5	· Gute und verständliche Korrekturhinweise vorhanden · Tipps auch durch Nutzer
Hilfe	5	· Dokumentation, Probleme können gemeldet werden, andere Nutzer können gefragt werden

Tabelle A.6: Bewertungen - Khanacademy [27]

A.7 Udacity

Kriterium	**Punkte** (0 - 5)	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (67%)		
Registrierung	5	· E-Mail, Facebook, Google · Registrierung für freie Kurse nicht zwingend notwendig
Anspruch	3	· Unterschiedliche Kurse mit verschiedenen Ansprüchen
Anwendungsgebiet	2	· speziell · Viele Programmiersprachen · Nur Videos
Aufgabenangemessenheit (100%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	
Verwendung von Makros, Voreinstellungen und Shortcuts	-	
Speichern von ausgefüllten Daten	5	· Eingegebene Lösungen werden dauerhaft gespeichert, können aber zurückgesetzt werden
Selbstbeschreibungsfähigkeit (84%)		
Erklärung von Schritten	3	· Aufgabenstellung nur in Video-Form, muss das ganze Video schauen um Aufgabenstellung zu wiederholen
Eindeutige Bezeichnungen	5	
Adaptive Hilfe	3	· Es wird keine adaptive Hilfe angeboten
Rückmeldung zum Bearbeitungsstand	5	· Bearbeitungsstand wird gut angezeigt
Überblickinformationen	5	· Guter Überblick über Aufgaben
Lernförderlichkeit (80%)		
Unterstützung des Nutzers	3	· Nur Handschrift bei Aufgaben, manchmal schwierig zu lesen

Konsistenz	5	
Steuerbarkeit (80%)		
Steuerung der Richtung durch Nutzer	4	· Nutzer kann zu beliebiger Position springen
Steuerung der Geschwindigkeit durch Nutzer	5	· Nutzer kann Geschwindigkeit vorgeben
Schritte können rückgängig gemacht werden	3	· Lösungen können korrigiert werden, aber nicht zurückgesetzt werden
Erwartungskonformität (100%)		
Einheitliche Dialogverhalten	5	
Einheitliche Darstellung gleicher Bedienelemente	5	
Individualisierbarkeit (50%)		
Anpassung vom Umfang der Erläuterungen	2	· Anpassung nicht möglich
Anpassung der Darstellung	3	· Videos können in Vollbild dargestell werden
Fehlertoleranz (73%)		
Abfangen von Fehlern	5	· Auf Fehler wird hingewiesen
Korrekturhinweise	2	· Es gibt keine Korrekturhinweise bis auf die Anzeige der Lösung
Hilfe	4	· Diskussionsforum · Anfragen an den Support möglich

Tabelle A.7: Bewertungen - Udacity [38]

A.8 Learneroo

Kriterium	**Punkte (0 - 5)**	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (73%)		

Registrierung	3	· E-Mail, Facebook · Erklärungen auch ohne Registrierung · Challenges nur mit Registrierung · Kostenpflichtige Module
Anspruch	4	· Unterschiedliche Voraussetzungen für Kurse, sind nicht gekennzeichnet · teilweise wird Code-Verständnis vorausgesetzt
Anwendungsgebiet	4	· Java, Web Development (HTML, Ruby), Algorithmen · Programmiersprache kann teilweise selbst ausgewählt werden
Aufgabenangemessenheit (90%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	5	
Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Shortcuts für Kopieren, Ausschneiden, Einfügen sind vorhanden
Speichern von ausgefüllten Daten	4	· Eingegebene Lösungen werden dauerhaft gespeichert, können nicht zurückgesetzt sondern nur selbst gelöscht werden
Selbstbeschreibungsfähigkeit (96%)		
Erklärung von Schritten	5	· Schritte werden in Textform eindeutig beschrieben
Eindeutige Bezeichnungen	5	· Bezeichnungen sind eindeutig
Adaptive Hilfe	5	· Hinweise können eingeblendet werden
Rückmeldung zum Bearbeitungsstand	5	· Bereits angesehene und bereits abgeschlossene Aufgaben werden gekennzeichnet · Bearbeitungsstand innerhalb eines Moduls wird dargestellt (Anfänger, Fortgeschritten, etc.)
Überblickinformationen	4	· Überblick über Module, die im Moment bearbeitet werden ist gegeben
Lernförderlichkeit (100%)		
Unterstützung des Nutzers	5	· Walk-through vom Code-Editor gegeben · Hinweise bei einzelnen Schritten · Quick Reference vorhanden

Konsistenz	5	
Steuerbarkeit (80%)		
Steuerung der Richtung durch Nutzer	4	· Nutzer kann zu beliebigen Schritten springen
Steuerung der Geschwindigkeit durch Nutzer	3	· Geschwindigkeit wird durch Nutzer vorgegeben · Beim Ausführen des Codes können einzelne Schritte nicht gesteuert werden
Schritte können rückgängig gemacht werden	5	· Lösungen können zurückgesetzt werden · Vorheriger Code nach Änderung kann erneut geladen werden
Erwartungskonformität (100%)		
Einheitliche Dialogverhalten	5	
Einheitliche Darstellung gleicher Bedienelemente	5	
Individualisierbarkeit (100%)		
Anpassung vom Umfang der Erläuterungen	5	· Quick Reference · Hinweise können eingeblendet werden
Anpassung der Darstellung	5	· Schriftgröße kann in Editor angepasst werden · Breite/Höhe vom Editor kann angepasst werden · Beschreibung und Editor kann in 1- oder 2-Spalten Layout angezeigt werden
Fehlertoleranz (80%)		
Abfangen von Fehlern	5	· Fehler werden abgefangen und angezeigt
Korrekturhinweise	3	· Korrekturhinweise nur durch Shell-Ausgaben, für Anfänger unverständlich
Hilfe	4	· Quick Reference · Externes Debug Visualisierungs Tool · Kommentarfunktion

Tabelle A.8: Bewertungen - Learneroo [35]

B Bewertung der Werkzeugvorlage

Kriterium	**Punkte** (0 - 5)	**Anmerkungen**
Aufgabenspezifische Bewertungskriterien (100%)		
Registrierung	5	· keine Registrierung
Anspruch	5	· Beschreibungen und Erklärungen durch Hover · Code wird schrittweise visualisiert
Anwendungsgebiet	5	· Allgemein: Pseudo Code · Grundlegene Algorithmen der Programmierung
Aufgabenangemessenheit (80%)		
Fernhaltung von internen Aufgaben	5	
Aufgabenorientierte Informationseingabe und -ausgabe	3	· Informationsausgabe im Protokoll zu Beginn etwas schwer verständlich · Eingaben werden nicht im Code angezeigt, nur im Protokoll und in der Visualisierung
Verwendung von Makros, Voreinstellungen und Shortcuts	4	· Nicht vorhanden
Speichern von ausgefüllten Daten	4	· Eingaben werden nicht gespeichert, aber in dem Kontext auch nicht notwendig

