Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Entwicklung eines modularen Workshops für Themen aus dem Software Engineering

Marco Scheffel, B. A.

Studiengang:	Technikpädagogik, B. Sc.
Prüfer/in:	Prof. Dr. Stefan Wagner
Betreuer/in:	Verena Ebert, M. Sc.
Beginn am:	11. Dezember 2019
Beendet am:	27. März 2020

Kurzfassung

Ziel dieser Arbeit ist die Entwicklung eines Workshops für junge Menschen um diese für das Studium der Softwaretechnik zu begeistern. Bisherige Workshops und didaktische Theorien werden für dieses Ziel herangezogen. Der entwickelte Workshop wurde durchgeführt und evaluiert. Die Ergebnisse aus dem gehaltenen Workshop und die Evaluation stimmen hinsichtlich der Zielerreichung positiv und der Workshop erweist sich als eine solide Basis für zukünftiges Arbeiten.

Inhaltsverzeichnis

1. Einleitung **13**

2. Vorhandene Workshopkonzepte **15**
2.1. Softwareentwicklung für ein SmartHome 15
2.2. Workshop zum systematischen Testen 16

3. Didaktische Theorie **19**
3.1. Bildungstheoretische Didaktik nach Wolfgang Klafki 19
3.2. Lerntheoretische Didaktik (Berliner Modell) 21
3.3. Didaktische Ansätze und Methoden 23

4. Technische Vorbedingungen **27**
4.1. Betriebssystem 27
4.2. Bau und Programmierung der Roboter 30
4.3. Netzwerkinfrastruktur 31

5. Workshop **39**
5.1. Definition der Workshopziele 39
5.2. Analyse der Zielgruppe 40
5.3. Workshopmodule 40
5.4. Möglicher Workshop mit 90 - 110 Minuten 42
5.5. Ideen zur Nutzung einzelner Module für Messen oder andere Veranstaltungen . 43

6. Evaluation **47**
6.1. Experteninterview 47
6.2. Durchführung des Workshops, Beobachtung und Teilnehmerfeedback 48
6.3. Erkenntnisse 50

7. Ausblick **53**
7.1. Erweiterung des Moduls: Systematisches Testen im SystemTestPortal 53
7.2. Erweiterung des Moduls: Programmierung von Robotern: Programmierung mit Python 53
7.3. Alternative Programmierumgebung mittels OpenRoberta 54
7.4. Programmierung mittels Tablets 54

8. Fazit **55**

Literaturverzeichnis **57**

A. Anhang **61**
A.1. Ausführliche Evaluation des Workshops am 24.07.2019 61

A.2. Begründung für die Bewertung der Betriebssysteme 63
A.3. Transkript Experteninterview 65
A.4. Abbildungen 66
A.5. Tabellen 74

Abbildungsverzeichnis

3.1. Hauptlinien didaktischer Positionen 20
3.2. Berliner Modell 22
3.3. Anatomie Android Betriebssystem vor Version 5.0 24
3.4. Schaubild Betriebssystem 25

4.1. Dateistruktur von Porteus auf einem USB-Stick. 29
4.2. Porteus Module 30
4.3. Möglicher Aufbau eines LEGO Mindstorm NXT Roboters 32
4.4. Möglicher Aufbau eines LEGO Mindstorm EV3 Roboters 33
4.5. NXT Programme 35
4.6. EV3 Programme 36
4.7. Möglicher Aufbau eines Dexter Industries BrickPi Roboters 37
4.8. Netzwerktopologie für einen Workshop 38

5.1. NXT Programm Parkourschalter 43
5.2. BrickPi Programm 44
5.3. Ablaufplan Workshop 90 - 110 Minuten 45

A.1. Checkliste zur Workshopdurchführung 67
A.2. Aufgabenblatt 1: Systematisches Testen 69
A.3. Aufgabenblatt 2: Programmieren 71
A.4. Feedbackbogen 73

Tabellenverzeichnis

4.1. Bewertung Betriebssysteme nach Anforderungsanalyse 28

6.1. Ergebnisse der Evaluation des Testworkshops am Informatiktag 2020 51

A.1. Operatorenliste 75

Akronyme

AFB Anforderungsbereich. 24

Endgerät PC bzw. Laptop. 27

MINT Mathematik / Naturwissenschaften, Informatik, Technik. 13

STP SystemTestPortal. 16

SuS Schülerinnen und Schüler. 24

TuT Teilnehmerinnen und Teilnehmer. 15

1. Einleitung

Immer wieder kann den Medien entnommen werden, dass in den sog. Mathematik / Naturwissenschaften, Informatik, Technik (MINT)-Berufen ein Fachkräftemangel herrscht [Bun19, S. 19]. Betrachtet man die Statistiken konkret, entsteht ein etwas differenzierteres Bild: Es besteht zwar kein umfassender MINT-Mangel, jedoch eine zunehmende Knappheit an qualifizierten Arbeitskräften, besonders durch viele Ruhestandseintritte in den nächsten Jahren und der steigenden Anzahl der zu besetzenden Stellen. Immer mehr Ausbildungsbetriebe und Hochschulen legen daher vermehrt ihren Fokus darauf, junge Menschen für eine Ausbildung oder ein Studium in diesem Bereich zu gewinnen.

Der in dieser Arbeit entwickelte Workshop zielt darauf ab junge Menschen spielerisch an die Themen der Softwaretechnik heranzuführen und im besten Falle für diese zu begeistern. Hierfür werden Themen aus dem Bereich der Softwareentwicklung herangezogen: Testen und Implementieren von Software.

Im Rahmen dieser Arbeit werden zuerst bestehende Workshopkonzepte des Instituts für Softwaretechnik betrachtet und analysiert, um aus den bereits bestehenden Erfahrungen zu lernen. Bevor auf dieser Grundlage ein neuer Workshop entwickelt wird, findet eine Betrachtung der didaktischen Theorie statt um ein pädagogisch-wissenschaftliches Fundament für den neuen Workshop zu schaffen. Ausgehend von dieser Untersuchung werden die technischen Vorbedingungen betrachtet um schließlich einen auf pädagogischen Pfeilern gestützten wissenschaftlich fundierten Workshop zu beschreiben.

Die eingangs gestellte Zielformulierung, junge Menschen durch den Workshop für die Softwaretechnik zu begeistern, wird schließlich in einem konkreten Workshop auf die Probe gestellt und evaluiert. Basierend auf der Evaluation wird ein Ausblick gegeben, wie der Workshop weiterentwickelt werden kann und in einem abschließendem Fazit überlegt, inwiefern der entwickelte Workshop die gesteckten Ziele erreicht.

2. Vorhandene Workshopkonzepte

Seit mindestens zehn Jahren wird an der Universität Stuttgart von der Fakultät 05: Informatik, Elektrotechnik und Informationstechnik der Informatiktag angeboten [Roh20; Vis20]. Die vom Institut für Softwaretechnik angebotenen Workshops basieren auf LEGO Mindstorm Robotern bzw. auf Dexter Industry BrickPi Robotern. Seit 2015 sind drei verschiedene Workshops angeboten worden: *Workshop: LEGO Mindstorms - Spielend programmieren lernen*, *Workshop: Softwareentwicklung für ein Smart Home* und *Workshop: Systematisches Testen autonom fahrender Fahrzeuge*. Als Basis für die didaktische Entwicklung eines neuen, modularen Workshops werden die zwei zuletztgenannten Konzepte herangezogen und näher betrachtet. Zum erstgenannten Workshop sind keine Unterlagen verfügbar.

2.1. Softwareentwicklung für ein SmartHome

2.1.1. Beschreibung des Workshops

Basierend auf dem SmartHomeSimulator von Ivan Bogicevic [Bog18] hat dieser einen Workshop für den Informatiktag 2017 konzipiert und zuletzt 2018 abgehalten [Roh20]. Der Workshop ist auf neun Teilnehmer ausgelegt, erweitert eine bestehende Java-Software und testet die Ergebnisse an einem SmartHome-Modell. Zur Analyse steht keine Teilnehmerevaluation zur Verfügung, allerdings können die vorhandenen Unterlagen des Referents unter didaktischen Gesichtspunkten näher betrachtet werden.

Der neunzigminütige Workshop ist in neun Phasen eingeteilt, wobei sich theoretische Teile (Vorstellung, Einführung und Abschluss/Feedback) sowie praktische Teile (SmartHome-Modell kennen lernen, Codeanalyse, Implementierung und Test) abwechseln. Die Präsentation zum einführenden Vortrag umfasst Themen wie die Vorstellung der Referenten, der Universität und begeleitet anschließend die Teilnehmer durch die praktischen Aufgaben des Workshops.

2.1.2. Folgerungen aus dem Workshopkonzept

Ohne den Workshop selbst erlebt zu haben wirken die Inhalte des Workshops sehr herausfordernd und setzen höchstwahrscheinlich erweiterte Grundkenntnisse voraus. Die fehlende Differenzierung führt möglicherweise dazu, dass ein Teil der Teilnehmerinnen und Teilnehmer (TuT) den Inhalten des Workshops nicht folgen kann und daher weniger motiviert ist. Eine mehrfache Differenzierung wäre sinnvoll.

Die Nutzung eines SmartHome-Modells ist eine praktikable Möglichkeit um den Teilnehmern einen virtuellen Anker zu bieten, an dem sie ihre Entwicklungsergebnisse sofort testen und im Einsatz sehen können. Die Mobilität des Modells ist allerdings eingeschränkt, weshalb der Workshop örtlich an den Computerpool-Raum des Institutes gebunden ist, in dem das Modell steht.

Äußerst positiv schätze ich die Mischung aus Theorie, Implementierung und Testen ein, denn so bleibt der Workshop abwechslungsreich und der Wechsel der unterschiedlichen Tätigkeiten führt zu einer erhöhten Teilnehmeraktivität.

Zusammenfassend bewerte ich die Einführungsprasentation und den hohen Anteil an praktischen Aufgaben mit hoher Teilnehmeraktivität als sehr gut. Der Grundgedanke, die Teilnehmer sowohl testen als auch programmieren zu lassen, erachte ich als erstrebenswert um einen interessanten Workshop zu entwickeln. Als problematisch betrachte ich, dass in Java programmiert wird, da zwar davon auszugehen ist, dass die meisten TuT an Informatikthemen interessiert sind, es aber nicht sichergestellt sein kann, dass Java-Grundlagen vorhanden sind. Die hierdurch entstehenden Unwegsamkeiten beim Workshopablauf als auch Demotivation bei den TuT stellen ein Risiko dar.

2.2. Workshop zum systematischen Testen

2.2.1. Beschreibung des Workshops

Der Workshop zum systematischen Testen wurde im Rahmen einer Bachelorarbeit von Ali Ayan konzipiert. „Das Ziel dieser Bachelorarbeit ist die Konzipierung eines Anfänger-Workshops zum systematischen Testen. Dazu wird die Frage gestellt: Kann ein Workshop mit dem Thema ‚systematisches Testen' interessant genug aufgebaut werden, damit es die Teilnehmer motiviert einen informatiknahen Studiengang zu belegen." [Aya19] Als Oberfläche für die systematischen Tests wurde das SystemTestPortal (STP) genutzt, das im Rahmen von Studentenprojekten am Institut für Softwaretechnik an der Universität Stuttgart entwickelt wird [Kul+19]. Die Laptops, die zum Ausführen des STPs genutzt werden, booten in ein Linux-Live-System.

Der Workshop besteht aus einer Einführung und zwei praktischen Teilen, bei denen zuerst vordefinierte Tests im STP durchgeführt werden. Anschließend erstellen die TuT selbst Tests, welche danach von anderen Gruppen bearbeitet werden. Für jeden der beiden praktischen Teile sind ausführliche Aufgabenblätter vorhanden, die den Ablauf beschreiben. Zwischen den zwei Einheiten wird eine kurze Auswertung der Ergebnisse und Erkenntnisse durchgeführt. Zum Abschluss des Workshops findet eine Evaluation statt.

2.2.2. Folgerungen aus dem Workshopkonzept

Diesen Workshop kann ich aus eigener Erfahrung bewerten, da ich bei einer Durchführung durch Verena Ebert und Daniel Kulesz am 24.07.2019 teilnahm. Durch Beobachtung konnte ich erkennen, dass die Grundidee des Workshops gut umgesetzt wurde und das Ziel das Thema des *systematischen Testens* zumindest teilweise erreicht werden konnte. Der Workshop schien jedoch Mängel in der Einführung zu haben, da den Teilnehmern nicht klar zu sein schien, warum getestet werden muss.

Das lag möglicherweise daran, dass der Kontext für die Tests fehlte (Kap. 3.3.4) oder die TuT zu wenig Vorwissen / Interesse an Informatikthemen hatten.

Die Arbeitsblätter waren durchdacht, bestanden allerdings aus zu viel Text, während der Aufbau des STP nicht erklärt wurde. Nach meinen Beobachtungen führte das dazu, dass die Teilnehmer viel Testzeit verloren hatten um sich zuerst selbstständig in die Funktion des STP einzuarbeiten. Eine grafische Erklärung des STP hätte möglicherweise den Umgang mit der Software erleichtert, wodurch die Teilnehmer mehr Zeit zum *systematischen Testen* gehabt hätten.

Ein weiteres Problem, das ich beim Workshop beobachten konnte, sind die unterschiedlichen Roboter-Typen (LEGO Mindstorm NXT und EV3 sowie Dexter Industries BrickPi), insbesondere da die BrickPi-Roboter immer per Kabel eingesteckt werden mussten um die zu testenden Programme zu starten.

Eine weiterer Stolperstein des Workshops war der zweite Aufgabenteil. In diesem erstellten die Teilnehmer selbst Tests, welche eine andere Gruppe im STP aufrufen und ausführen sollte. Zur Vernetzung wurde jeweils zwischen zwei Laptops ein AdHoc-Netzwerk aufgebaut, welches allerdings recht instabil war. Ein Netzwerk mittels Crossover-Kabel ist in diesem Workshopkonzept keine Option, da die BrickPis über das Netzwerkkabel angeschlossen und programmiert wurden. Eine Zusammenfassung des gesamten Feedbacks wurden von Daniel Kulesz dokumentiert, siehe hierzu Anhang A.1.

Zusammenfassend zeigt der Workshop, dass durch die Nutzung des STP *systematisches Testen* durchaus interessant gestaltet werden kann, allerdings hat der Workshop einige technische und didaktische Schwierigkeiten. Die Idee Teilnehmer selbst Tests entwickeln zu lassen, erachte ich als erstrebenswert, kann allerdings bei Teilnehmern mit wenig Vorkennnissen dazu führen, dass diese überfordert sind. Die Umsetzung mittels vorkonfigurierten Betriebssystem ist sehr gut und werde ich für den noch zu entwickelnden Workshop übernehmen, allerdings muss für die BrickPi Roboter eine Alternative zur kabelgebundenen Steuerung gefunden werden.

3. Didaktische Theorie

Die Durchführung eines Workshops, egal zu welchem Thema, verlangt dem Referenten pädagogisches Handeln ab. „Professionelles pädagogisches Handeln […] setzt voraus, dass pädagogisch Handelnde selbst über das verfügbare Wissen verfügen und dasselbe situationsadäquat einsetzen können.“ [Nic08, S. 2] Hierbei hilft der Rückgriff auf wissenschaftliche Erkenntnisse und objektive Theorien. Daher wird zur Gestaltung des Workshops zuerst ein Blick in die pädagogischen Theorien, in die Didaktik, geworfen.

Vorab die Einschränkung, dass im Folgenden lediglich ein Überblick über einzelne, zentrale didaktische Theorien gegeben werden kann. Weder ist dieses Kapitel eine vollumfängliche Zusammenfassung aller bestehenden didaktischen Theorien und Modelle, noch kann auf jede Kritik dieser eingegangen werden. Vielmehr sollen die beschriebenen Konzepte einen Rahmen für die Erarbeitung des im späteren Verlauf dieser Arbeit erstellen Workshops bieten.