Selbstbeschreibungsfähigkeit (96%)		
Erklärung von Schritten	5	· Einzelne Schritte werden durch Tooltip erklärt · zusätzliche Visualisierung
Eindeutige Bezeichnungen	5	
Adaptive Hilfe	5	· Tooltips können aktiviert werden
Rückmeldung zum Bearbeitungsstand	4	· Bearbeitungsstand von Modulen wird nicht angezeigt · Position innerhalb eines Moduls wird gut angezeigt
Überblickinformationen	5	· Übersicht über Module · Gute Übersicht aktuelle Position innerhalb eines Moduls mit Visualisierung
Lernförderlichkeit (80%)		
Unterstützung des Nutzers	3	· Teilweise unverständliche Navigation
Konsistenz	5	
Steuerbarkeit (93%)		
Steuerung der Richtung durch Nutzer	4	· Nutzer kann in Einzelschritte vorwärs aber nicht rückwärts gehen · Module können beliebig gestarte und beendet werden
Steuerung der Geschwindigkeit durch Nutzer	5	· Nutzer kann Geschwindigkeit der Code-Ausführung in 9 Stufen regeln · Pause möglich
Schritte können rückgängig gemacht werden	5	· Modul kann zurückgesetzt werden
Erwartungskonformität (80%)		
Einheitliche Dialogverhalten	3	· Manchmal unverständlich wann sich neues Fenster öffnet und wann Information im gleichen Fenster geladen wird · Überfüllung bei Schleifen wird nicht visualisiert
Einheitliche Darstellung gleicher Bedienelemente	5	
Individualisierbarkeit (70%)		
Anpassung vom Umfang der Erläuterungen	4	· Tooltips können aktiviert werden · Genauere Beschreibungen zu Ablauf in Modulen fehlen

Anpassung der Darstellung	3	· Anpassung nicht möglich
Fehlertoleranz (80%)		
Abfangen von Fehlern	5	· Fehler werden abgefangen
Korrekturhinweise	4	· Keine Korrekturhinweise, aber nicht zwingend erforderlich
Hilfe	3	· Hilfe nur durch Tooltips

Tabelle B.1: Bewertungen - Werkzeugvorlage [11]

C Evaluation

C.1 Fragebogen

1. Erscheinungsbild

Wie beurteilen Sie das Erscheinungsbild der folgenden Komponenten?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht
Gesamte Anwendung	◯	◯	◯	◯	◯	◯	◯
Navigation	◯	◯	◯	◯	◯	◯	◯
Hilfe-Tour	◯	◯	◯	◯	◯	◯	◯
Aufteilung des Moduls in Bereiche	◯	◯	◯	◯	◯	◯	◯
Bereich "Einleitung"	◯	◯	◯	◯	◯	◯	◯
Bereich "Aufgabe"	◯	◯	◯	◯	◯	◯	◯
Bereich "Editor"	◯	◯	◯	◯	◯	◯	◯
Fehlermeldungen bei falscher Reihenfolge	◯	◯	◯	◯	◯	◯	◯
Bereich "Steuerung"	◯	◯	◯	◯	◯	◯	◯
Bereich "Visualisierung"	◯	◯	◯	◯	◯	◯	◯
Bereich "Variablen"	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zum Erscheinungsbild von EduCode

2. Funktionen

Wie beurteilen Sie die folgenden <u>Funktionen</u> der Anwendung?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht	Wie wichtig ist Ihnen diese Funktion? --	-	+	++
Hilfe-Tour	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Bereich "Einleitung"	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Ein- / Ausblenden der Bereiche für Einleitung und Aufgabe	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Sortierung der Codeblöcke durch den Benutzer	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Auswahlfelder für Variablenwerte im Editor	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Hinweis-Tooltip für Codeblöcke im Editor	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Fehlermeldungen	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Automatische Wiedergabe des Programms	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Manuelle Wiedergabe des Programms	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Steuerung der Geschwindigkeit	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Visualisierung	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Variablen-Tabelle	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zu den oben genannten Funktionen von EduCode

3. Modul: Variablen

Wie beurteilen Sie die Metapher mit dem Regal?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht
Regal	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zu den Modulen in der Gruppe Variablen

4. Modul: Verzweigungen

Wie beurteilen Sie die Metapher mit dem Akkustand des Smartphones?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht
Akkustand	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zu den Modulen in der Gruppe Verzweigungen

5. Modul: Schleifen

Wie beurteilen Sie die Metapher mit dem Wasserglas?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht
Wasserglas	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zu den Modulen in der Gruppe Schleifen

6. Modul: Unterprogramme

Wie beurteilen Sie die Metapher mit dem Flächenberechnung?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht
Flächenberechnung	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zu den Modulen in der Gruppe Unterprogramme

7. Modul: Datenstrukturen

Wie beurteilen Sie die Metaphern mit dem Schuh und der Wiedergabeliste?

	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Ungenügend	Weiß nicht
Schuh	◯	◯	◯	◯	◯	◯	◯
Wiedergabeliste	◯	◯	◯	◯	◯	◯	◯

Anmerkungen zu den Modulen in der Gruppe Datenstrukturen

8. Allgemein

Finden Sie die Umsetzung als Webanwendung sinnvoll?

◯ ja

◯ nein

Würden Sie die Anwendung im Unterricht einsetzen?

◯ ja

◯ nein

Welche Verbesserungsvorschläge haben Sie für die Anwendung?

Zusätzliche Funktionen, weitere Module, Änderungen, Probleme, ...

Weitere Anmerkungen, die im Fragebogen nicht angesprochen wurden.

8. Allgemein

Warum würden Sie die Anwendung nicht im Unterricht einsetzen?

9. Persönliche Informationen

Welche Tätigkeit führen Sie aus?

Bitte wählen...

Wie alt sind Sie?

- ◯ < 18
- ◯ 18 - 25
- ◯ 26 - 35
- ◯ 36 - 45
- ◯ 46 - 60
- ◯ > 60
- ◯ Keine Angabe

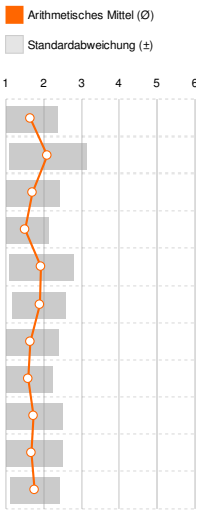
Antwortmöglichkeiten zu „Welche Tätigkeit führen Sie aus?“

- Lehrerin (Informatik)
- Lehrerin (Sonstiges)
- StudentIn (Lehramt Informatik)
- StudentIn (Medien-/Informatik)
- StudentIn (Sonstiges)
- SchülerIn
- Sonstiges

C.2 Ergebnisse

1. Wie beurteilen Sie das Erscheinungsbild der folgenden Komponenten?

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Gesamte Anwendung	20x	47,62	18x	42,86	3x	7,14	1x	2,38	-	-	-	-	-	1,64	0,73
Navigation	13x	30,95	17x	40,48	9x	21,43	1x	2,38	2x	4,76	-	-	-	2,10	1,03
Hilfe-Tour	17x	41,46	20x	48,78	3x	7,32	1x	2,44	-	-	-	-	-	1,71	0,72
Aufteilung des Moduls in ...	24x	57,14	15x	35,71	3x	7,14	-	-	-	-	-	-	-	1,50	0,63
Bereich "Einleitung"	15x	36,59	15x	36,59	10x	24,39	1x	2,44	-	-	-	-	-	1,93	0,85
Bereich "Aufgabe"	13x	31,71	20x	48,78	8x	19,51	-	-	-	-	-	-	-	1,88	0,71
Bereich "Editor"	21x	52,50	14x	35,00	4x	10,00	1x	2,50	-	-	-	-	-	1,63	0,77
Fehlermeldungen bei fal...	20x	47,62	18x	42,86	3x	7,14	-	-	-	-	-	-	1x	1,59	0,63
Bereich "Steuerung"	17x	40,48	19x	45,24	4x	9,52	1x	2,38	-	-	-	-	1x	1,73	0,74
Bereich "Visualisierung"	20x	48,78	15x	36,59	5x	12,20	1x	2,44	-	-	-	-	-	1,68	0,79
Bereich "Variablen"	15x	36,59	21x	51,22	5x	12,20	-	-	-	-	-	-	-	1,76	0,66



2. Anmerkungen zum Erscheinungsbild von EduCode

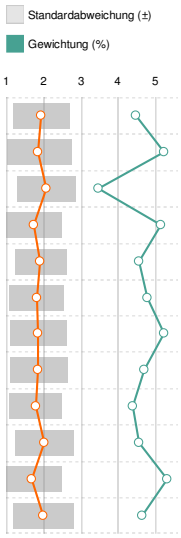
- Ich vermisse einen Button "zur nächsten Aufgabe", es ist ungewöhnlich, dass nach erfolgreicher Lösung einer Aufgabe nicht zur nächsten Aufgabe verwiesen wird.
  Der Einleitungsbereich ist sehr textlastig und erschlägt dadurch etwas. Warum nicht Einleitungsfenster schalten?
  Zu Beginn der jeweiligen Lektion erscheint ein Fenster in der Mitte des Bildschirms, welches in mehreren Seiten das Lernobjekt erklärt.
- Sehr gutes Design
- Einleitung teilweise mit sehr viel Text verbunden (evtl. an der ein oder anderen Stelle Möglichkeit zur einblendung von mehr Text vorsehen)
- Menschen können ungeduldig sein..
  ich habe das hilfe Fenster weggeklickt
  Das Pulldown Menü habe ich leider erst spät entdeckt , da etwas unscheinbar (könnte größer und besser ausgeschildert sein ;))
- Habe den Bereich für die Variablen zwar wahrgenommen, durch den visuellen Bereich aber nicht unbedingt darauf geachtet.
- Bei kleinem Fenster ist die Visualisierung abgeschnitten. Wäre schön wenn nicht, da dann die Anwendung auch auf mobilen Geräten zum Üben gnutzt werden könnte.
- - für ältere Lehrer könnte die Lesbarkeit verbessert werden: Kontrast, zu kleine Schrift...
  - für "normale" Schüler fast schon zu viel Text auf einmal, wollen "schnell durch" ...
  ==> evt. "zerhacken" und nacheinander anzeigen lassen
- Das Auge benötigt noch ein Feld zur Erklärung, was dort zu finden ist. Eine fertige Lösung mit der man etwas einer Klasse sofort präsentieren kann ohne erst alles in die richtige Reihenfolge zu bringen wäre hilfreich.
  Das die Aufgabe gelöst werden muss durch Drag und Drop muss man erst mal rausfinden ohne die gesamte Einführung durchzugehen.
  Vielleicht noch eine kleine Anmerkung dazu im Aufgabenteil (der vllt besser wäre über/unter den Editor zu setzen).
- Ich habe die erste Aufgabe gemacht und erst hinterher gesucht/gefunden,wo oben das blaugrüne Feld zur Wahl der Aufgaben ist.
  Vielleicht sollte man den Startbildschirm optimieren, dass man das Auswahlmenü gleich findet.

  Bildschirmfarbe ist gewöhnungsbedürftig, aber durchaus gut lesbar.
- Einleitungstexte zu lang oder breiteres Fenster, dafür kann Visualisierung kleiner sein
  Aufgabentexte farblich vom Rest abheben. Funktionen von Steuer unnötig, Start reicht
- Gute Idee, übersichtlicher Aufbau, intuitive Bedienung.
  Grundsätzlich besteht m. E. aber die Gefahr, dass die Visualisierung gleich gesetzt werden könnte mit Bildschirmausgaben, obwohl natürlich keinerlei Ausgabeanweisungen erfolgen. Vielleicht sollte darauf noch stärker hingewiesen werden.
- In den Aufgaben sollte klar gestellt werden das die Elemente sortiert werden sollen.