Als Grundlage werden die ersten modernen didaktischen Theorien betrachtet, da alle späteren Theorien auf diesen aufbauen, bzw. Weiterentwicklungen dieser sind. Natürlich ist die Didaktik heute schon wesentlich weiter als noch 1960, dennoch kann nicht über Didaktik geschrieben werden ohne die *bildungstheoretische Didaktik* von *Wolfgang Klafki* zu beachten. Abbildung 3.1 zeigt den Verlauf der Hauptlinien und die gegensietige Beeinflussung der modernen didaktischen Positionen aus welchen,wie bereits erwähnt, die *bildungstheoretische Didaktik* nach *Klafki* und die lerntheoretische Didaktik nach *Paul Heimann*, *Gunter Otto* und *Wolfgang Schulz* näher betrachtet werden.

In den folgenden Anschnitten wird häufig von Unterricht gesprochen, da sich die didaktische Forschung primär mit der Gestaltung von Unterricht befasst. Der Begriff *Unterricht* wird immer als Synonym zu *Workshop* genutzt. Ebenso wird der Begriff *Lehrender* als Synonym für den den *Workshopreferenten* genutzt bzw. *Schülerinnen und Schüler* und *Lernender* im Sinne von *Teilnehmerinnen und Teilnehmer*.

3.1. Bildungstheoretische Didaktik nach Wolfgang Klafki

Bei der bildungstheoretischen Didaktik von *Klafki* steht der Bildungsbegriff im Vordergrund. [Kla63, 135ff.]. Die bildungstheoretische Didaktik entstammt aus den Geisteswissenschaften und „im Mittelpunkt steht […] die Frage, nach welchen Gesichtspunkten Bildungsinhalte bestimmt und ausgewählt werden können.“ [Nic08] Die Erwähnung dieser Theorie ist besonders wichtig, da sie als zentraler Ausgangspunkt der modernen Didaktik angesehen wird und deren Grundgedanken hilfreich bei der Unterrichtsplanung sind [Pet92, S. 77].

Klafki fomuliert fünf Leitfragen, die beschreiben welche Vorüberlegungen beim Unterrichtsentwurf für ein bestimmtes Thema bedacht werden sollten. Sie lassen sich wie folgt zusammenfassen:

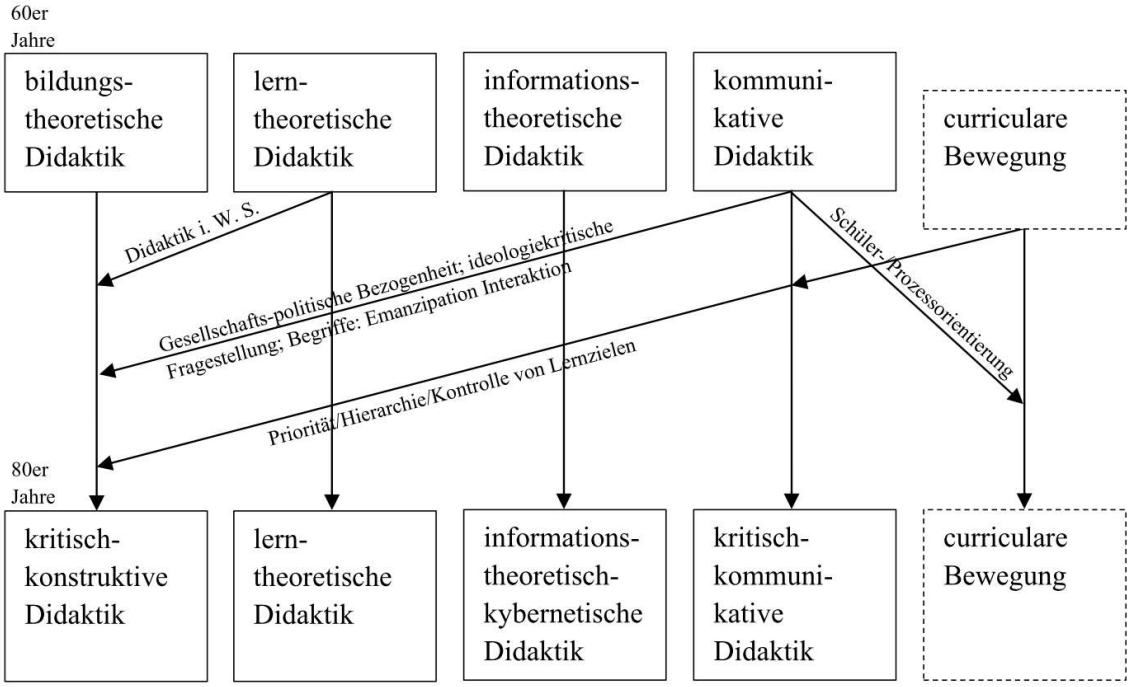


Abbildung 3.1.: Hauptlinien gegenseitiger Beeinflussung didaktischer Positionen [Pet92].

1. *Exemplarität*: Welchen größeren bzw. allgemeinen Problemzusammenhang kann das Thema erschließen?
2. *Gegenwartsbedeutung*: Welche Bedeutung hat das Thema bereits im geistigen Leben der TuT und welche Bedeutung sollte es aus pädagogischen Überlegungen erhalten?
3. *Zukunftsbedeutung*: Worin liegt die Bedeutung des Themas für die Zukunft der TuT?
4. *Inhaltliche Struktur*: Wie ist die Struktur des zu vermittelnden Inhalts unter Berücksichtigung von 1., 2. und 3.?
5. *Zugänglichkeit*: Welches sind die besonderen Fälle, Phänomene, Beispiele, an denen den TuT das Wesen des Themas interessant und begreifbar gemacht werden könnte?

Die Frage nach der inhaltlichen Struktur fächert *Klafki* in sechs Teilfragen weiter auf. Von diesen sechs Fragen sind für die Entwicklung eines Workshop für Themen des Software Engeneerings zwei Fragen besonders interessant (die anderen Fragen beziehen sich auf die Planung von Unterrichtssequenzen[1]):

1. Welches sind die einzelnen Momente des Inhalts als eines Sinnzusammenhanges?
5. Welche Eigentümlichkeiten des Inhalts werden den TuT den Zugang zur Sache vermutlich schwer machen?

[1] Ein zusammenhängender Unterricht über mehrere Wochen bzw. Monate.

Die von *Klafki* aufgezeigten Leitfragen sind als didaktische Grundlage insbesondere interessant, da sowohl die *Strukturiertheit* als Qualitätsmerkmal von Unterricht empirisch bestätigt ist als auch die Betrachtung des Ausgangswissen des Lernenden ein Prädikator für Lernerfolg darstellt. [Nic08, S. 38]. Die fünfte Leitfrage nach der Zugänglichkeit zeigt auf, wie wichtig es ist, das Interesse der Teilnehmer zu wecken um so wiederrum einen positiven motivationalen Effekt zu erzeugen. Da die Motivation der Lernenden im Unterricht von zentraler Bedeutung ist, geht *Klafki* auch auf diese näher ein. So fomuliert er weitere drei Fragen, die einem Lehrenden helfen sollen die richtige Auswahl an Sachverhalten, Phänomene, Situationen, Versuche, Anschauungen, Hinweise und Aufgaben zu treffen um eine möglichst hohe Motivation der Lernenden zu erlangen. [Kla63, S. 140ff] Im Ramen dieser Arbeit wird dies nicht näher betrachtet. Die Motivation der TuT soll unter Zuhilfenahme der in Abschnitt 3.3 beschriebenen Methoden geweckt werden.

Die aufgestellten Leitfragen von *Klafki* stellen zwar gute Richtlinien da, die bei der Analyse und Planung von Lehr-Lernprozessen helfen, geben aber kaum konkrete Handlungsanweisungen um Unterricht zu planen und jede Situation muss neu bewertet werden. Weiter wird an *Klafkis* Modell kritisiert, dass sich die Frage nach der *Exemplarität* als hochkomplex erweisen kann, denn es ist nicht gesichert, dass der Transfer von spezifischen Erkenntnissen auf allgemeine Sachverhalte stattfindet. [Nic08, S. 39] Im Falle der Planung eines Workshop ist die Kritik, dass jede Situation individuell betrachtet werden muss, nicht relevant, da es nicht um die Planung einer ganzen Unterrichtssequenz geht. Lediglich die Frage nach der *Exemplarität* muss genauer betrachtet werden, was im Rahmen der Betrachtung von *didaktischer Reduktion* in Abschnitt 3.3.1 geschieht.

3.2. Lerntheoretische Didaktik (Berliner Modell)

Die *lerntheoretische Didaktik* entstand als Gegenentwurf zur *bildungstheoretischen Didaktik* von *Klafki* und ist in der Ursprungsfassung erfahrungswissenschaftlich orientiert. [KJS14; Nic08] Allgemein ist die lerntheoretische Didaktik unter dem Begriff *Berliner Modell* bekannt. Der Grundgedanke hinter diesem Modell ist es dem Lehrnenden die Möglichkeit zu schaffen die Wirklichkeit selbst empirisch zu erschließen, da es den Autoren nicht möglich scheint für jede Eventualität der pädagogischen Wirklichkeit eine detaillierte wissenschaftliche Orientierungshilfe zu geben, wie es z.B. Hilbert Meyer [Mey03] versucht. *Heimann*, *Otto* und *Schulz* gehen in ihrem Modell von der Annahme aus, dass alle Unterrichtsprozesse strukturelle Ähnlichkeiten und sechs Strukturelemente aufweisen: Zwei Bedingungsfaktoren (soziokulturelle und anthropologisch-psychologische Voraussetzungen und deren Folgen) und vier Entscheidungsfelder (Intentionen, Inhalte, Methoden und Medien) [HOS72].

Unter soziokulturellen Voraussetzungen verstehen sich gesellschaftliche Einflüsse, die sich auf das Lehrgeschehen auswirken, wie z. B. gesellschaftlich-politische Strukturen. Anthropologisch-psychologische Voraussetzungen dagegen sind die Persönlichkeitsmerkmale der Lernenden, also deren kognitive, emotionale und motivationale Voraussetzugen.

Zwischen den Entscheidungsfeldern existieren Interdependenzen, also können z. B. bestimmte Intentionen nur mit bestimmten Inhalten, Methoden und Medien erreicht werden, welche wiederum alle durch die Bedingungsfaktoren eingeschränkt bzw. beeinflusst werden. Veranschaulicht wird das Berliner Modell durch das in Abbildung 3.2 dargestellte Schaubild.

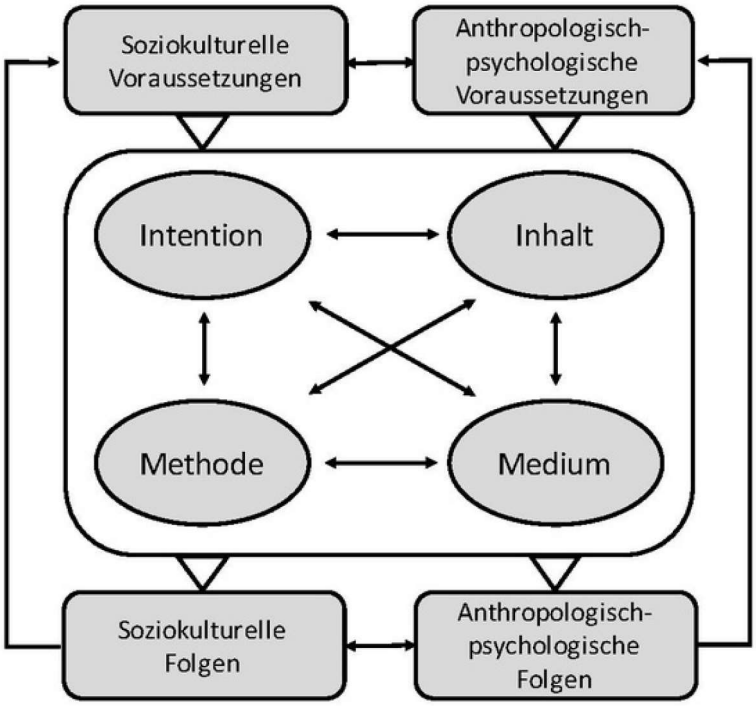


Abbildung 3.2.: Darstellung des Berliner Modells als didaktisches Entscheidungsmodell nach Paul Heimann (1962) [SR15]

Ähnlich wie bei der bildungstheoretischen Didaktik nach *Klafki* besteht die Leistung des *Berliner Modells* insbesondere darin, dass Lehrende eine Orientierungshilfe erhalten, welche sie bei der Planung und Analyse von Lehr-Lernprozessen berücksichtigen können. *Heimann*, *Otto* und *Schulz* gehen allerdings einen Schritt weiter und stellen „eine ganze Reihe von Planungsbeispielen […] zur Verfügung“ [Nic08, S. 47], welche allerdings offen lassen, welche sozio-kulturellen und anthropoligisch-psychologischen Voraussetzungen in welcher Situation zu berücksichtigen sind. Durch den Modellcharakter gibt es viele weitere Ansätze, die auf den Theorien von *Heimann*, *Otto* und *Schulz* aufbauen, es würde allerdings zu weit führen diese hier weiter auszuführen. [2]

Daraus resultierend folgt dür den hier zu erarbeitenden Workshop, dass die Zielgruppe des Workshops zuvor genau definiert sein muss (Abschn. 5.2), da sich in Abhängigkeit der anthropologisch-psychologische Vorraussetzungen unterschiedliche Methoden fpr eine Personengruppe eignen. Ein weiteres Ergebnis basierend auf der lerntheoretischen Didaktik ist, dass sich beispielorientierte Lernmaterialien in Verbindung mit einer situationsflexiblen Unterstützung durch die Lehrkraft als günstig erweisen [SRG03], was bei der Gestaltung des Lernszenarios in Abschnitt 3.3.4 berücksichtigt wird.

[2] Zur vertiefenden Recherche ist insbesondere [AKB14; Blo73; HW97; Wei00] zu empfehlen.

3.3. Didaktische Ansätze und Methoden

In Abgrenzung zu den beschriebenen didaktischen Theorien zielen Ansätze und Methoden mehr auf die Umsetzung didaktischer Überlegungen als auf eine allgemeine Strukturierung didaktischer Probleme ab. Folgend werden der Ansatz der didaktischen Reduktion und der Kompetenzorientierung sowie die Methode des didaktischen Einstiegs und des Lenszenarios beschrieben.

3.3.1. Ansatz: Didaktische Reduktion

Im Rahmen von Unterricht stellt sich immer die Frage, wie komplexe, fachwissenschaftliche Sachverhalte so vermittelt werden können, dass Lernende diese verstehen und begreifen. Bei dieser Fragestellung hilft der Ansatz der didaktischen Reduktion, welcher auf *Dietrich Hering* und seiner Veröffentlichung „Zur Fasslichkeit naturwissenschaftlicher und technischer Aussagen“ von 1958 zurück geht, damals noch unter dem Begriff der *didaktischen Vereinfachung* [Her59; Nic08, S. 59]. 1978 wird der von *Hering* vorgestellte Ansatz durch *Gustav Grüner* weiter differenziert, indem er die didaktische Reduktion in zwei Dimensionen teilt: Die horizontale und vertikale Reduktion [Grü78; Nic08, S 61].

Der Grundgedanke der didaktischen Reduktion ist, dass eine wissenschaftliche Aussage so vereinfacht wird, dass Lernende entsprechend ihres aktuellen Lernstandes in der Lage sind diese zu begreifen. Bei der von *Hering* und *Grüner* beschriebenen vertikalen didaktischen Reduktion wird der Gültigkeitsumfang der Ursprungsaussage reduziert. Ähnliches passiert bei der horizontalen didaktischen Reduktion, allerdings wird hierbei der Gültigkeitsumfang nicht reduziert, „die wissenschaftliche Aussage wird nur konkreter – oft unter Zuhilfenahme von Analogien, Metaphern und Beispielen – dargestellt und damit leichter eingänglich gemacht.“ [Grü78, S. 85] In der Praxis kommen meist Mischformen der vertikalen und der horizontalen didaktischen Reduktion vor.

Abbildung 3.3 zeigt die Anatomie eines Android Betriebssystems vor Version 5.0. Diese Abstraktion ist vertikal reduziert, indem die detaillierten Programmcodes der einzelnen Module nicht dargestellt werden und horizontal reduziert, indem die Module sortiert und kategorisiert werden.

Abbildung 3.4 zeigt eine andere Abstraktion der Rolle eines Betriebssystems. Es wird wesentlich stärker vertikal reduziert, indem z.B. die einzelnen Module des Betriebssystems nicht näher erläutert werden. Eine horizontale didaktische Reduktion findet dagegen kaum statt, da alle Elemente, die mit dem Betriebssystem in Bezug stehen gezeigt werden.