  Beim Besuch der Seite sollte es evtl. nicht sofort mit einem Bereich los gehen sondern eine Übersicht der möglichen Bereiche erscheinen damit eine Auswahl getroffen werden kann.
- Der Bereich der Aufgabe geht meist etwas unter, was aber nicht unbedingt schlecht ist, da das zu erreichende Ziel mir persönlich oft schon allein durch den Quellcode klar wurde.
  Gerade für Schüler ist es da schon problematischer, dass der Bereich Variablen relativ "unentdeckt" bleibt. Zumindest ging mir es so.
  Allerdings wüsste ich nicht wie ich es besser machen würde, angesichts auch der Tatsache, dass dieses Feld im Vergleich zu den anderen weniger Priorität besitzt und somit auch nicht aufdringlich werden sollte. Desshalb auch die Bewertung "gut".
- Angenehmer dunkler Hintergrund, kontrastreiche Tour, sinnvolle Anordnung der Bereiche. Navigationsmenü aufgeräumt, anfangs aber nicht als solches erkennbar
- bspw. könnten Änderungen im Bereich Variablen durch einen kurzen flash signalisiert werden
- Die Variablen sollen untereinander in der Tabelle erscheinen. Zumal die Spaltenbreite sonst unterschiedlich ist.
- Der Untergrund ist zu dunkel und für eine längere Dauer der Bearbeitung nicht günstig - Farbgebung Bernstein o.ä. wäre für Augen und Kontrast günstiger.
  Die Aufgabenstellung müsste deutlich vom anderen Text abgehoben sein.
  Es müsste eine Möglichkeit geben, den "Programmteil" des Editors bei längeren Beispielen noch einmal im Ganzen sehen zu können.

3. Wie beurteilen Sie die folgenden Funktionen der Anwendung?

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)			
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±	W
Hilfe-Tour	11x	28,21	20x	51,28	6x	15,38	1x	2,56	-	-	-	-	1x	1,92	0,75	69%
Bereich "Einleitung"	14x	35,90	20x	51,28	3x	7,69	1x	2,56	1x	2,56	-	-	-	1,85	0,87	85%
Ein- / Ausblenden der Be...	7x	17,95	24x	61,54	3x	7,69	3x	7,69	-	-	-	-	2x	2,05	0,78	49%
Sortierung der Codeblöc...	17x	43,59	17x	43,59	4x	10,26	1x	2,56	-	-	-	-	-	1,72	0,76	83%
Auswahlfelder für Variab...	10x	25,00	24x	60,00	4x	10,00	1x	2,50	-	-	-	-	1x	1,90	0,68	71%
Hinweis-Tooltip für Code...	13x	32,50	20x	50,00	4x	10,00	1x	2,50	-	-	-	-	2x	1,82	0,73	76%
Fehlermeldungen	14x	35,00	20x	50,00	5x	12,50	1x	2,50	-	-	-	-	-	1,83	0,75	84%
Automatische Wiedergab...	13x	33,33	21x	53,85	3x	7,69	2x	5,13	-	-	-	-	-	1,85	0,78	74%
Manuelle Wiedergabe de...	13x	32,50	22x	55,00	3x	7,50	1x	2,50	-	-	-	-	1x	1,79	0,70	68%
Steuerung der Geschwin...	10x	25,64	21x	53,85	6x	15,38	2x	5,13	-	-	-	-	-	2,00	0,79	71%
Visualisierung	20x	50,00	14x	35,00	5x	12,50	1x	2,50	-	-	-	-	-	1,68	0,80	86%
Variablen-Tabelle	12x	30,00	18x	45,00	9x	22,50	1x	2,50	-	-	-	-	-	1,98	0,80	73%



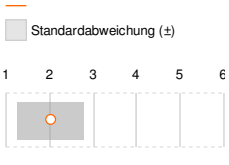
4. Anmerkungen zu den oben genannten Funktionen von EduCode

- Einleitungsbereich sollte auch den Nutzen herleiten ...
  Waru brauchen wir denn eine Schleife?
- die Aufgabenstellung lässt mehr Eigenarbeit der Schüler erwarten. Tatsächlich bringt der Schüler die Anweisungen "nur" in die richtige Reihenfolge. Wünschenswert wäre hier eine mögliche Differenzierung:
  vorgegebene Variante: der Schüler schreibt Quelltext selbst im Editor
  Variante 2 (leichter): ein vorgegebenes Grundgerüst muss der Schüler vervollständigen
  Variante 3 (ganz leicht): der Schüler bringt die vorgegebenen Anweisungen in die richtige Reihnefolge

  Die Fehlermeldungen sind sehr detailliert - auch hier evtl. differenzieren
- - toller Baustein, erweitert die Einsatzmöglichkeiten
- Vielleicht ein zwei Beispiele mehr, auch mit anderen Funktionen..Wurzel..Quadrat...
- Hilfetour oder Hinweis darauf vor dem ersten Start einblenden
- Die Fehlermeldungen sind durch die große Anzahl möglicher Fehler nicht immer eindeutig. Das ist sicher auch nicht Sinn der Sache zu jedem DAU Fehler eine eindeutige Fehlermeldung zu generieren. Aber es sollte z. B. in der Einleitung darauf verwiesen werden, dass die Fehlermeldung eben nicht immer eindeutig ist.
- Es dauert recht lang die Hilfe Tour bei Beginn der Benutzung des Programms zu durchlaufen
- Nach einem korrektem Durchlauf fehlt eine Bestätigung
- Visualisierung der Schugröße ist noch nicht gelungen (es reicht nicht die Größenbezeichnung zu ändern, der Schuh sollte tatsächlich größer oder kleiner werden)
- Aufgabe sollte nicht ausgeblendet werden können.
  Mehr Freiheiten bei den Auswahlfeldern wären wünschenswert.

5. Wie beurteilen Sie die Metapher mit dem Regal?

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Regal	9x	22,50	23x	57,50	6x	15,00	2x	5,00	-	-	-	-	-	2,03	0,77



6. Anmerkungen zu den Modulen in der Gruppe Variablen

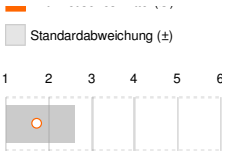
- Regal wird oft als Metapher für Speicher verwendet.
  Deshalb verwende ich für Variablen eine Kistensammlung als Metapher. Dabei haben die Kisten unterschiedliche Größen (da Variablen entsprechend ihres Typs auch unterschiedlich viel Speicherplatz benötigen)
- Es fehlt eine Anmerkung zur größe des "Regals"
  also das das Regal je nach benötigten variablen unterschiedlich groß sein kann
- Wirklich schön geschrieben und ich denke für Anfänger oder Schüler auch ein einprägsames Beispiel
- Ich würde das Regal beschriften mit einem Namen. Also Name der Variable = Beschriftung für Regalfach. und dann kann man verschiedene Dinge dort hineinlegen und herausnehmen. Und es ist wichtig dass gesagt wird, dass in einem Regal immer nur EIN Wert liegen kann.
  Ich finde allerdings nicht so schön das die variablentopen nicht mit angegeben werden müssen (String, Zahl, Farbe,...) Ist mir zu sehr miteinander vermischt.
- Die Variablennamen rot, gelb, blau suggerieren eher schon Werte von Variablen. Sicher lässt sich so das Regal besser farblich zuordnen. Vielleicht wäre eine Metapher im Sinne von Briefkästen mit Namensschild überlegenswert.

  Der Aspekt der Wertzuweisung spielt bisher leider keine Rolle. Erfahrungsgemäß stolpern Schüler häufig über die Nichtvertauschbarkeit der Seiten einer Wertzuweisung. Nicht umsonst schreiben Sie :=

  Das Problem der Typverträglichkeit könnte auch thematisiert werden.
- Problem bei Regal ist, dass die Variablen dort irgendwie physisch zusammenhängen, was nicht wirklich der Fall ist, besser wäre z.B. die Metapher Schukarton
- Eine vertikale anstatt einer horizontalen Tabelle wäre besser.

7. Wie beurteilen Sie die Metapher mit dem Akkustand des Smartphones??

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Akkustand	19x	48,72	14x	35,90	4x	10,26	2x	5,13	-	-	-	-	-	1,72	0,86

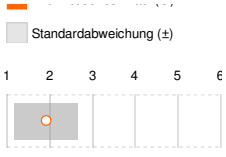


8. Anmerkungen zu den Modulen in der Gruppe Verzweigungen

- Ebenfalls sehr gutes Beispiel, aus dem Alltag gegriffen und deswegen relevant
- - passt gut in die Erfahrungswelt der Schüler
- Ein 'Weiter' Button nach Beendigung einer Aufgabe, sodass nicht jedes mal das gesamte Menü geöffnet werden muss wäre schön.
- Veranschaulichung des Herunterzählens der Anzahl wäre gut
- sehr einfallsreich!
- Vertauscht man die Meldungen im Programm "Einseitige Verzweigung" kommen irreführende Fehlermeldungen.
- Farbgebung bzgl Kontrast prüfen.

9. Wie beurteilen Sie die Metapher mit dem Akkustand des Smartphones??

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Wasserglas	11x	28,21	21x	53,85	6x	15,38	1x	2,56	-	-	-	-	-	1,92	0,74

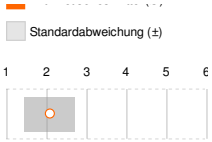


10. Anmerkungen zu den Modulen in der Gruppe Verzweigungen

- aber man könnte ja auch bei dem bekannten Akku bleiben... oder das Wasserglas durchziehen
- - kleinere Fehler:
  im Einleitungsteil: 9 Durchläufe,
  fußgesteuert: "Denn dann ist x(10) nicht mehr größer als 10"?
  DIN 5008: 200 ml statt 200ml
  - Zählschleife: vielleicht könnte als Wert der Variablen Anzahl auch ein Wert kleiner oder gleich Startwert angeboten werden

11. Wie beurteilen Sie die Metapher mit dem Flächenberechnung?

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Flächenberechnung	5x	12,82	27x	69,23	6x	15,38	1x	2,56	-	-	-	-	-	2,08	0,62



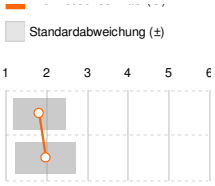
12. Anmerkungen zu den Modulen in der Gruppe Verzweigungen

- Motivation der Schüler ist bei Beispielen aus ihrer Lebenswelt höher als bei mathematischen Problemstellungen
- Deklarationsfehler: breite deklariert, laenge als Parameter verwendet

  var hoehe := 10;
  var breite := 15;
  procedure berechneFlaecheninhalt(breite, laenge)
- - Fehler bei Prozeduraufruf berechneFlaecheninhalt(breite,laenge); (statt laenge müsste hoehe stehen)
  - Wie wärs mit formalen Parametern im UP, also z. B. x, y statt laenge, breite?
  - Prozedur mit Variablenparameter:
  Druckfehler "innheralb"
  Aufgabenstellung bedingt nicht unbedingt die Notwendigkeit der aufsummierten Flächeninhalte
- Eine Trennung des Unterprogramms zum Prozeduraufruf wäre wünschenswert
- aus mathematischer Sicht sollte die Funktion aber berechneFlaecheninhaltRechteck heißen
- Das Modul sollte mit weiteren Anwendungsbeispielen(Bsp. Berechnung des Volumens von Quadern, Mittelwert ,...) erweitert werden.

13. Wie beurteilen Sie die Metaphern mit dem Schuh und der Wiedergabeliste?

	Sehr gut (1)		Gut (2)		Befriedigend (3)		Ausreichend (4)		Mangelhaft (5)		Ungenügend (6)		Weiß nicht (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Schuh	11x	28,21	25x	64,10	2x	5,13	1x	2,56	-	-	-	-	-	1,82	0,64
Wiedergabeliste	10x	25,64	21x	53,85	7x	17,95	1x	2,56	-	-	-	-	-	1,97	0,74



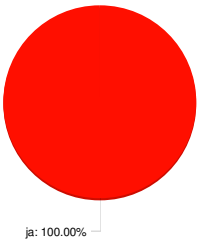
14. Anmerkungen zu den Modulen in der Gruppe Datenstrukturen

- Objektdefinition ist auf diese weise fragwürdig, es gehört immer in eine Klasse bzw. muss in einer Klasse erzeugt und instanziiert werden. Ich arbeite nur objektorientiert und hatte meine Not mit dem Verständnis der Aufgabe
- Fehlermeldung: Wieso muss "Größe" dem Schuh nach "Farbe" zugeordnet werden?
- Unklar wieso der Schuh erst eine Farbe und dann eine Größe braucht.

15. Finden Sie die Umsetzung als Webanwendung sinnvoll?

39 (100.0%): ja

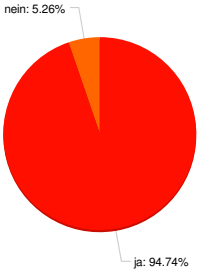
- (0.0%): nein



16. Würden Sie die Anwendung im Unterricht einsetzen?

36 (94.7%): ja

2 (5.3%): nein



19. Warum würden Sie die Anwendung nicht im Unterricht einsetzen?

- ich bin kein Lehrer
- Als Übungsmaterial für SUS zuhause eine Ergänzung, aber um Erklärungen geben zu können müsste für mich die Einleitung ebenfalls visualisiert und einzeln dargestellt werden können wie das für die Aufgabe gelöst wurde oder die Lösung für die Aufgabe sofort ausführbar sein.

17. Welche Verbesserungsvorschläge haben Sie für die Anwendung?

- Anwendungen plastischer gestalten, in einen Kontext setzen.
  Die Anwendung ist von der Navigation her sehr gewöhnungsbedürftig und unübersichtlich.
  Evtl. dynamisch Teile ausblenden oder farblich absetzen.
  Expertenmodus in denen nicht nur Code-Zeilen sondern kleinere Fragmente angeordnet werden müssen, wäre wünschenswert.