3.3.2. Ansatz: Kompetenzorientierung

Im Bildungssektor geschieht aktuell eine „Paradigmenverschiebung von der Input-Orientierung hin zu Output-Orientierung“ [Bra08], was bedeutet, dass zunehmend zielorientiertes und individuelles Lernen gefördert wird. Dies stellt die Hinwendung zu einem kompetenzorientierten Unterricht dar. Der Bildungsplan für das Fach Informatik am dreijährigen beruflichen Gymnasium in Baden-Württemberg beschreibt diese Kompetenzorientierung wie folgt: „Die Schülerinnen und Schüler erwerben im Fach Informatik Kompetenzen, die sie befähigen, ein Problem mit den dazu zur Verfügung stehenden Daten zu analysieren, im Sinne eines prozessorientierten Vorgehens zu lösen und mit

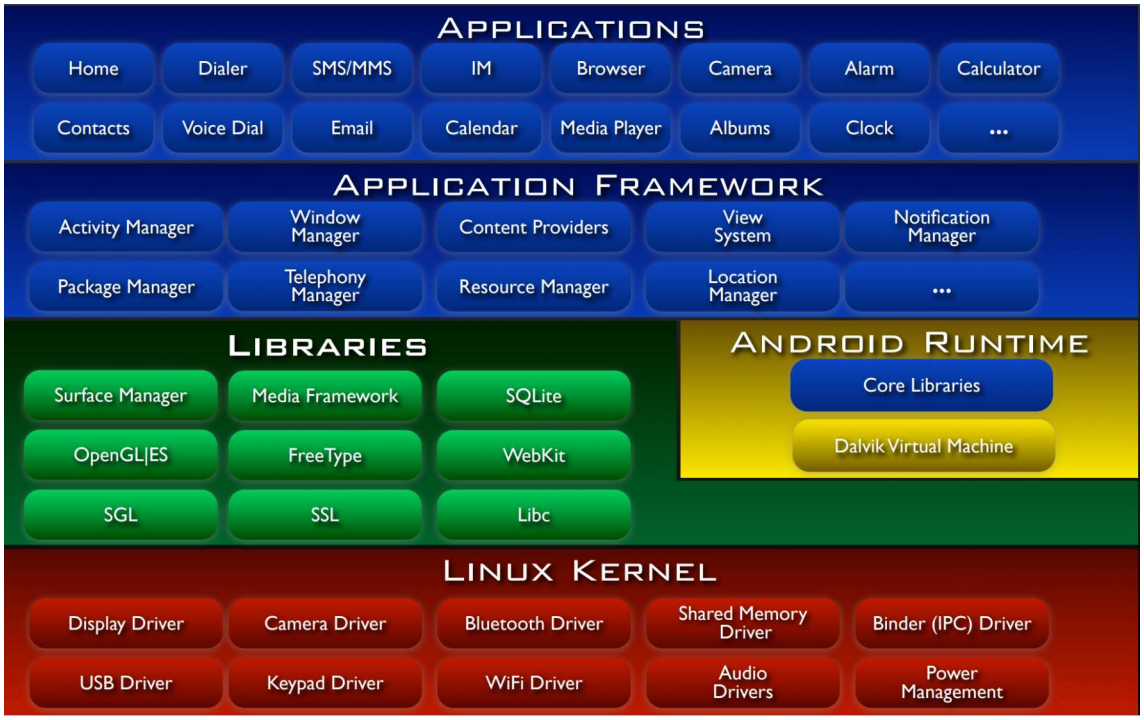


Abbildung 3.3.: Abstrahierung der Anatomie des Android Betriebssystems nach Patrick Brady vor Android Version 5.0 [Bra12].

fachspezifischen Werkzeugen und Methoden hinreichend zu dokumentieren. Die im Fach Informatik erworbenen Kompetenzen beziehen sich sowohl auf die Inhalte des Faches Informatik, als auch auf zentrale Prozesse und Arbeitsweisen der Informatik als angewandte Wissenschaft.“ [Min19]

Auch der hier entwickelte Workshop ist kompetenzorientiert aufgebaut, weshalb in Abschnitt 5.4 für alle Einheiten des Workshops die in [Min19] beschriebenen Operatoren genutzt werden. Die Operatoren selbst sind in drei Anforderungsbereich (AFB) eingeteilt, wobei AFB I die Wiedergabe von bekannten Sachverhalten, AFB II die selbstständige Verwendung bekannter Sachverhalte und AFB III das planmäßige Verarbeiten komplexer Gegebenheiten umfasst.

Da die Nutzung von Operatoren ein sehr taugliches Hilfsmittel ist um Unterricht- und Lehreinheiten zu planen, sind diese in Anhang A.5 aufgelistet.

3.3.3. Methode: Didaktischer Einstieg

Im schulischen Unterricht ist der Unterrichtseinstieg eine der Hauptphasen und dient zur Hinführung zum Unterrichtsthema [Mey15]. Die Phase des Unterrichtseinstiegs als didaktische Methode zu bezeichnen mag für einen Pädagogen befremdlich wirken, macht aber im Rahmen dieser Ausarbeitung durchaus Sinn, da diese als Mischform von Methode und Phase im Workshop in Abschnitt 5.4 genutzt wird.

Ein Einstieg erfüllt mehrere Aufgaben. So dient er einer Strukturierung und Stabilisierung, indem die Schülerinnen und Schüler (SuS) sich einem Ritual *beugen*, wodurch Ruhe in die Klasse kommt und ed dem Lehrer möglich ist die Klasse in Richtung seines Unterrichtsgegenstands zu lenken [Mey15].

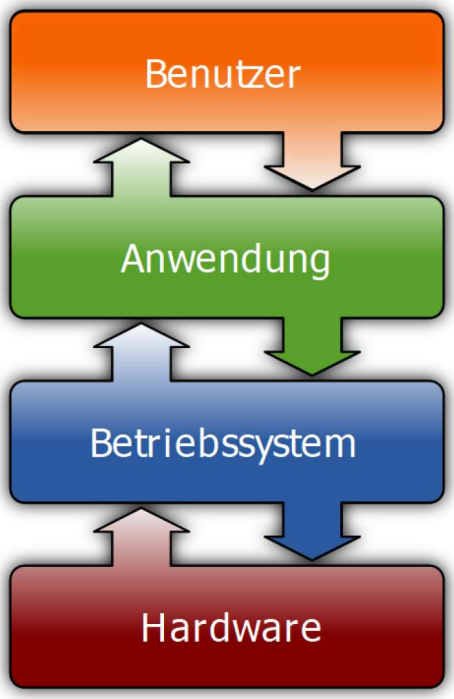


Abbildung 3.4.: Zusammenhang zwischen Betriebssystem, Hardware, Anwendungssoftware und dem Benutzer [Wik20a]

Den SuS ist dieses Ritual bereits bekannt, wodurch für die SuS klar ist, dass nun der Unterricht beginnt. Im schulischen Umfeld bietet dieses Ritual eine Orientierungshilfe und wirkt gleichzeitig integrativ, indem neue SuS ebenfalls an diesem Ritual teilnehmen. Für den hier zu entwickelnden Workshop ist der zentrale Effekt, dass eine Spannung auf das Neue, das zu Erlebende und zu Erforschende aufgebaut wird und die SuS so zur Arbeit motiviert werden sollen..

Im Rahmen des didaktischen Einstiegs werden die Regeln für den gesamten Workshop definiert und das Gelingen oder Scheitern desselben liegt meist am Gelingen oder Scheitern des Einstiegs. Im hier entwickelten Workshop wird die in Abschnitt 2.1 beschriebene Präsentation als Vorlage genutzt und entsprechend der hier beschriebenen didaktischen Theorien und Modelle angepasst.

3.3.4. Methode: Lernszenario

Der Workshop entspricht einer Lernsituation, welche durch ein Lernszenario in ein sinnstiftendes Umfeld eingebettet wird [Rie01]. Die Methode des Lernszenarios orientiert sich an der in [Ber+06] beschriebenen Methode, wird allerdings im Workshop in erweiterter Form eingesetzt, so dass diese beinahe schon Rollenspielcharakter annimmt. Das so gegebene sinnstiftende Lernszenario soll den TuT helfen die Aufgaben einzuordnen und so ein persönliches Ziel zu bilden. Grund für die Wahl dieser Methode sind die Erkenntnisse bei der Untersuchung des *Workshops: Systematisches Testen* in Abschnitt 2.2.2.

Die Methode wird im Workshop so genutzt, dass in der didaktischen Einführung (Kap. 3.3.3) den SuS beschrieben wird, dass sie sich aktuell in einem *Escape Room* befinden, aus welchem sie unter Zuhilfenahme eines Roboter ausbrechen sollen. Im Rahmen dieses Szenarios macht es Sinn zuerst dessen Funktionsweise zu studieren und dann die Funktionsfähigkeit des Roboters zu testen um diesen anschließend zu programmieren.

4. Technische Vorbedingungen

Nachdem nun die theoretische Grundlage für den zu erarbeitenden Workshop geschaffen ist, wird in diesem Kapitel die technischen Vorarbeit für den Workshop betrachtet. Die in diesem Kapitel beschriebenen Vorbereitungen sind durchzuführen, egal in welcher Form der Workshop gehalten wird. In den einzelnen Abschnitten wird daher abgewogen und begründet, weshalb bestimmte Systeme bzw. Vorgehensweisen genutzt werden.

Zuerst wird die Frage nach dem optimalen Betriebssystem gestellt und beantwortet. Anschließen werden verschiedene Robotersysteme vorgestellt sowie deren Aufbau erörtert. Abschließend wird die Netzwerkinfrastruktur beschrieben, die für den Workshop empfehlenswert ist. In Abbildung A.1 im Anhang findet sich eine Checkliste mit allen benötigten Materialien um einen Workshop durchzuführen.

4.1. Betriebssystem

Zur Bearbeitung von Aufgaben sollen die Teilnehmer des Workshops an PCs bzw. Laptops (Endgeräte) arbeiten, schließlich ist es ein Workshop zu Themen aus dem Software Engineering. Diese Endgeräte benötigen ein Betriebssystem und die Auswahl dessen ist schwerer, als es auf den ersten Blick zu sein scheint. Ähnlich wie im Software Engeneering wird zuerst eine Anforderungsanalyse durchgeführt, anschließend werden mögliche Betriebssysteme betrachtet, ob diese den Anforderungen grundsätzlich gewachsen sein könnten. Basiernd auf diesen Erkenntnissen wird ein finales Betriebssystem gewählt, mit welchem der Workshop statt findet.

4.1.1. Anforderungsanalyse an das Betriebssystem

1. Ein Workshop soll möglichst einfach in der Vorbereitung und flexibel in der Ausgestaltung sein. (Einfach und flexibel)
2. Die Konfiguration des Betriebssystems muss vor Veränderungen durch Anwender geschützt sein. (Geschützt gegen Veränderung)
3. Nach Starten des Systems sollten die Referenten möglicht keine individuellen Konfigurationen mehr vornehmen müssen. (Vorkonfiguriert)
4. Das Betriebssystem sollte direkt von einem Wechseldatenträger (bevorzugt USB-Stick) startbar sein, um einen meist langwierigen Installationsprozess zu umgehen und kurzfristig auf alternative Endgeräte ausweichen zu können. (Wechseldatenträger)
5. Das Bestriebssystem muss auf älteren Endgeräte möglichst flüssig funktionieren. (Funktionsfähig auf alten Endgeräte)

6. Auf dem Betriebssystem muss die Software STP ausführbar sein, da ansonsten ein Netzwerk mit einem Server, auf dem die Software installiert ist, mit geplant werden muss. (STP)
7. Die Programmierung zumindest eines der Roboter-Typen muss über das Betriebssystem möglicht einfach gewährleistet sein. (Programmierung)

4.1.2. Auswahl des Betriebssystems

Für die Wahl des Betriebssystems kommen macOS, Windows oder Linux in Betracht. MacOS wird allerdings nicht näher betrachtet, da Apple Betriebssysteme laut Software Lizenzvertrag nur auf Apple Hardware installiert werden dürfen und dies sowohl den finanziellen Rahmen, als auch den Verwaltungsaufwand des Workshops sprengen würde [App, S. 34].

Die Auswertung der Anforderungsanalyse für die Betriebssysteme Windows und Linux ist in Tabelle 4.1 dargestellt, wobei die Einschätzung anhand von Schulnoten durchgeführt ist (1 = sehr gut bis 6 = ungenügen). Eine ausführliche Begründung für die jeweilige Einschätzung kann in Anhang A.2 nachgelesen werden.

Mit einer Gesamtnote von 1,64 erfüllt Linux die Anforderungen besser als Windows mit einer Note von 2,21, wobei insbesondere die Schwierigkeiten Windows von einem Wechseldatenträger zu starten und Einschränkungen beim Betrieb auf älterer Hardware zu einer Abwertung führen.

Bleibt nun also die Frage offen, welche Linux-Distribution alle Anforderungen erfüllt, was bei Weitem keine leichtere Frage ist als die Wahl des Betriebssystems. Bei der Durchsicht aller in Frage kommender Distrubutionen [Wik20b] fiel die Wahl auf das Slackware-Derivat *Porteus*, da es 32-bit fähig ist, den systemschonenden LXDE-Desktop unterstützt und die komplette Konfiguration komfortabel durch einzelne Module eingerichtet wird. Auf der Webseite von Porteus wird es als schnell, portabel und modular angepriesen [Por20], was in einer längeren Testphase bestätigt werden konnte.

Anforderung	**Windows**	**Linux**
1 Einfach und flexibel	2	2,5
2 Geschützt gegen Veränderung	2	1
3 Vorkonfiguriert	2	2
4 Wechseldatenträger	4	1
5 Funktionsfähig auf alten Endgeräte	3	1
6 STP	1,5	1
7 Programmierung	1	3
Ergebnis (Mittelwert)	2,21	1,64

Tabelle 4.1.: Bewertung der Betriebssysteme Windows und Linux nach Schulnoten anhand den in Kapitel 4.1.1 aufgestellten Anforderungen. Die ausführliche Begründung für die Bewertung findet sich in Anhang A.2.

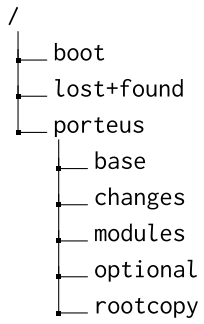


Abbildung 4.1.: Dateistruktur von Porteus auf einem USB-Stick.

4.1.3. Konfiguration des Betriebssystems

Eine ausführliche Anleitung, wie Porteus konfiguriert und personalisiert wird, ist auf der offiziellen Webseite `http://www.porteus.org/` zu finden. In diesem Kapitel wird beschrieben, was im Rahmen des Workshops bei der Konfiguration des Betriebssystems zu beachten ist.

Nach Herunterladen von Porteus von einem offiziellen *Mirror* muss es nur auf einem USB-Stick entpackt und dieser mittels mitgeliefertem Script bzw. Programmdatei im Ordner `/boot` bootfähig gemacht werden. Anschließend kann das Betriebssystem direkt auf einem beliebigen Endgerät, das von USB starten kann, gestartet werden.

Porteus ist modular aufgebaut, wobei die Module als *.xzm-Pakete* bereitgestellt werden, ein speziell für Porteus entwickeltes Archiv-Format. Alle für das System relevanten Pakete liegen im Ordner `porteus`, welcher fünf weitere Unterordner enthält (siehe Abbildung 4.1).

Im Ordner `base` liegen die Hauptdateien des Systems, in welchem dauerhafte Anpassung, wie z. B. das Tastaturlayout vorgenommen werden können.

Der Ordner `changes` wird genutzt um Veränderungen am System zu speichern. Hierbei wird das System nicht geändert, sondern eine Kopie der Ordnerstruktur im Ordner `changes` angelegt, in der alle geänderten Dateien abgelegt werden. Durch Angabe des Startparameters `changes=/porteus/` werden beim Start des Betriebssystem alle im Ordner `/changes` enthaltene Dateien nach allen anderen Modulen geladen, wodurch die Änderungen quasi persistent sind. Diese Funktion wird für den Workshop ausgeschaltet, bzw. der Startparameter nicht genutzt, da das System entsprechend Anforderung 2 nicht durch Anwender änderbar sein soll.