  Ein oder mehrere komplexe Aufgaben am Ende, als Abschluss, wären förderlich. Schüler wollen immer m ende einn Erfolg sehen :-)
- nein
- ich würde die Anwendung gern einsetzen, arbeite aber mit Lazarus --> wenn die Auswahl der Programmiersprache (Java oder Pascal) möglich wäre, würde ich die Anwendung sofort nutzen
- Die verketteten Listen sind nicht leicht verständlich
- Als offline Version.
  Lösungen für Lehrpersonen nach Anmeldung oder offline verfügbar machen.

  Es wird übrigens nicht eine Zeile, sondern ein Kommando mit ; beendet (Einleitung Wertzuweisungen)
- Typverträglichkeit (s. o.)
- wichtigste Hilfestellungen direkt unüberlesbar anzeigen, aber nur ganz wenige, wie drag and drop.
  weitere Hilfen so optional anbieten
- Vielleicht wären ein paar einleitende Worte sinnvoll?
- Einleitungstext z.T. noch etwas einfacher formulieren (zB bei den Listen)
- wirkt sehr schlicht, was an sich nicht unbedingt negativ, aber wichtige Informationen/Stellen sollten hervorgehoben werden
- es gibt noch inhaltliche Fehler, die abgestellt werden sollten, besonders im Bereich Unterprogramme. Hier fehlt die Unterscheidung formaler und aktueller Parameter, was wiederum für das Verständnis von Werte- bzw. Referenzparamter sinnvoll wäre. Funktionen und Prozeduren sind besonders auch beim Aufruf unterschiedlich und nicht "Im Gegensatz zu Funktionen besitzen Prozeduren keinen Rückgabewert. Das Schlüsselwort return ist hierbei also unzulässig. Ansonsten funktionieren Prozeduren auf die gleiche Art wie Funktionen"
  Auch "Änderungen an den Variablen innerhalb der Prozedur beeinflussen nicht die Variablen, die beim Funktionsaufruf angegeben wurden (Call by Value). " ist inkonsistent.

  Bei Operationen var gelb/gruen prüfen
  Bei Einleitung zweiseitige V. fehlt eine }

  Auch unklar bleibt, wann nun ein bestimmter Schleifentyp verwendet werden sollte...

18. Weitere Anmerkungen, die im Fragebogen nicht angesprochen wurden.

- Wie geht es weiter? Ist es nur für das Verständisse von Funktionen schleifen etc? Was passiert nun????
  Mit welchem Programmierprogramm ist es danach sinnvoll anzufangen??
  Robot Karol, Skratch oder ??
- Einbindungsmöglichkeit in E-Learningumgebungen (z. B. Moodle) zwecks Abrechnungsfunktionen zum Bearbeitungsstand

20. Welche Tätigkeit führen Sie aus?

9 (23.7%): LehrerIn (Informatik)

- (0.0%): LehrerIn (Sonstiges)

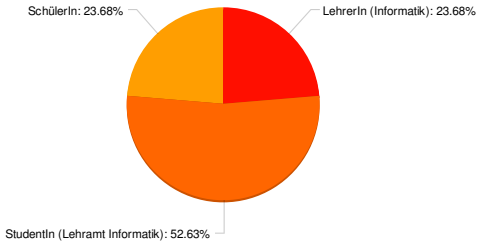
20 (52.6%): StudentIn (Lehramt Informatik)

- (0.0%): StudentIn (Medien-/Informatik)

- (0.0%): StudentIn (Sonstiges)