In den Ordnern `/modules` und `/optional` liegen die Module der Programme, die das System zusätzlich laden soll, wobei die Module in `/modules` beim Start geladen werden und Module im Ordner `/optional` nachträglich mit `root`-Rechten gestartet werden können. Für den Workshop werden neben vom Betriebssystem vorgegebene modulen das *SystemTestPortal*, der Internetbrowser *Firefox* und Desktoplinks zu den BrickPis als Modul bereitgestellt.

Änderungen an Systemmodulen und Modulerstellung werden als `root`-Benutzer durchgeführt. Bestehende Pakete werden direkt in der Systemoberfläche über das Kontextmenü entpackt, bearbeitet und wieder gepackt. Neue Module werden als Ordner angelegt, wobei die Pfade im Ordner so sind,

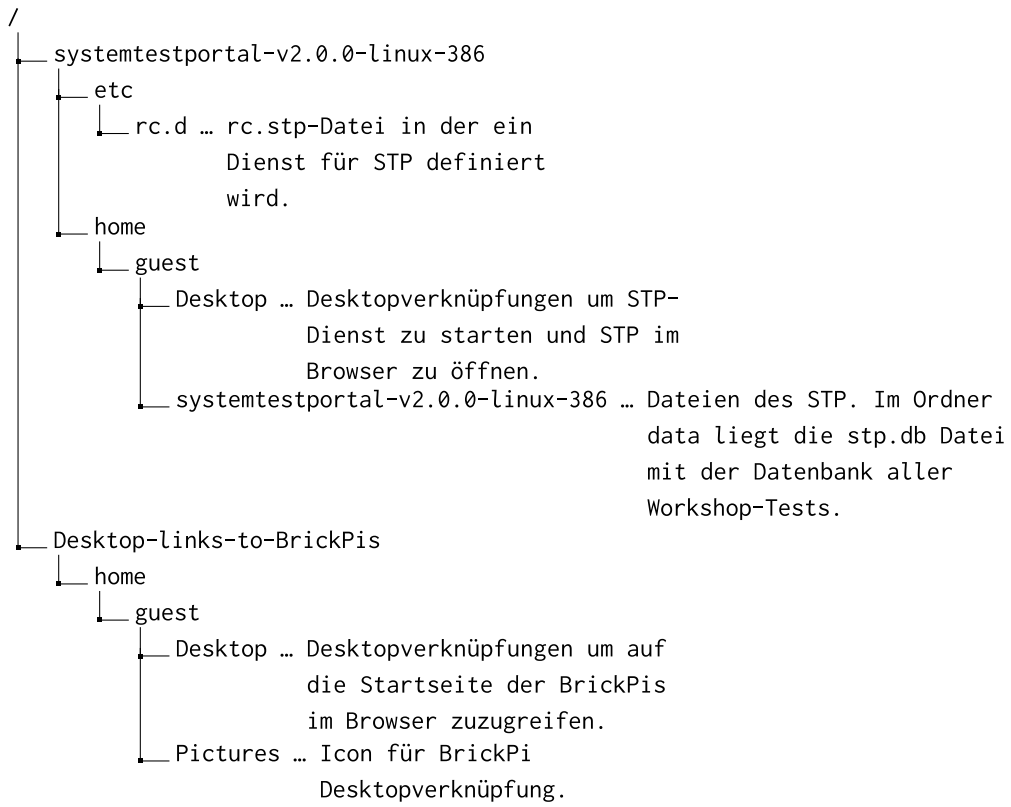


Abbildung 4.2.: Struktur der für den Workshop erstellten Porteus *.xzm-Module* für das STP und die Desktoplinks der BrickPis

wie sie später auch im Dateisystem sein sollen und anschließend über das Kontextmenü zu einem *.xzm-Modul* gepackt. Abbildung 4.2 zeigt die Dateistruktur der für den Workshop entwickelten Module `systemtestportal-v2.0.0-linux-386` und `Desktop-links-to-BrickPis`.

4.2. Bau und Programmierung der Roboter

Der große Vorteil der verwendeten LEGO Mindstorm bzw. Dexter Industries BrickPi Robotern ist, dass sie enorm flexibel sind und quasi in beinahe jeder erdenklichen Weise zusammengebaut werden können. Dieser Vorteil kann natürlich auch als Nachteil aufgegriffen werden, denn welches ist der jeweils richtige Aufbau? Als Lösung auf diese Fragestellung wird ein möglichst einfacher Aufbau der Roboter vorgeschlagen, der natürlich jederzeit erweitert werden kann, z. B. um erweiterte Tests oder Programmieraufgaben einzuführen.

4.2.1. LEGO Mindstorm NXT und EV3

Für die LEGO Mindstorm Roboter der Typen EV3 und NXT wird als Basis ein Fahrgestell des EV3 [LEG15] genutzt. An dieses Fahrgestell werden je nach Bedarf Sensoren angebracht. Für den in Abschnitt 5.4 beschriebenen Workshop werden der Farbsensor (Anschluss 2) und der Ultra-

schallsensor (Anschluss 4 (NXT) bzw. Anschluss 3 (EV3)) angebracht. Bei den Sensoren ist zu beachten, dass diese nicht abwärtskompatibel sind, der NXT also mit Sensoren für den EV3 nicht funktioniert.

Für den bereits nach dieser Ausarbeitung durchgeführten Workshop (siehe Kapitel 6) wurden die Mindstorm Roboter entsprechend Abbildungen 4.3 und 4.4 zusammengebaut. Die auf den Robotern geladenen Programme sind in Abbildungen 4.5 für den NXT und in Abbildungen 4.6 für den EV3 dargestellt.

4.2.2. Dexter Industries BrickPi

Beim Bau des Dexter Industries BrickPi gibt es wesentlich mehr Freiheiten, was allerdings auch dazu führt, dass sich der Bau komplizierter gestaltet. Beim Bau eines Roboters auf Basis des BrickPi ist zu beachten, dass der Schwerpunkt des Roboters möglichst tief liegt um diesen zu stabiliseren, was eine Erkenntniss aus dem in Abschnitt 2.2 beschriebenen Workshop ist. Das schwerste Element des BrickPi ist die Stromversorgung (8 AAA-Batterien), welche möglichst tief im Roboter verbaut, aber gleichzeitig leicht zugänglich bleiben muss. Hierzu werden zwei Anleitungen [Dex13a; Dex13c] herangezogen und dann den Workshopbedürfnissen entsprechend angepasst, was zum in Abbildung 4.7 dargestellten Ergebnis führt.

4.3. Netzwerkinfrastruktur

Der Aufbau einer Netzwerkinfratruktur hört sich komplizierter an, als er tatsächlich ist. Die Programmierung der Dexter Industries BrickPis erfolgt über eine VNC Verbindung, weshalb diese entweder über ein Netzwerkkabel direkt mit einem Endgerät verbunden werden müssen oder besser über einen Wifi-Accesspoint verbunden sind. Der Vorteil der Wifi-Verbindung liegt auf der Hand: Die BrickPi Roboter können programmiert werden ohne zwischen Endgerät und Roboter bei jedem Programmiervorgang eine Kabelverbindung herzustellen.

Eine empfohlene Netzwerktopologie ist in Abbildung 4.8 dargestellt. Für die Verbindung wird ein IPv4 Netzwerk aufgebaut, bei dem die im Workshop genutzten und mit dem Netzwerk verbundenen Robotern eine feste IP Adresse zugewiesen bekommen. Wie den Robotern eine feste IP-Adresse zugewiesen wird, kann der Anleitung des Access-Points entnommen werden. Sollten andere als die in Kapitel 6.1 genutze Hardware eingesetzt werden müssen die Desktoplinks aus Abschnitt 4.1.3 angepasst werden.

Die Nutzung einer Netzwerkinfrastruktur bietet weitere Möglichkeiten, wie der Workshop weiter ausgebaut werden kann, worauf in Kapiteln 7.1 und 7.3 eingegangen wird.

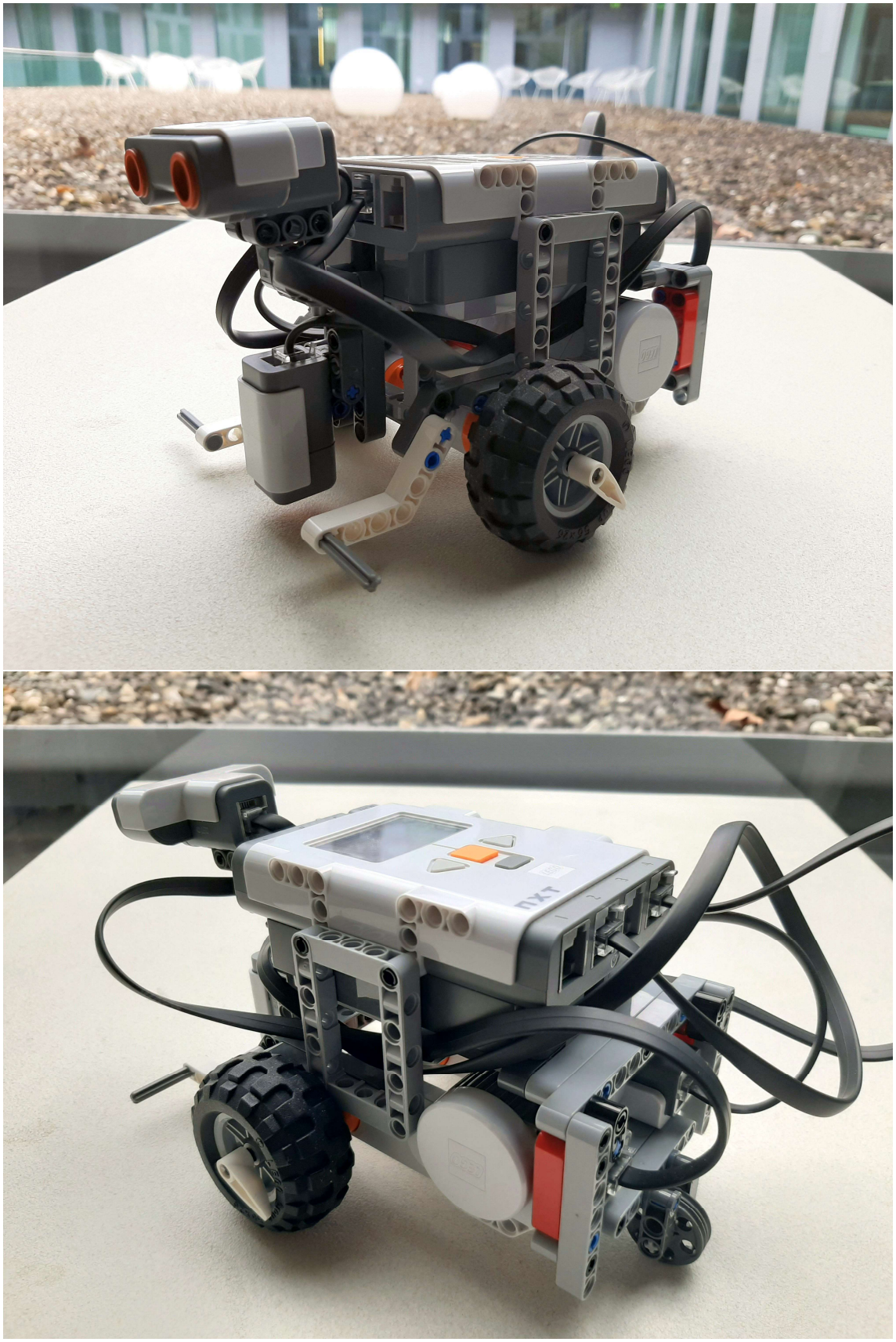


Abbildung 4.3.: Möglicher Aufbau eines LEGO Mindstorm NXT Roboters

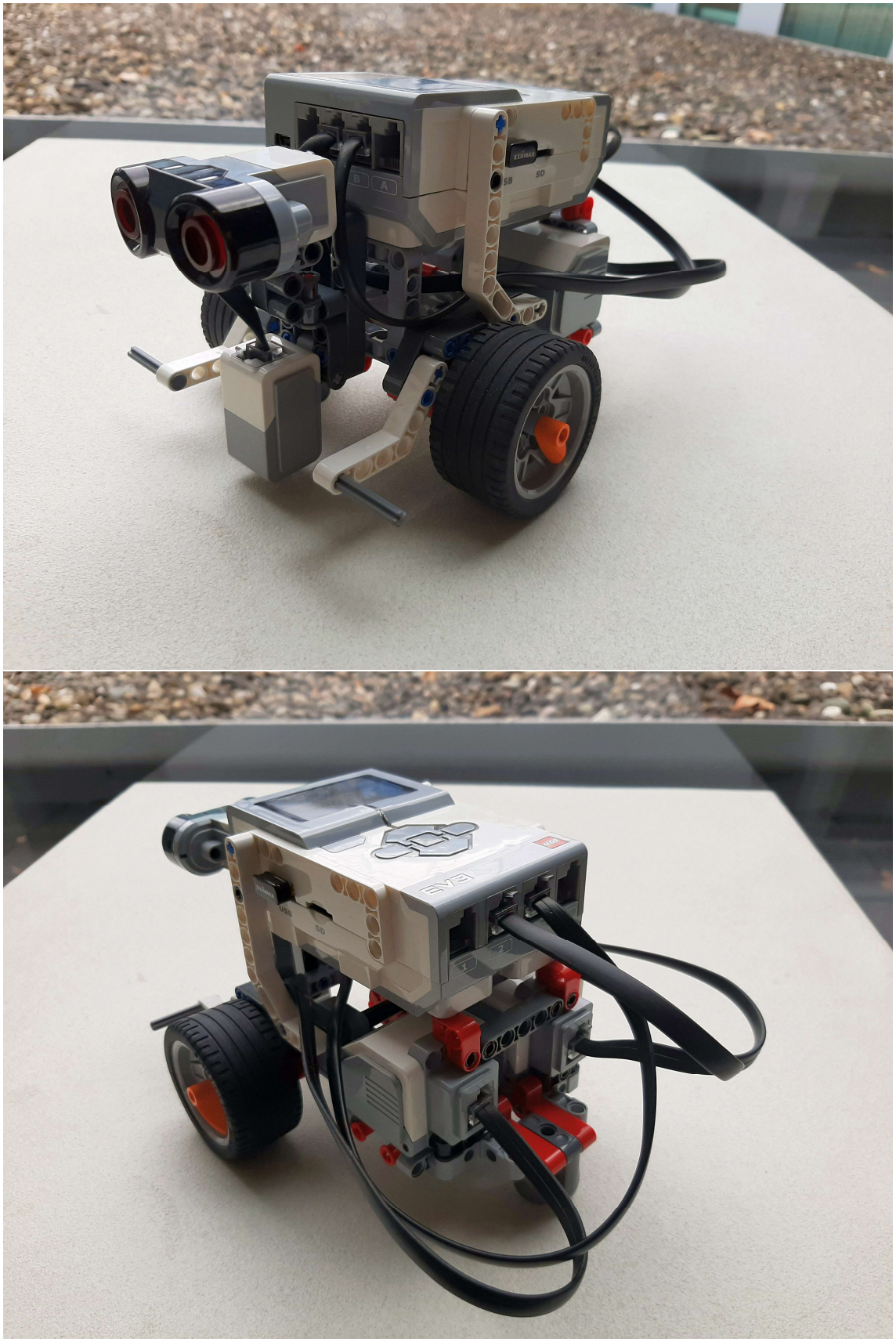
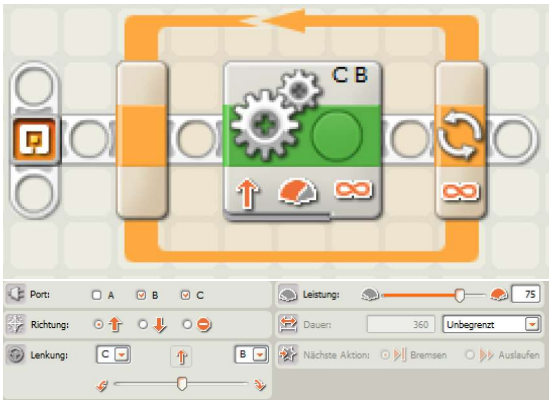
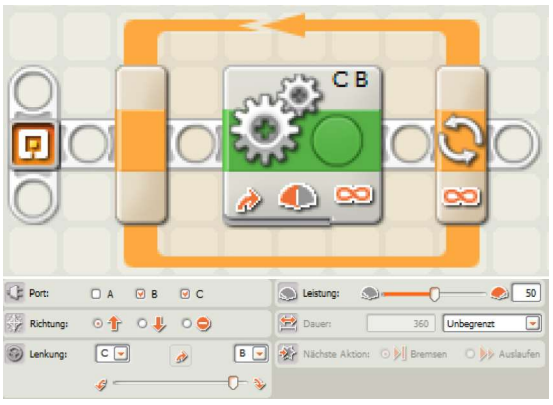


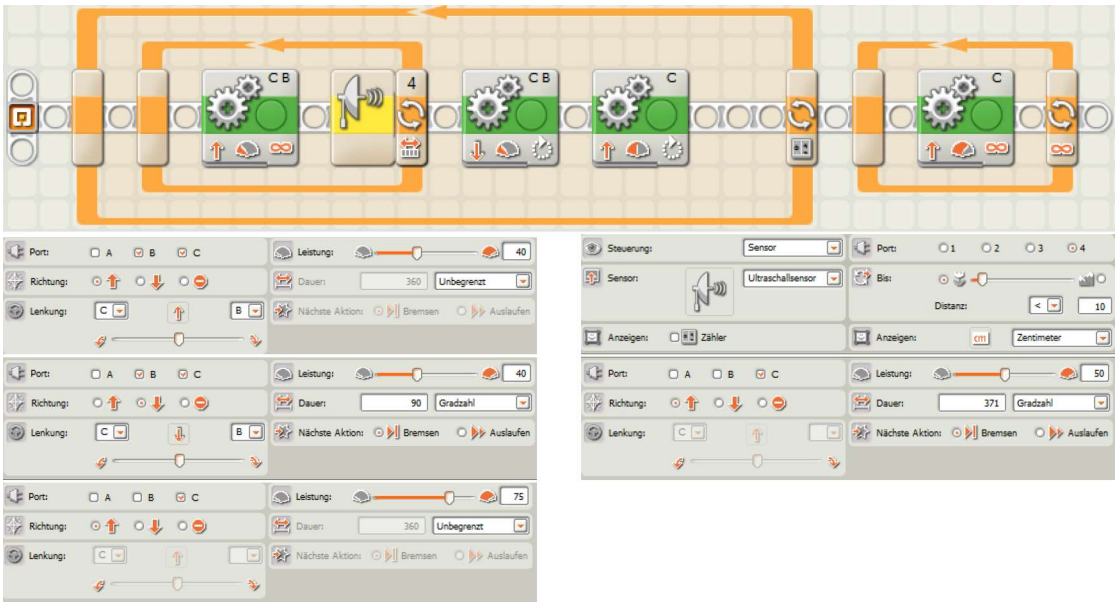
Abbildung 4.4.: Möglicher Aufbau eines LEGO Mindstorm EV3 Roboters



(a) NXT Programm für den Workshop, mit welchem der Roboter geradeaus fährt.

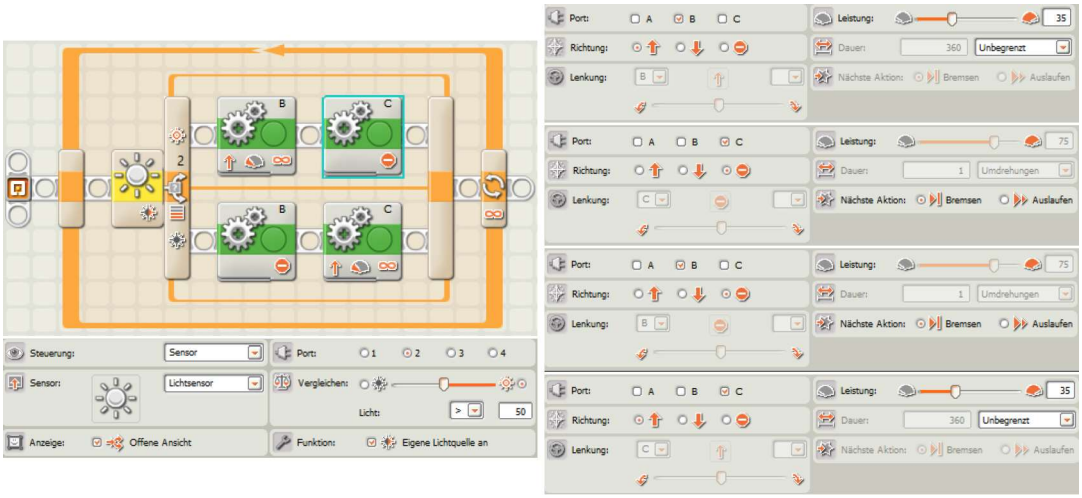


(b) NXT Programm für den Workshop, mit welchem der Roboter im Kreis fährt.



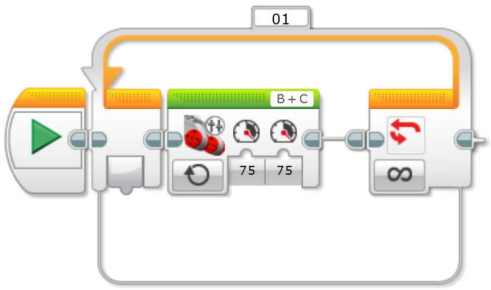
(c) NXT Programm für den Workshop, das die Reaktion des Roboters auf durch den Ultraschallsensor erkannte Hindernisse regelt.

Abbildung 4.5.: Programme auf dem LEGO Mindstorm NXT Roboter, die während dem Workshop im STP gestestet werden.

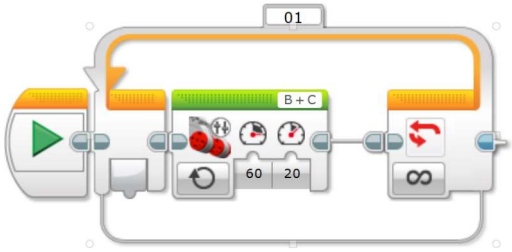


(d) NXT Programm für den Workshop, das dem Roboter erlaubt mittels Farbsensor einer schwarzen Linie zu folgen.

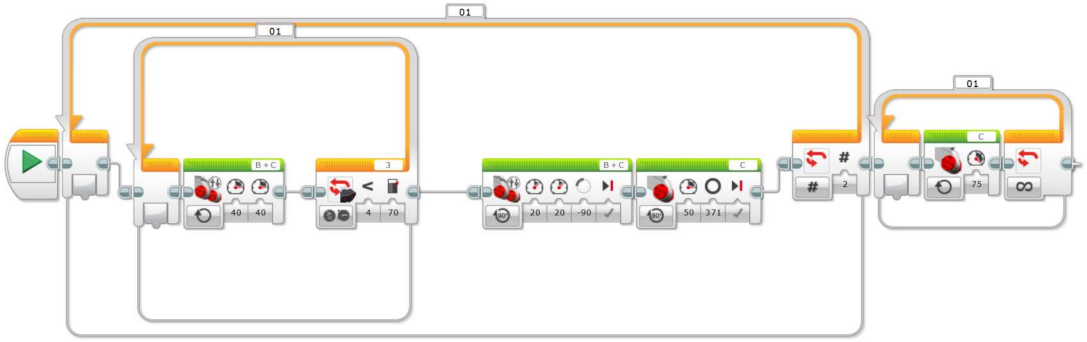
Abbildung 4.5.: Programme auf dem LEGO Mindstorm NXT Roboter, die während dem Workshop im STP gestestet werden.



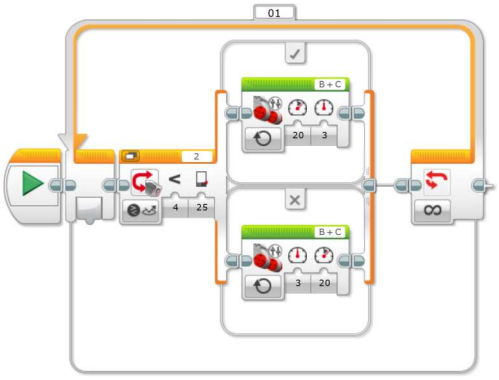
(a) EV3 Programm für den Workshop, mit welchem der Roboter geradeaus fährt.



(b) EV3 Programm für den Workshop, mit welchem der Roboter im Kreis fährt.



(c) EV3 Programm für den Workshop, das die Reaktion des Roboters auf durch den Ultraschallsensor erkannte Hindernisse regelt.



(d) EV3 Programm für den Workshop, das dem Roboter erlaubt mittels Farbsensor einer schwarzen Linie zu folgen.

Abbildung 4.6.: Programme auf dem LEGO Mindstorm EV3 Roboter, die während dem Workshop im STP gestestet werden.

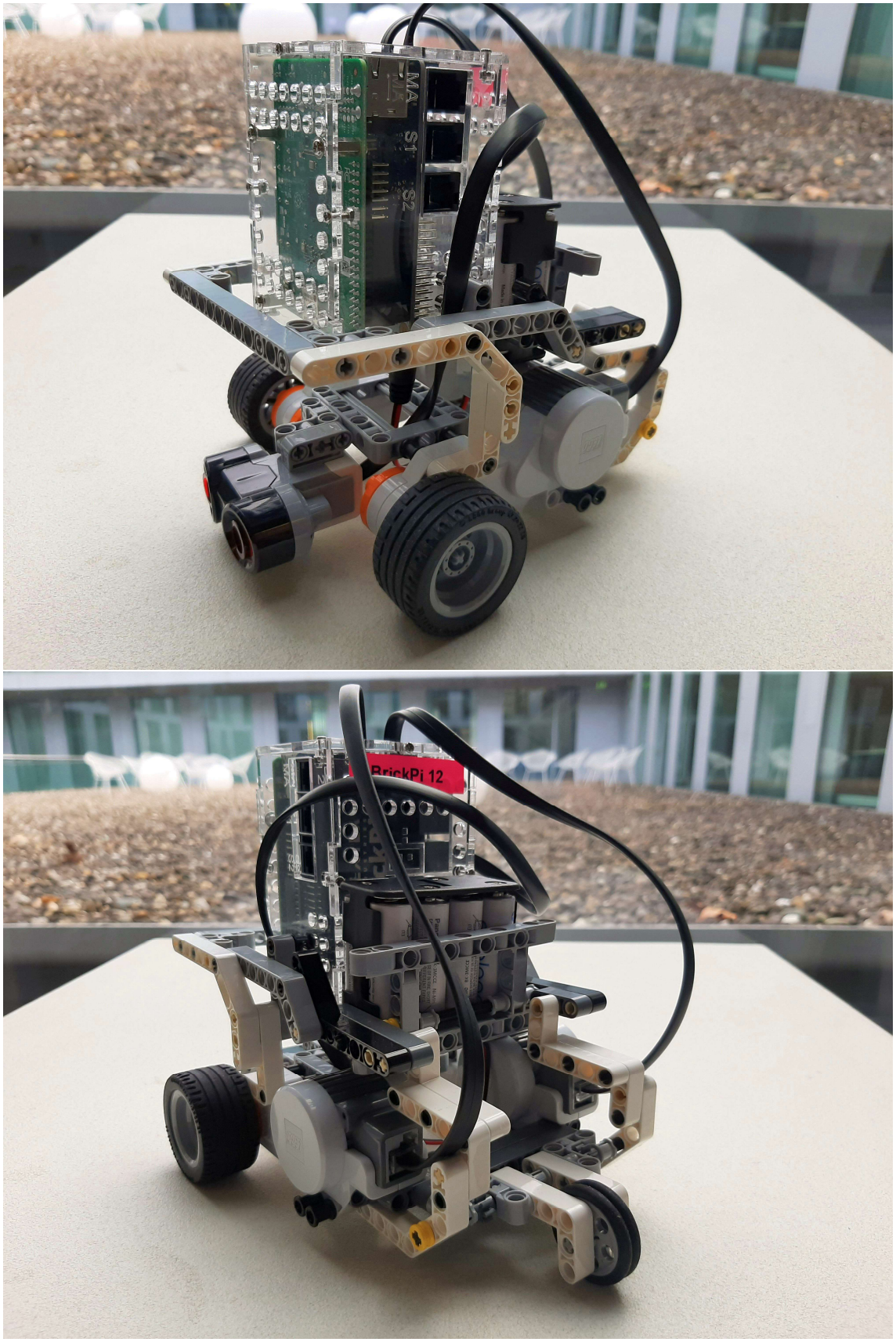


Abbildung 4.7.: Möglicher Aufbau eines Dexter Industries BrickPi Roboters

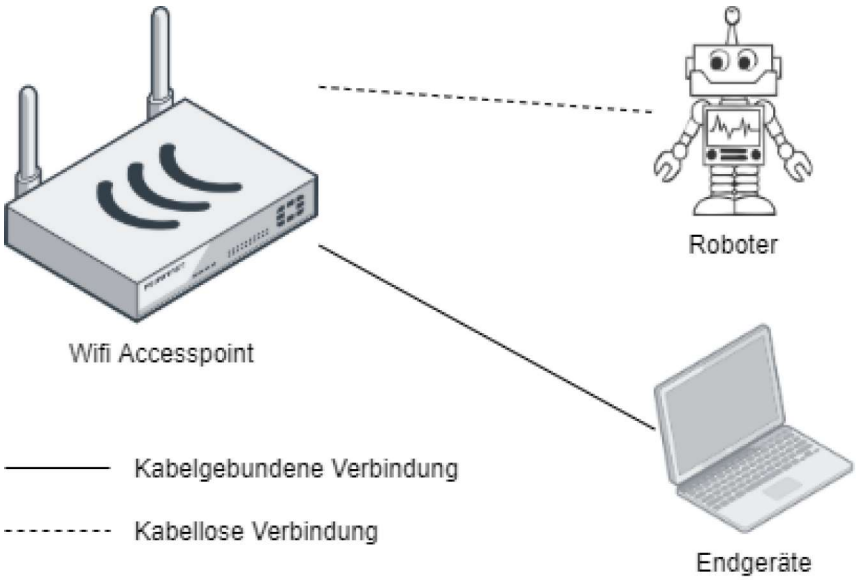


Abbildung 4.8.: Netzwerktopologie für einen Workshop

5. Workshop

Der zu entwickelnde *modulare Workshop für Themen aus dem Software Engineering* wird basierend auf den in Kapitel 3 beschriebenen didaktischen Theorien und deren Weiterentwicklungen aufgebaut. Zuerst werden die allgemeinen als auch die konkreten Workshopziele definiert um dann die Zielgruppe des Workshops genauer zu betrachtet. Nach der Definition der Workshopmodule wird in Abschnitt 5.4 ein kompletter Workshopablauf dargestellt, welcher im Rahmen des Informatiktag 2020 abgehalten und evaluiert wurde. Auf die Evaluation selbst wird in Kapitel 6 eingegangen.

Die aktuelle Fassung des Workshops ist für berechtigte Personen künftig im Gitlab-Account des STP-Teams verfügbar [STP].

5.1. Definition der Workshopziele

Pädagogen sprechen von Leit-, Richt-, Grob- und Fein(lern)zielen, wobei Leitziele die höchste Ebene darstellen. Die Ziele sind von Leit- zu Feinlernzielen immer spezifischer definiert. So sind Leitziele z. B. in der Verfassung oder anderen übergeordneten rechtlichen Grundlagen verankert, Richtziele auf Lehrplaneben, während sich Grob- und Fein(lern)ziele auf Unterrichtseinheiten bzw. einzelne Unterrichtselemente beziehen [The14].

In diesem Sinne sind die Richtziele für den zu entwickelnden Workshop die Ziele des Workshopanbieters, also die Abteilung für Software Engieneering. Im Gespräch mit Frau *Verena Ebert*, Vertreterin der Abteilung für Software Engineering, wird als Ziel genannt, bei den TuT Interesse am Studium der Softwaretechnik oder zumindest einem informatiknahen Studiengang zu wecken. Dieses Ziel ist sehr allgemein und hat die Funktion eines Leitgedanken, einer Vision, welche durch Grob- und Fein(lern)ziele untermauert werden muss.