9 (23.7%): SchülerIn

- (0.0%): Sonstiges



21. Wie alt sind Sie?

- (0.0%): < 18

24 (61.5%): 18 - 25

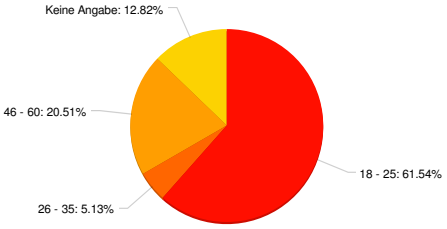
2 (5.1%): 26 - 35

- (0.0%): 36 - 45

8 (20.5%): 46 - 60

- (0.0%): > 60

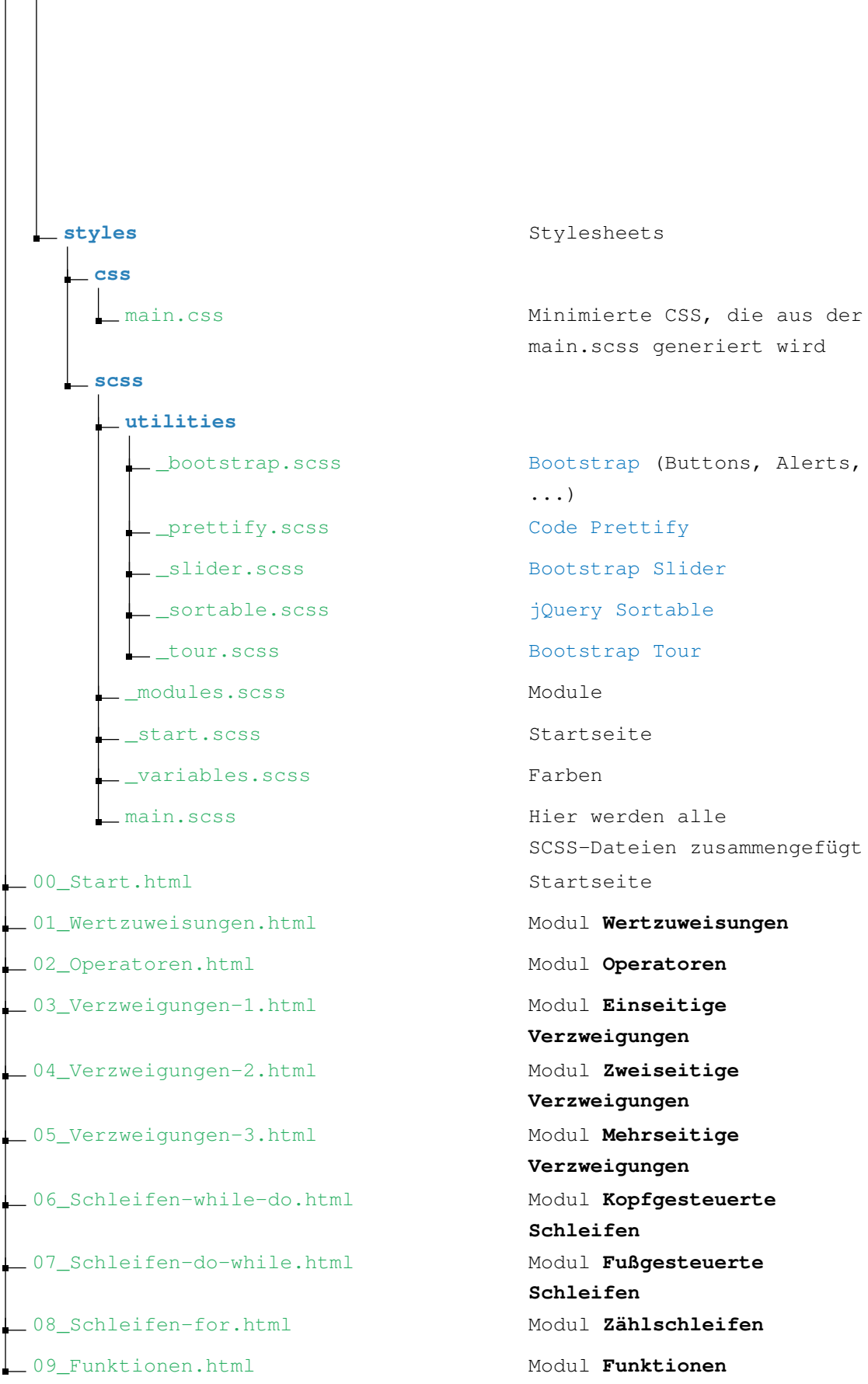
5 (12.8%): Keine Angabe



D Dokumentation

D.1 Ordnerstruktur

07-schleifen-do-while.js	Modul **Fußgesteuerte Schleifen**
08-schleifen-for.js	Modul **Zählschleifen**
09-funktionen.js	Modul **Funktionen**
10-prozeduren-1.js	Modul **Prozeduren mit Werteparameter**
11-prozeduren-2.js	Modul **Prozeduren mit Referenzparameter**
12-objekte.js	Modul **Objekte**
13-listen-1.js	Modul **Verkettete Listen erstellen**
14-listen-2.js	Modul **Verkettete Listen durchsuchen**
baseModule.js	Basismodul, von dem die Submodule erben. Die Submodule können auf alle Funktionen des Basismoduls zugreifen.
main.js	Hauptklasse: Navigation
navigation.js	HTML Navigation in JS-Variable, damit Navigation in alle Module geladen werden kann
tour.js	Bootstrap Tour - Definition
libs	Externe Bibliotheken
animate	Animate.css
bootstrap	Bootstrap
fontawesome	Font Awesome
jquery	jQuery und jQuery Sortable
prettify	Code Prettify
raphael	Raphael.js
scrollbar	jQuery Custom Scrollbar
slider	Bootstrap Slider
tour	Bootstrap Tour


styles
Stylesheets
css
main.css
Minimierte CSS, die aus der main.scss generiert wird
scss
utilities
_bootstrap.scss
Bootstrap (Buttons, Alerts, ...)
_prettify.scss
Code Prettify
_slider.scss
Bootstrap Slider
_sortable.scss
jQuery Sortable
_tour.scss
Bootstrap Tour
_modules.scss
Module
_start.scss
Startseite
_variables.scss
Farben
main.scss
Hier werden alle SCSS-Dateien zusammengefügt
00_Start.html
Startseite
01_Wertzuweisungen.html
Modul Wertzuweisungen
02_Operatoren.html
Modul Operatoren
03_Verzweigungen-1.html
Modul Einseitige Verzweigungen
04_Verzweigungen-2.html
Modul Zweiseitige Verzweigungen
05_Verzweigungen-3.html
Modul Mehrseitige Verzweigungen
06_Schleifen-while-do.html
Modul Kopfgesteuerte Schleifen
07_Schleifen-do-while.html
Modul Fußgesteuerte Schleifen
08_Schleifen-for.html
Modul Zählschleifen
09_Funktionen.html
Modul Funktionen

10_Prozeduren-1.html	**Prozeduren mit Werteparameter**
11_Prozeduren-2.html	Modul **Prozeduren mit Referenzparameter**
12_Objekte.html	Modul **Objekte**
13_Listen-1.html	Modul **Verkettete Listen erstellen**
14_Listen-2.html	Modul **Verkettete Listen durchsuchen**

D.2 Modul hinzufügen

Um das hinzufügen von Modulen zu erleichtern wurden Template-Dateien erstellt (*educode/xx_Template.html* und *educode/assets/js/modules/xx-template.js*). Dieses Template-Modul kann auch bereits im Browser aufgerufen werden und enthält ein einfaches Beispiel, um die Funktionsweise eines Moduls leichter nachvollziehen zu können. Sie dienen zusätzlich als Grundlage für das erstellen neuer Module.

D.2.1 Benötigte Dateien duplizieren

1. Die Datei **educode** → **xx_Template.html** duplizieren und dem Modul entsprechend umbenennen. (Beispiel: 15_Variablentypen.html)
2. Die Datei **educode** → **assets** → **js** → **modules** → **xx-template.js** duplizieren und dem Modul entsprechend umbenennen. (Beispiel: 15-variablentypen.js)

D.2.2 HTML-Datei anpassen

In der erzeugten **15_Variablentypen.html** den [TITEL], die [BESCHREIBUNG] und die [AUFGABE] anpassen.

Codeblöcke können in der Beschreibung auf die folgende Weise hinzugefügt werden:

```
<pre class="prettyprint"><code>var x := 12;
var y := 20;</code></pre>
```

Innerhalb des pre-Tags werden Umbrüche, Tabulatoren und Leerzeichen ausgewertet und dargestellt. Daher die etwas unübersichtliche Formatierung.

In der Zeile 136 (unter *Current Module*) die JS von *xx-template.js* zu *15-variablentypen.js* verändern.

D.2.3 JS-Datei anpassen

Codeblöcke erstellen

In der Variable *this.editor_codeBlocks* werden die Codeblöcke für das Modul in der richtigen Reihenfolge angegeben.

index
Der Index fängt bei 0 an und wird für jeden Codeblock hochgezählt. Er dient zur Identifizierung des Codeblocks.

text
Der Text enthält den Code, der letztendlich im Codeblock angezeigt wird. HTML-Code wird ausgewertet und kann zur Formatierung genutzt werden.

description
Die Beschreibung wird in einem Tooltip angezeigt, wenn der Nutzer mit der Maus über das Fragezeichen am Ende eines Codeblocks fährt. Wenn die Beschreibung weggelassen wird, wird auch das Fragezeichen in diesem Codeblock nicht angezeigt.

methods
In dem Array können verschiedene Methoden angegeben werden, die bei der Aktivierung des Codeblockes im Editor ausgeführt werden.