Die Grob- und Fein(lern)ziele präzisieren das Richtziel und brechen die eher abstrakte Vorstellung auf konkret abprüfbare Ziele herunter. Mit Rückgriff auf die Ansätze und Methoden aus Abschnitt 3.3 geht es darum, dass die Teilnehmer etwas lernen und beim Lernen Spaß haben, am besten, indem sie selbst experimentieren und entdecken dürfen. Mit Hilfe der in Anhang A.5 beschriebenen Operatoren werden die Grob- und Fein(lern)ziele für den Workshop wie folgt definiert (die röm. Zahlen in den Klammern stehen für den AFB):

1. TuT können die Rolle von systematischen Tests in der Softwareentwicklung erklären (I, II).
2. TuT können einfache Programme für LEGO Roboter implementieren (II).
3. TuT können die Notwendigkeit systematischer Tests im Rahmen einer informationstechnologischen Implementierung begründen (II).

5.2. Analyse der Zielgruppe

Mit dem hier konzipierten Workshop wird in erster Linie die Intention verfolgt SuS für Themen der Softwaretechnik zu begeistern, wie es bereits im vorangegangenen Kapitel beschrieben wird. Ob nun die einzelnene Module in einem vollständigen Workshop oder z. B. auf Schülermessen eingesetzt werden ist dabei irrelevant, denn die Zielgruppe bleibt dieselbe.

Um SuS rechtzeitig in der Wahl ihres beruflichen bzw. akademischen Werdegangs zu unterstützen sollte diesen möglichst frühzeitig Entscheidungshilfen angeboten werden. Die Inhalte dieses Workshops werden primär auf SuS der Sekundarstufe II angepasst, also junge Menschen ab 15 Jahre, da ansonsten der didaktische Aufwand die Inhalte für jüngere Altergruppen anzupassen zu groß wäre. Hinzu kommt, dass SuS der Sekundarstufe II mit hoher Wahrscheinlichkeit eine Hochschulzugangsberechtigung erlangen und somit als künftige Studenten der Softwaretechnik in Frage kommen.

Neben der Betrachtung des Alters und der Schulart der Zielgruppe sollte auch der Grad der Vorkenntnisse betrachtet werden. Für den Workshop sind insbesondere die SuS interessant, die bereits ein gesteigertes Interesse an Informatikthemen und daher höchstwahrscheinlich auch Vorkenntnisse haben. Ziel der Workshopentwicklung ist es dennoch durch Differenzierung sowohl Personen mit ausgeprägten als auch Personen ohne Vorkenntnisse eine positive Erfahrung zu bieten. Insbesondere im Hinblick auf meine Bewertung des *Workshops: SmartHome* in Kapitel 2.1 denke ich, dass dies sinnvoll, sogar notwendig, ist. Allerdings sollte bei allen teilnehmenden Personen ein grundlegendes Interesse an Informatikthemen bestehen.

Zusammenfassend ist die primäre Zielgruppe für den Workshop SuS der Sekundarstufe II an allgemeinbildenden und beruflich-technischen Gymnasien.

5.3. Workshopmodule

In diesem Kapitel werden die einzelnen Workshopmodule beschrieben, die in dieser Form zusammen oder einzeln eingesetzt werden können. Die Module sind in ihrer Ausgestaltung sehr flexibel, die Bearbeitung eines Moduls kann 30 Minuten oder auch mehreren Stunden in Anspruch nehmen, sie können sowohl für die Nutzung an einem Messestand, als auch in einem zusammenhängenden Workshop individualisiert werden. Beim Einsatz der Module im Rahmen eines Workshops ist es unbedingt erforderlich den TuT vorab in einer Einführung (Abschn. 3.3.3) die Hintergründe näher zu bringen und sie in ein Lernszenario (Abschn. 3.3.4) zu versetzen.

Einige Ideen zur Anpassung der Module werden bereits in diesem Kapitel beschrieben, wobei es noch viele weitere Möglichkeiten gibt, auf die in Kapitel 7 eingegangen wird.

5.3.1. Modul: Systematisches Testen im SystemTestPortal

Dieses Modul ist induktiv aufgebaut, d.h. vom Einzelfall hin zu allgemeingültigen Regeln, in diesem Sinne also vom individuellen Test zum Verständnis von Testsystematik, und nutzt das *SystemTestPortal*, welches von der Abteilung für Software Engineering [Kul+19] entwickelt wird.

Die TuT sollen am Portal selbst erfahren, wie systematisches Testen in der Softwareentwicklung abläuft, welche Probleme hier auftreten können und dass das Formulieren von Ergebnissen häufig schwerer ist als angenommen.

Die Induktivität zeigt sich darin, dass die TuT den Einzelfall eines Tests betrachten und sie durch die Durchführung der Testschritte die Abhängigkeiten und Einflussfaktoren beim *systematischen Testen* optimalerweise erfahren und erkennen [Ott11]. Jeder TuT bzw. jede Kleingruppe (2-3 Personen) erhält Zugang zu einem Endgerät mit gestartetem Betriebssystem und erhält einen LEGO Mindstorm Roboter (NXT oder EV3). Sie beginnen anschließend entsprechend einem Arbeitsblatt (siehe Abbildung A.2) das STP zu öffnen und bearbeiten eine vorgegebene Testsequenz (aufgrund von technischen Besonderheiten gibt es für die Robotertypen NXT und EV3 verschiedene Testsequenzen, jedoch sind diese inhaltlich identisch).

Die Testsequenz besteht aus 6 Testfällen, ist nach den Prinzipien der didaktischen Reduktion aufgebaut und führt die TuT zuerst an das Arbeitsgerät, den Roboter und dessen Funktionalität, heran. Die Testsequenz besteht aus folgenden Testfällen:

1. Grundfunktionen testen
    - Die TuT testen, ob sich der Roboter einschalten lässt und ob die Tasten sowie die Menünavigation funktionieren.
2. Geradeaus fahren
    - Wie bei *Geradeaus fahren*, allerdings fährt der Roboter im Kreis und erfüllt nicht das erwartete Testergebnis.
3. Kreisfahrt
    - Die TuT lernen ein Programm zu starten und testen, wie der Roboter beim Auftreffen auf ein Hindernis reagiert.
4. Test Ultraschallsensor
    - Die TuT lernen die Funktion eines Sensors kennen und testen diesen in verschiedenen Szenarien.
5. Test Farbsensor
    - Die TuT experimentieren mit dem Farbsensor und testen ob dieser zuverlässig einer Linie folgen kann.
6. Kreative Tests
    - Mit diesem Test wird zur Differenzierung den TuT eine Aufgabe aus AFB III gestellt. Sie können das Gelernte aus den vorher durchgeführten Tests jetzt anwenden und darüber reflektieren.

Für die Bearbeitung der Testsequenz werden 30-40 Minuten benötigt, was die Erfahrungen aus Abschnitt 6.2 bestätigen.

Das STP bietet die Möglichkeit die Ergebnisse der Testsequenz als PDF zu exportieren, was genutzt werden sollte um die Ergebnisse am Ende des Workshops den TuT ausgedruckt mitzugeben. Das Aushändigen der selbst erstellten Ergebnisse bietet verschiedene Vorteile, so haben die TuT etwas, das sie am nächsten Schultag ihrem Lehrer und ihren Mitschülern zeigen können und es entsteht

zumindest ein wenig *echte Reziprozität*, also das Gefühl etwas zurückgeben zu wollen, wenn man etwas erhält [Adl05]. Der Verweis auf die *echte Reziprozität* mag weit hergeholt sein, stellt aber ein wichtiges Mittel dar um das Leitziel, die TuT für das Studium der Softwaretechnik zu begeistern, zu fördern.

Im Anschluss an die Bearbeitung der Testsequenz werden die Erfahrungen und Erkenntnisse der TuT beim Testen in einer offenen Runde diskutiert. Optimalerweise wird diese offene Diskussionsrunde am Ende eines Workshops abgehalten, bei dem die TuT beide hier beschriebenen Module bearbeitet haben.

5.3.2. Modul: Programmierung von Robotern

Dieses Modul ist deduktiv aufgebaut, d. h. von allgemeinen Regeln hin zu spezifischen Zusammenhängenden, also nach einem forschenden Ansatz, und ermöglicht den TuT anhand der grafischen Programmiersprache *Scratch* die BrickPi Roboter zu programmieren.

Die Wahl fiel auf diese Art der Programmierung, da so unabhängig vom Vorwissen der TuT Erfolge erzielt werden können, was der Motivation zuträglich ist. Erweiterungen und alternative Aufgaben zu diesem Modul werden in Kapitel 7 diskutiert.

Im Gegensatz zum vorherigen Modul ist dieses bewusst deduktiv gestaltet, was bedeutet, dass die TuT über das Arbeitsblatt A.3 die allgemeinen Grundlagen zur Programmierung erklärt bekommen, jedoch die Ideen, wie daraus die Aufgabe zu lösen ist, selbst entwickeln müssen.

Im Rahmen des in Abschnitt 3.3.4 vorgestellten Lernszenarios sollen die TuT den Roboter so programmieren, dass dieser entweder gesteuert oder selbstständig durch einen Parkours findet und einen Schalter auslöst. Der Aufbau des Parkours ist dem Referenten überlassen, als Schalter am Ende des Parkours kann ein LEGO Mindtrom NXT Roboter genutzt werden, auf dem das in Abbildung 5.1 gezeigte Programm gestartet ist.

Mindestziel ist es, dass jede Gruppe am Ende der Bearbeitungszeit den Roboter mittels Tastatureingaben steuern kann. Optimales Ergebnis ist, dass die Gruppen bereits mit dem Ultraschallsensor experimentiert haben und die Herausforderung von autonom fahrenden Fahrzeugen erkannt haben. Es wird nicht erwartet, dass eine Gruppe in der Lage ist, den Roboter tatsächlich den Parkours selbstständig meistern zu lassen, da hierzu der Zeitrahmen in einem regulären Workshop zu kurz ist. Das finale Ergebnis der Prgrammierung kann wie in Abbildung 5.2 aussehen.

5.4. Möglicher Workshop mit 90 - 110 Minuten

Abbildung 5.3 zeigt einen möglichen Workshopablauf, wie er auch am Informatiktag 2020 durchgeführt wurde. Dieser Ablaufplan für einen Workshop mit 90 - 110 Minuten kann jederzeit genutzt werden. Die Ergebnisse und Erkenntnisse aus diesem sind in Kapitel 6 beschrieben.

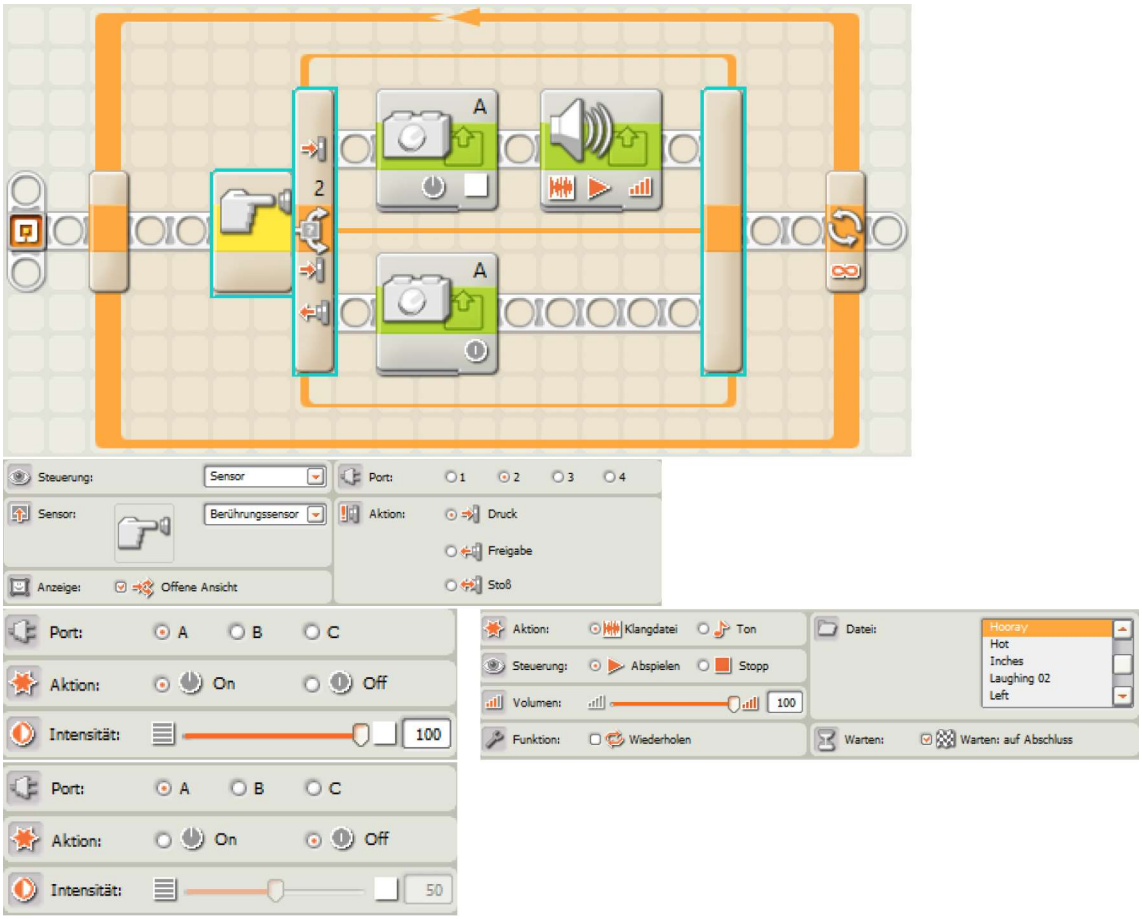


Abbildung 5.1.: NXT Programm für den Workshop, um den Roboter als Schalter am Ende des Parkours einzusetzen.

5.5. Ideen zur Nutzung einzelner Module für Messen oder andere Veranstaltungen

Beide beschriebenen Module eignen sich gut um diese auch auf Messen oder vergleichbaren Veranstaltungen einzusetzen, müssen jedoch entsprechend angepasst werden. Da im Rahmen einer solchen Veranstaltung weder eine didaktische Einführung, noch eine ausführliche Reflexionsphase / Sicherungsphase möglich ist, muss der Ablauf und Umfang der Module angepasst werden.

Das *Modul: Systematisches Testen im SystemTestPortal* (Abschn. 5.3.1) eignet sich sehr gut zu Demonstrationszwecken, allerdings empfielt es sich das Portal nicht wie in Abbildung A.2 beschrieben mit dem Zugang *tester* anzumelden, sondern einen Zugang mit erweiterten Rechten zu nutzen. Grund hierfür ist, dass dann interessierte Personen auch die Anlage von Testfällen am STP testen und erfahren können. Die Durchführung ganzer Testsequenzen eignet sich auf Messen nur bedingt, statt dessen können interessierte Personen einzelne Testfälle bearbeiten und so das STP im Anwendungsfall kennenlernen. Durch den Einsatz der LEGO Mindstorm Roboter erhöht sich die Handlungsorientierung der Präsentation, wodurch mit hoher Wahrscheinlichkeit ein höheres Interesse geweckt wird.

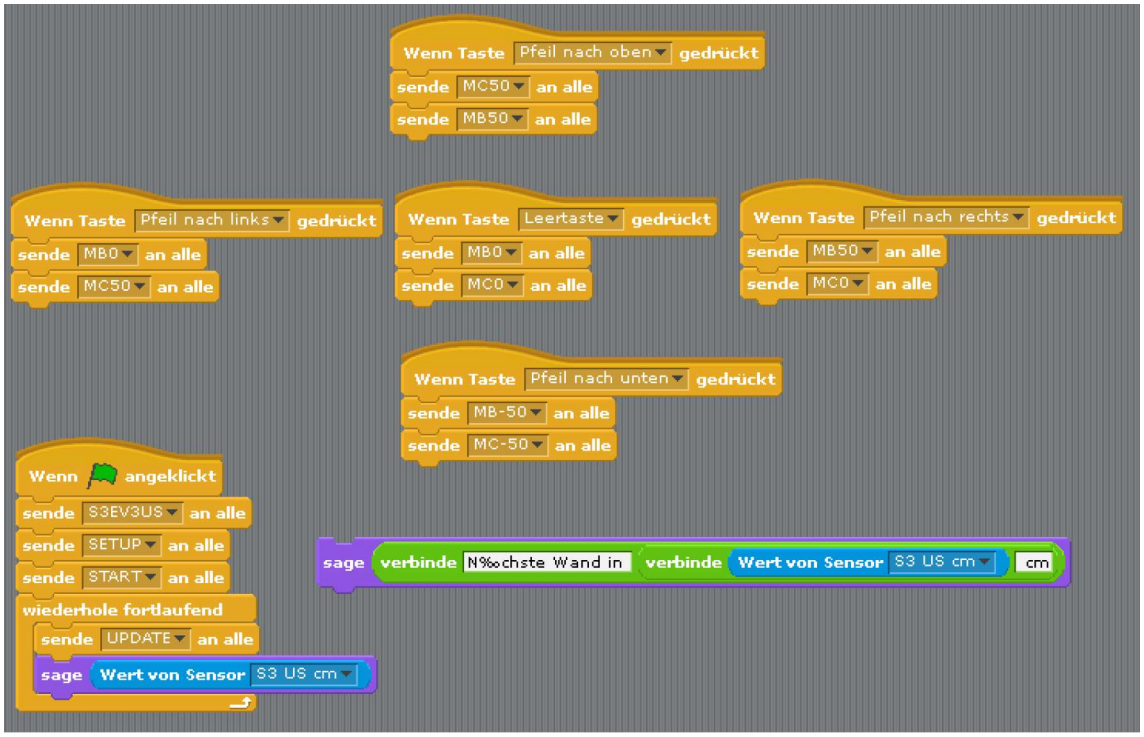


Abbildung 5.2.: Mögliches Ergebnis des *Moduls: Programmierung von Robotern*.

Beim *Modul: Programmierung von Robotern* kann das Arbeitsblatt wie in Abbildung A.3 gezeigt in identischer Form eingesetzt werden. Da ein deduktiver Ansatz gewählt wird, ist die Art und Weise, wie Interessierte den Roboter programmieren, vollkommen offen. Als Zielsetzung für die Programmierung kann auch hier vorgegeben werden, dass ein Weg gefunden werden soll, wie der Roboter z. B. über die Computertastatur gesteuert werden kann. Ergänzend zu einem Workshopeinsatz empfielt es sich beim Einsatz auf einer Messe sicherzustellen, dass sehr viele Batterien verfügbar sind. Acht voll geladene AAA-Batterien versorgen den BrickPi Roboter ca 90 - 120 Minuten mit Energie. Folglich sollten beim Einsatz von drei Robotern und einer Messe von acht Stunden mindestens 48 Batterien verfügbar sein, sollten diese parallel geladen werden können. Ansonsten empfielt es sich mindestens fünf Sätze zu je 24 Batterien (120) vorzuhalten.

Kernanliegen	Indem die TuT Tests und Programmierung an LEGO Robotern durchführen, erkennen sie die Relevanz von systematischen Tests in der Informatik und erlangen so eine Grundkompetenz im Testen bei Softwareentwicklung.
Groblernziele / Feinlernziele	1. TuT können die Rolle von systematischen Tests in der Softwareentwicklung erklären (I, II). 2. TuT können einfache Programme für LEGO Roboter implementieren (II). 3. TuT können die Notwendigkeit systematischer Tests im Rahmen einer informationstechnologischen Implementierung begründen (II).

P	R-A	TuT-A	Med	SF	Z
E	R heißt TuT willkommen, stellt sich sowie Helfer vor und führ kurz in das Thema „Systematisch Testen“ bzw. Programmierung auf LEGO Robotern ein.	TuT setzen sich in Stuhlkreis, stellen sich ebenfalls kurz vor und folgen dem Vortrag des R. TuT teilen sich nach Vortrag unter Anleitung von R und Helfer in 4-6 Gruppen ein (2er oder 3er Gruppen) und gehen anschließend an einen der vorbereiteten Arbeitsplätze.	Beamer / Monitor, Powerpoint Präsentation	UG	15
EAP	R und Helfer stehen den TuT unterstützend zur Seite, gehen auf Fragen ein und helfen bei technischen Problemen.	TuT bearbeiten die Aufgaben auf ihrem jeweiligen Aufgabenblatt: AB1: Systematisches Testen AB2: Programmierung Roboter	LEGO Roboter (NXT, EV3, BrickPi), 1 Laptop je Gruppe, Router + Kabel, AB1 und AB2	GA	35
EAP	R bzw. Helfer speichern die Ergebnisse von Gruppe A auf einen USB-Stick und starten dann den PC neu um diesen zu resetten.	TuT Gruppen wechseln die Stationen. TuT haben die Möglichkeit auf die Toilette zu gehen.	---	---	5
EAP	R und Helfer stehen den TuT unterstützend zur Seite, gehen auf Fragen ein und helfen bei technischen Problemen.	TuT bearbeiten die Aufgaben auf ihrem jeweiligen Aufgabenblatt: AB1: Systematisches Testen AB2: Programmierung Roboter	LEGO Roboter (NXT, EV3, BrickPi), 1 Laptop je Gruppe, Router + Kabel, AB1 und AB2	GA	35
SP	R stellt reflektierende Fragen zu den bearbeiteten Aufgaben mit der Zielsetzung die Relevanz von systematischen Tests in der Softwareentwicklung herauszustellen. Helfer druckt dabei die Ergebnisse der TuT-Gruppen aus Gruppe A aus, damit diese später verteilt werden können.	TuT kommen wieder im Stuhlkreis zusammen und reflektieren das Erarbeitete.	AB3, ausgedruckte Ergebnisse Gruppe A	Pl	10
	R teilt AB3 Feedbackbogen aus.	TuT beantworten AB3 Feedbackbogen.		EA	5

Legende:

GA:	Gruppenarbeit	R-A:	Referent Aktivität
E:	Einstieg	SF:	Sozialform
EA:	Einzelarbeit	SP:	Sicherungsphase
EAP:	Erarbeitungsphase	TuT:	Teilnehmerinnen und Teilnehmer
Med:	Eingesetzte Medien	TuT -A:	Teilnehmerinnen und Teilnehmer Aktivität
P:	Phase	UG:	Unterrichtsgespräch
Pl:	Plenum	Z:	Zeit in Minuten
R:	Referent		

AB1: Workshop „Systematisches Testen“ - Roboter testen
AB2: Workshop „Systematisches Testen“ - Anleitung Programmierung
AB3: Umfrage zum Workshop „Systematisches Testen autonom fahrender Fahrzeuge“

Abbildung 5.3.: Ein möglicher Ablaufplan eines Workshops von 90 - 110 Minuten. Die flexible Zeitangabe begründet sich auf die flexible Modulgestaltung. Die Dauer der Bearbeitungszeit pro Modul kann je nach Bedarf und der Rahmenbedingungen angepasst werden.

6. Evaluation

In diesem Kapitel wird das Konzept des Workshops auf zwei verschieden Arten getestet. Zuerst wird der in Abschnitt 5.4 erstellte Workshopentwurf mit einem Experten für Didaktik und Unterricht im Interview besprochen, um so mögliche Fallstricke in der Planung zu identifizieren. Zudem wird der Workshop im Rahmen des Informatiktags 2020 an der Universität Stuttgart [Roh20] durchgeführt und evaluiert. Schließlich werden die daraus gezogenen Erkenntnisse zusammengefasst und bewertet.

6.1. Experteninterview

Im Rahmen eines Experteninterviews wird der Ablaufplan (Abschn. 5.4) betrachtet und der Experte gibt seine Einschätzung ab, wo möglicherweise Schwierigkeiten auftreten und wie diesen als Referent begegnet werden kann. Didaktische Expertin für das Interview ist Frau Isabelle Karger. Sie hat Grundschullehramt an der Universität Bamberg studiert, mit der zweiten Staatsprüfung abgeschlossen und kann nunmehr auf mehr als neun Jahre Unterrichtspraxis zurückgreifen. Im Rahmen ihrer Tätigkeit ist sie zudem als Mentorin für Referendare tätig, wodurch sie Erfahrung bei der didaktischen Bewertung von Unterrichtsplanungen hat. Das Transkript des Interviews ist in Anhang 6.1 nachzulesen.

Frau Karger hat insbesondere auf die Differenzierung hingewiesen. So empfielt sie nicht nur eine Differenzierung *nach oben*, also Aufgaben in einem höheren AFB, sondern auch eine Differnzierung *nach unten*, also Aufgaben aus einem niedrigeren AFB bzw. zusätzliche Hilfestellung, anzubieten. Für den Workshop würde das bedeuten, dass für das *Modul: Systematisches Testen im SystemTestPortal* ein fertig ausformulierter Testfall bereits gestellt werden sollte. Die TuT haben die Möglichkeit ein echtes Beispiel zu betrachten um so eine Lösung nachvollziehen zu können und so eine bessere Idee davon zu erhalten, was als Ergebnis bei systematischen Tests erwartet wird.

Beim *Modul: Programmierung von Robotern* empfehle ich zwei Differenzierungen *nach unten* anzubieten: Eine Darstellung der fertigen Programmierung, wie der BrickPi Roboter über die Tastatur gesteuert werden kann, falls die TuT tatsächlich Schwierigkeiten mit der Programmierung haben. Die Lösung sollte allerdings nicht mit an den Arbeitsplatz genommen werden, sondern dort höchsten aus der Erinnerung programmiert werden. Für die Programmierung des Ultraschallsensors empfehle ich in ähnlicher Weise vorzugehen und auch hier eine mögliche Lösung bereitzustellen. Als Grundlage für die Hinweise kann Abbildung 5.2 genutzt werden. Im Rahmen der Workshopbeobachtungen beim Testworkshop fiel mir selbst auf, dass auch programmiererfahrene TuT hiermit Schwierigkeiten hatten.

Zusätzlich empfielt die Expertin von jedem technischen Gerät, also Roboter und Endgeräte, einen Ersatz bereitzuhalten, um einen reibungslosen Ablauf zu garantieren. Inwiefern dieser Empfehlung Rechnung getragen werden kann, obliegt dem Durchführenden. Alternativ kann auch die Anzahl der Teilnehmer reduziert werden um sicherzustellen, dass genug Ersatzgeräte zur Verfügung stehen.

Frau Karger konnte aufgrund des Ablaufplans (Abb. 5.3) keine weiteren Mängel oder Gefahren erkennen, die den Workshopablauf negativ beeinträchtigen könnten. Letztendlich wird es keine absolute Sicherheit geben, dass der Workshop immer gut ablaufen wird, denn auch an der Schule kann derselbe Unterrichtsentwurf in verschiedenen Klassen komplett anders laufen.

6.2. Durchführung des Workshops, Beobachtung und Teilnehmerfeedback

Um die Konzeption des erdachten Workshops zu testen wurde dieser am 14.02.2020 im Rahmen des Informatiktags 2020 [Roh20] mit 11 Schülerinnen und Schüler durchgeführt. Bei der Durchführung wurde der in Abbildung 5.3 gezeigte Workshopablauf genutzt. In diesem Abschnitt werden die Erfahrungen und Beobachtungen bei der Vorbereitung und Durchführung beschrieben sowie das Feedback der Teilnehmer betrachtet.

6.2.1. Vorbereitung

Für die Vorbereitung des Workshops wird primär auf die Checkliste in Abbildung A.1 verwiesen. Die größten Schwierigkeiten für die Durchführung des Workshops ergaben sich im Bau des Parkours für das Lernszenario *Befreiuung aus einem Escape Room mittels eines Roboters*. Die Schwierigkeit lag in der Überlegung, wie der Parkours gebaut werden sollte. Letztendlich wurde der Parkours mit Legosteinen gebaut, wobei die dafür benötigten Teile von der Fachgruppe Informatik[1] ausgeliehen wurden.

6.2.2. Durchführung

Die hier getroffenen Aussagen entstammen meiner persönlichen subjektiven Beobachtung und sollen nicht den Anschein von Vollständigkeit erwecken, sondern vielmehr eine Hilfe bei der Bewertung und Verbesserung der Workshopdurchführung darstellen.

Zu Beginn des Workshops waren, wie zu erwarten, die TuT sehr zurückhaltend und wirkten unsicher. Im Rahmen der didaktischen Einführung in den Workshop schien diese Unsicherheit nachzulassen und die Atmosphäre schien zunehmen entspannter zu werden. Ich denke, dass sowohl die lockere Art der Referenten als auch die kurzweilige Einführungspräsentation hierzu beigetragen hat. Negativ aufgefallen ist mir, dass die Übergänge bei der Präsentation teilweise etwas holprig waren, wobei die TuT das scheinbar nicht bemerkten.

Bei Beobachtungen zum *Modul: Systematisches Testen im SystemTestPortal* ist aufgefallen, dass die TuT teilweise lediglich die Checkboxen angekreuzt haben, aber die Ergebnisse nicht ausformuliert in das STP eintrugen. Auch technische Schwierigkeiten wurden beobachtet, als sich bei einem der Endgeräte das Netzteil lockerte und sich das Gerät ausschaltete. Da das System so konfiguriert ist, dass keine Änderungen gespeichert werden, waren alle bis dahin erarbeiteten Ergebnisse wieder gelöscht.

[1] https://fius.informatik.uni-stuttgart.de/

Im Rahmen von Beobachtungen zu *Modul: Programmierung von Robotern* fiel auf, dass die meisten TuT die Aufgabe begeistert bearbeiteten und auch selbstständig Ideen entwickelten, was man alles programmieren könnte. Verbesserungswürdig war, dass der mittels Scratch programmierte Ultraschallsensor die ermittelten Werte nur langsam übertrugen, was dazu führte, dass es kaum möglich war den Roboter so zu programmieren, dass er selbstständig den aufgestellten Parkour lösen konnte.

Die zeitliche Einschätzung des Workshopablaufs (siehe Abb. 5.3) passte beinahe minutengenau. Bei der Workshopdurchführung am Informatiktag fehlte allerdings noch der Test zur Differenzierung im STP (siehe Kap. 5.3.1), was dazu führte, dass manche Gruppen ca. 7 Minuten vor Ablauf der Bearbeitungszeit fertig waren.

6.2.3. Teilnehmerevaluation

Am Ende des Workshops füllten die TuT den in Abbildung A.4 dargestellten Feedbackbogen aus. Abweichend von dem o. g. Fragebogen wurde im Testworkshop zusätzlich noch die Schule abgefragt. Auf dieses Feld wird zu Gunsten einer höheren Anonymisierung verzichtet, da es in der Auswertung keinen realen Mehrwert bietet.

Von elf TuT haben neun die Evaluation ausgefüllt und zurück gegeben. Die Ergebnisse der Evaluation sind in Tabelle 6.1 dargestellt. Zusätzlich zu den rein statistischen Daten haben die Teilnehmer auch eine Menge an Freitextfeedback in Form von *Lob, Kritik, Anmerkungen* gegeben:

- Mehr Zeit zum Programmieren
- Gute Vorbereitung
- Vielleicht nächstes Mal mit „greenfoot“ arbeiten
- Mehr Zeit zum Programmieren
- Mitarbeiter waren gut vorbereitet, standen immer für Fragen und Probleme schnell zur Verfügung
- Freundliche und lustige Gestaltung des Workshops
- Arbeit mit Scratch gewöhnungsbedürftig, aber sehr guter Ansatz für die Umsetzung des Workshopzieles. (Alternativ: analoge Programmierübersicht)
- Erstellung von Tests, nicht Durchführung
- Gestaltung, Aufbau, etc. war gut
- Eventuell Themen intensiver behandeln, mehr Herausforderungen, bzw. was zum knobeln
- Zeitlicher Ablauf hat gut gepasst
- Dankeschön und alles Gute
- Scratch + BrickPi inkompatibel
- Rechtschreibfehler sind im Testprogramm vorhanden
- Es war sehr toll, dass die Betreuer entsprechende Kenntnisse haben

- Scratch sehr einfach -> andere Programmiersprachen
- Rechtscheibfehler im Testprogramm
- Veranstalter sehr nett
- Scratch + BrickPi => inkompatibel
- SSH statt VNC zum Editieren des Pi
- Zeitlicher Ablauf und kurze Einleitung war passend
- Beim Testen evtl. eigene Tests überlegen (kreative Tests)
- Beim Programmieren gute Erklärung und Aufgabe, aber evtl. ein bisschen Herausforderung für die, die das Programm schon kennen.