Da in allen Modulen Variablen definiert werden und diese in der Tabelle im Bereich unter der Visualisierung angezeigt werden sollen, wurde dafür eine Methode im Basismodul erstellt. Diese kann auf folgende Weise dem methods-Array hinzugefügt werden:

```
methods: [{
    method: this.setVariable,
    parameter: { 'x': '20' }
}]
```

In der *parameter*-Eigenschaft werden die Variablen mit ihren Werten angegeben, die in diesem Schritt hinzugefügt werden sollen.

Es können jedoch auch eigene Methoden in der Modulklasse definiert werden (Siehe in der *xx-template.js* die Methode **customMethod**).

Regeln festlegen

In der Variable *this.editor_codeBlock_rules* werden die Regeln für die Reihenfolge der Codeblöcke festgelegt. Dafür stehen drei verschiedene Varianten der Regeldefinition zur Verfügung:

Zeile X soll vor Zeile Y stehen

Zeile 5 muss vor der Zeile 6 stehen.

```
0: {
    line: 5,
    before: 6,
    message: "Die Variable <b>x</b> muss definiert werden, " +
             "bevor sie verwendet werden kann."
}
```

Zeile X darf nicht zwischen Zeile Y und Z stehen

Zeile 5 darf nicht zwischen Zeile 0 und 3 liegen.

```
0: {
        line: 5,
        before: 0,
        after: 3,
        message: "Die Variable <b>x</b> darf nicht " +
                 "innerhalb der Funktion definiert werden."
}
```

Zeile X muss zwischen Zeile X1 und Y1 oder X2 und Y2 oder [...] stehen

Zeile 1 muss zwischen Zeile 7 und 9 **oder** direkt nach Zeile 11 **oder** genau zwei Zeilen nach Zeile 11 stehen.

```
0: {
  line: 1,
  between: [
    {'before': 7, 'after': 9},  // Zwischen Zeile 7 und 9
    {'before': 11},             // Direkt nach Zeile 11
    {'before': 11, 'no': 2}     // Genau 2 Zeilen nach Zeile 11
  ],
  message: "Ueberpruefe die Position der Klammern."
}
```

Visualisierung erstellen

Die meisten Visualisierungen werden während der Ausführung bestimmter Codeblöcke ausgeführt. Für bestimmte Module wird jedoch eine Grundvisualisierung benötigt, wie zum Beispiel bei den Schleifen das Wasserglas.

Diese Visualisierungen werden in der Funktion *this.initModuleVisualization* erzeugt. In dem Template wird dort zum Beispiel ein Rechteck gezeichnet. Diese Methode wird bei jedem Start und auch bei jedem Reset erneut ausgeführt, um wieder zum ursprünglichen Zustand der Visualisierung zu gelangen.

D.2.4 Link in Navigation hinzufügen

Unter **educode** → **assets** → **js** → **navigation.js** kann ein neuer Link in einer existierenden Kategorie erstellt werden.

Um eine neue Kategorie zu erstellen, müssen die **Bootstrap Columns** angepasst werden (siehe `http://getbootstrap.com/css/#grid`).

D.3 Bibliotheken

Animate.css

Beschreibung: Ansammlung von verschiedenen CSS-Animationen.

Webseite: `https://daneden.github.io/animate.css/`

Lizenz: MIT

Version: 3.5.0

Verwendung:

- Startseite: Einfliegen des Titels
- Startseite: Blinken des Cursors

Bootstrap

Beschreibung: CSS-Framework mit Gestaltungsvorlagen für Buttons, Formulare, Grid-Systeme etc.

Webseite: http://getbootstrap.com/

Lizenz: MIT

Version: 3.3.6

Verwendung:

- Grid-System
- Buttons
- Editor: Fehlermeldungen (Alerts)
- Editor: Hinweis-Tooltips

Bootstrap Slider

Beschreibung: Erweitert Bootstrap um einen Slider.

Webseite: http://www.eyecon.ro/bootstrap-slider

Lizenz: Apache License v2.0

Version: 2.0.0

Verwendung:

- Steuerung: Wiedergabegeschwindigkeit

Bootstrap Tour

Beschreibung: Erweitert Bootstrap um eine Tour mittels Popovers.

Webseite: http://bootstraptour.com/

Lizenz: MIT

Version: 0.10.2

Verwendung:

- Hilfe- / Willkommenstour

Font Awesome

Beschreibung: Ansammlung von Icons.

Webseite: http://bootstraptour.com/

Lizenz: MIT (CSS), SIL OFL 1.1 (Fonts)

Version: 4.5.0

Verwendung:

- Animation in Modulen mit verketteten Listen: Musikspieler-Icon
- Buttons

jQuery

Beschreibung: Vereinfacht die Javascript-Programmierung.

Webseite: https://jquery.com/

Lizenz: MIT

Version: 2.2.0

jQuery Custom Scrollbar

Beschreibung: Ermöglicht spezifische Scrollleisten.

Webseite: http://manos.malihu.gr/jquery-custom-content-scroller/

Lizenz: MIT

Version: 3.1.3

Verwendung:

- Überall wo Scrollleisten verwendet werden

jQuery Sortable

Beschreibung: Ermöglicht das Sortieren von Elementen per Drag & Drop.

Webseite: https://johnny.github.io/jquery-sortable/

Lizenz: Modified BSD License

Version: 0.9.13

Verwendung:

- Editor: Zum Sortieren der Codeblöcke

Code Prettify

Beschreibung: Formatiert Codeblöcke.

Webseite: https://github.com/google/code-prettify

Lizenz: Apache License v2.0

Version: Version vom 13.04.2016

Verwendung:

- Einleitung: Beispiel-Codeblöcke
- Editor: Codeblöcke

Raphael.js

Beschreibung: Vereinfacht das zeichnen und animieren von SVG-Elementen.

Webseite: http://dmitrybaranovskiy.github.io/raphael/

Lizenz: MIT

Version: 2.2.0

Verwendung:

- Zeichnen und animieren der Visualisierungen

Erklärung

Hiermit erkläre ich, Herr Willi Zobel, die vorliegende Masterarbeit zum Thema

**Implementierung einer Webanwendung zur Visualisierung
von Programmiergrundlagen im Informatikunterricht**

selbständig und ausschließlich unter Verwendung der im Quellenverzeichnis aufgeführten Literatur- und sonstigen Informationsquellen verfasst zu haben.

Dresden, 02. August 2016