6.3. Erkenntnisse

Aufgrund des Interviews, der vorher dargestellten Evaluationsergebnisse und meiner persönlichen Beobachtungen ergeben sich folgende Erkenntnisse:

- Die Ausschreibung für den Workshop muss interessanter gestaltet werden, damit sich künftig mehr TuT anmelden.
- Eine weitere Differenzierung *nach oben* ist für beide Module sinnvoll. Mögliche Differenzierungen werden in Kapitel 7.1 und 7.2 beschrieben.
- Für das Modul STP ist eine Differenzierung *nach unten* sinnvoll, wie sie bereits in Kapitel 6.1 beschrieben wird, damit die TuT besser verstehen, welche Ergebnisse bei einem systematischen Test erwartet werden. Alternativ kann auch in der Sicherungsphase durch den Referent präsentiert werden, wie ein *Profi* die Aufgaben bearbeitet hat.
- Vor künftigen Workshops sollte der Referent mit Scratch experimentieren, um eine Möglichkeit zu finden, wie die Daten des Roboters so ausgelesen werden können, dass der Roboter damit tatsächlich selbst den Parkours lösen kann.
- Die Teilnehmer waren mit Scratch allgemein nur mäßig zufrieden, daher sollte Python (Kap. 7.2) oder OpenRoberta (Kap. 7.3) in Betracht gezogen werden.
- Es ist gut den TuT nach dem Einstieg selbst die Wahl zu lassen, ob sie Dreiergruppen oder Zweiergruppen bilden möchten, wobei Zweiergruppen zu bevorzugen sind.
- Je nach Gruppe (verschiedene Schulen oder alle von einer) und Workshoplänge kann es sinnvoll sein vor dem Einstieg ein Kennenlernspiel einzubauen. Ideen für Kennenlernspiele gibt es z. B. in [HT05].

Frage	T1	T2	T3	T4	T5	T6	T7	T8	T9	$\bar{x}$	s
Statistische Fragen											
Klasse	J1	J1	12	12	J1	12	J1	10	10		
Alter	17	18	18	17	17	18	17	15	15	16,89	1,17
Fragen zum Veranstaltungsort											
Der zeitliche Rahmen (Beginn und Dauer) war angemessen.	5	4	4	4	5	4	5	4	4	4,33	0,50
Der Termin ließ sich gut mit anderen Verpflichtungen des Schuljahres vereinbaren	4	5	5	5	3	5	5	5	5	4,67	0,71
Der Veranstaltungsort war gut erreichbar.	5	5	5	5	5	5	5	5	5	5,00	0,00
Die Organisation des Workshops entsprach meinen Vorstellungen und Wünschen.	5	4	3	4	5	2	5	4	4	4,00	1,00
Fragen zum Workshop											
Der Ablauf des Workshops (behandelte Themenbereiche) entsprach meinen Vorstellungen und Wünschen.	5	4,5	4	4	5	2	4	5	5	4,28	0,97
Der Inhalt wurde gut vermittelt.	5	5	4	5	5	4	5	5	5	4,78	0,44
Die in der Veranstaltung vermittelten Inhalte waren eine Bereicherung.	4	4	4	4	4	2	4	3	3	3,56	0,73
Die zuständigen Mitarbeiter/-innen waren gut vorbereitet.	5	5	5	5	5	5	5	5	5	5,00	0,00
Der Workshop hat mir gefallen.	5	5	5	5	5	3	4	5	5	4,67	0,71
Fragen zur Thematik											
Ich hatte schon Vorkenntnisse zum Thema.	4	5	5	5	5	5	5	5	5	4,89	0,33
Ich habe schon mal einen systematischen Test durchgeführt.	1	-	5	2	1	4	1	1	5	2,50	1,85
Ich war schon immer an Informatik-Themen interessiert.	3	5	5	5	3	5	5	4	5	4,44	0,88
Mein Interesse für einen informatiknahen Studiengang wurde geweckt.	5	4	1	4	4	4	4	3	5	3,78	1,20

Tabelle 6.1.: Ergebnisse der Evaluation des Testworkshops am Informatiktag 2020.
Legende: 5 = stimme voll zu, 4 = stimme etwas zu, 3 = neutral, 2 = stimme kaum zu, 1 = stimme nicht zu

- Bei der Gestaltung des Parkours sind Lego Platten genutzt worden. Optisch ist das passend gewesen, allerdings empfielt es sich für künftige Workshops einen Parkours mit einer ebenen Fläche zu bauen, da die Roboter dann besser fahren können. In der Vorbereitung habe ich ein Klebeband getestet, auf das Legobausteine gesetzt werden können. Diese Idee sollte weiter verfolgt werden, allerdings war die Verwendung des Bandes sehr herausfordernd, weshalb es mir im Rahmen der Workshopvorbereitung nicht gelang den Parkours mittels Klebeband aufzubauen. Der Entwurf des Parkours und ein Restbestand des Bandes liegt dem Institut zum Abgabezeitpunkt dieser Arbeit vor.
- Es muss vor Start des Workshops sichergestellt werden, dass alle Kabelverbindungen sicher fixiert sind, damit es hierdurch zu keinem Systemabsturz kommt. Sollte eine Verbindung locker sein empfielt es sich diese mit Klebeband zu fixieren.
- Das Aufgabenblatt *Anleitung zur Programmierung* (Abb. A.3) sollte leicht angepasst werden, unter anderem mit dem Hinweis, dass in Scratch mehrere gelbe Blöcke genutzt werden können. Zusammen mit der in Abschnitt 6.1 beschriebenen Differenzierung *nach unten* sollte es bei der Programmierung zu weniger Fragen der TuT kommen.

7. Ausblick

7.1. Erweiterung des Moduls: Systematisches Testen im SystemTestPortal

Bei der Erweiterung des *Moduls: Systematisches Testen im SystemTestPortal* blicke ich insbesondere auf die vorangegangenen Workshops, insbesondere auf den in Abschnitt 2.2 beschriebenen Workshop. Im Rahmen der Evaluation des durchgeführten Workshops (Abschn. 6.2.3) wurde angemerkt, dass die TuT gerne eigene Tests erstellt hätten. Zwar wurde dieses Ergebnis bereits in das Modul eingearbeitet, durch den Testfall *Kreative Tests*, jedoch können die Möglichkeiten noch wesentlich weiter ausgebaut werden.

Statt eines lokalen STP-Servers kann ein zentraler Server[1] direkt am Access-Point betrieben werden, welcher von allen Endgeräten erreichbar ist. Über dieses zentrale STP können die TuT sowohl Testsequenzen bearbeiten, als auch selbst Testfälle erstellen, mit denen sich anschließend andere Gruppen befassen können.

Sollte für einen Workshop 120 oder mehr Minuten verfügbar sein, kann man den Workshop auch so gestalten, dass die TuT zuerst die NXT und EV3 Roboter mittels Roberta Lab (s. Kap. 7.3) programmieren, anschließend Testfälle schreiben, welche dann von anderen Gruppen bearbeitet werden. So könnte ein wesentlich tieferer Einblick in die Systematik des Software Engineerings gegeben werden, als mit dem stark didaktisch reduzierten Workshop, der in dieser Arbeit vorgestellt wird.

7.2. Erweiterung des Moduls: Programmierung von Robotern: Programmierung mit Python

Die Dexter Industries BrickPi Roboter bieten grundsätzlich die Möglichkeit auch in Python programmiert zu werden [Dex13b]. Zusätzlich zum bestehenden Arbeitsblatt zur Programmierung (siehe Abb. A.3) kann ein weiteres für die Programmierung in Python erstellt werden, was einer weiteren Differenzierung *nach oben* entspricht. Dies bietet insbesondere den Vorteil, dass durch die Bereitstellung von Alternativen auch TuT mit mehr Programmiererfahrung eine Herausforderung haben, die sie motivieren kann.

[1] Ein zentraler Server kann z. B. mittels eines RaspberryPis aufgesetzt werden.

7.3. Alternative Programmierumgebung mittels OpenRoberta

Die Initiative *Roberta® – Lernen mit Robotern* des Frauenhofer Instituts IAIS hat zum Ziel den Nachwuchs in die digitale Welt mitzunehmen, indem SuS Roboter konstruieren und programmieren. Zu diesem Zweck wurde das Programm *Open Roberta* entwickelt, das als Image für ein RaspberryPi-System verfügbar ist [Rob19].

Der Workshop kann in der Form weiterentwickelt werden, dass die Roboter statt über Scratch durch die Nutzung eines lokalen Roberta Lab Servers programmiert werden. Die Entwicklungsumgebung von Roberta Lab ist ähnlich wie Scratch auch Drag and Drop, allerdings näher an der LEGO Mindstorm Programmierumgebung orientiert und daher auch leichter und intuitiver in der Bedienung. Unklar ist allerdings, ob auch die BrickPi Roboter mittels Roberta Lab programmiert werden können.

Die Option von Open Roberta wird hier aufgeführt, da durch die Nutzung eines Roberta Lab Servers die Wahl der Endgeräte sehr flexibel ist und von Destop PC bis zum Tablet reichen kann, was sowohl bei einem Workshop als auch auf einer Messe sehr viele Optionen bietet.

7.4. Programmierung mittels Tablets

Die LEGO Mindstorm EV3 Roboter (zum Zeitpunkt dieser Arbeit sind davon 3 am Institut verfügbar) können auch mittels „iOS- und Android-Geräte, Chromebooks und Touch-Geräte mit Windows 10“ [LEG19] programmiert werden. Dieser durchaus moderne Ansatz könnte eine Möglichkeit sein, wie kostengünstig das *Modul: Programmierung von Robotern* auch mit den EV3 Roboter durchgeführt werden kann. Die Sensoren beim EV3 funktionieren wesentlich besser als beim BrickPi.

8. Fazit

Ziel des Workshops war es junge Menschen für Themen und ggf. sogar einen beruflichen Werdegang im Bereich des Software-Engineerings zu interessieren. Ob dies im erwünschten Umfang erzielt wurde, wird die Zeit zeigen. Die Ergebnisse aus dem gehaltenen Workshop und die Evaluation stimmen jedoch positiv.

Die Überlegungen zum Workshopaufbau gingen von einem rein auf Implementierung basierenden Workshop (App-Entwicklung) bis hin zu einem reinen Roboterworkshop, der sogar den Aufbau der Roboter beinhaltet. Durch die Untersuchung der bestehenden Workshops und deren Erkenntnisse wurde der Entschluss getroffen, das bereits Vorhandene nicht völlig zu verwerfen, sondern vielmehr darauf aufzubauen.

Der entwickelte Workshop erwies sich als eine solide Basis für zukünftiges Arbeiten. Nichtsdestotrotz kann er, wie im Ausblick gezeigt, noch weiterentwickelt werden. Die Durchführung eines solchen Workshops ist auch immer an Rahmenbedingungen geknüpft, wie beispielsweise die (finanzielle) Ausstattung des Instituts oder die Sozialisation der Teilnehmer.

Literaturverzeichnis

[Adl05] F. Adloff. *Vom Geben und Nehmen - zur Soziologie der Reziprozität*. Frankfurt am Main: Campus Verlag, 2005 (zitiert auf S. 42).

[AKB14] L. W. Anderson, D. R. Krathwohl, S. Benjamin Bloom, Hrsg. *A taxonomy for learning, teaching and assessing: a revision of Bloom's Taxonomy of Educational Objectives*. Always learning. Harlow: Pearson, 2014 (zitiert auf S. 22).

[App] Apple. *SOFTWARELIZENZVERTRAG FÜR macOS Catalina*. Online. URL: https://www.apple.com/legal/sla/docs/macOSCatalina.pdf (zitiert auf S. 28).

[Aya19] A. Ayan. „Konzeption eines Anfänger-Workshops zum systematischen Testen". Bachelorarbeit. 2019. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=BCLR-2019-33&engl=0 (zitiert auf S. 16).

[Ber+06] Berlin State Institute for Schools and Media et al. *Hands across the campus : Praxishandbuch für Lehrkräfte, Schülerinnen und Schüler*. Berlin: American Jewish Committee/Berliner Landesinstitut für Schule und Medien, 2006 (zitiert auf S. 25).

[Blo73] B. S. Bloom, Hrsg. *Taxonomie von Lernzielen im kognitiven Bereich*. 3. Aufl. Beltz-Studienbuch ; 35. Institut für Leistungselektronik und Elektrische Antriebe. Weinheim [u.a.]: Beltz, 1973 (zitiert auf S. 22).

[Bog18] I. Bogicevic. *SmartHomeSimulator*. Online, Commit: 42836977e9d5f07c166fd100e01d1099b08cafed. 2018. URL: https://github.com/ibogicevic/SmartHomeSimulator (zitiert auf S. 15).

[Bra08] J.-P. Braun. *Leitfaden zur kompetenzorientierten Unterrichtsplanung. Anregung für Studierende, Lehramtsausbildung, Fortbildung, Schulaufsicht und schulische Fachgruppe*. Lengede, 2008 (zitiert auf S. 23).

[Bra12] P. Brady. *Anatomy & Physiology of an Android*. Online. 2012. URL: https://sites.google.com/site/io/anatomy--physiology-of-an-android (zitiert auf S. 24).

[Bun19] Bundesagentur für Arbeit. *Blickpunkt Arbeitsmarkt - MINT-Berufe*. Nürnberg, Aug. 2019 (zitiert auf S. 13).

[Dex13a] DexterIndustries. *Das „JJ" WiFi Auto*. Online. 2013. URL: https://edu.workbenchedu cation.com/cwists/preview/26713x (zitiert auf S. 31).

[Dex13b] DexterIndustries. *Python*. Online. 2013. URL: https://www.dexterindustries.com/BrickPi/brickpi-tutorials-documentation/program-it/python/ (zitiert auf S. 53).

[Dex13c] DexterIndustries. *SimpleBot*. Online. 2013. URL: https://edu.workbencheducation.com/cwists/preview/26769x (zitiert auf S. 31).

[Egg19] T. Eggerling. *Windows to Go! - So starten Sie Windows vom USB-Stick*. Online. 2019. URL: https://www.pcwelt.de/ratgeber/Windows-auf-dem-USB-Stick-9757702.html (zitiert auf S. 63).

[Grü78] G. Grüner. *Bausteine zur Berufsschuldidaktik*. Trier: Spee-Verl., 1978 (zitiert auf S. 23).

[Her59] D. Hering. *Zur Fasslichkeit naturwissenschaftlicher und technischer Aussagen ; eine Einführung in das Problem der Wissenschaftlichkeit und Fasslichkeit der Aussagen im naturwissenschaftlichen und technischen Unterricht.* Berlin: Volk und Wissen, 1959 (zitiert auf S. 23).

[HOS72] P. Heimann, G. Otto, W. Schulz. *Unterricht. Analyse und Planung. 6., bearb. Aufl.* Auswahl. Reihe B. 1/2. Hannover [u.a.]: Schroedel, 1972 (zitiert auf S. 21).

[HT05] B. Hugenschmidt, A. Technau. *Methoden schnell zur Hand: 66 schüler- und handlungsorientierte Unterrichtsmethoden*. Klett-Schulbuchverlag, 2005 (zitiert auf S. 50).

[HW97] A. Helmke, F. E. Weinert. „Bedingungsfaktoren schulischer Leistungen“. In: *Max-Planck-Institut für Psychologische Forschung* (1997) (zitiert auf S. 22).

[KJS14] F. W. Kron, E. Jürgens, J. Standop. *Grundwissen Didaktik. mit 36 Abbildungen und 17 Tabellen*. 6., überarb. Aufl. München ; Basel: Reinhardt, 2014 (zitiert auf S. 21).

[Kla63] W. Klafki. *Studien zur Bildungstheorie und Didaktik*. Weinheim: Beltz, 1963 (zitiert auf S. 19, 21).

[Kul] D. Kulesz. *lessons learned*. Online, Commit: d3a14a6c990487634378fe3c20cbc55323c5107e. URL: https://gitlab.com/stp-team/newcomer-workshop/-/blob/master/lessons-learned.md (zitiert auf S. 61).

[Kul+19] D. Kulesz et al. *SystemTestPortal*. Online, Commit: c43fa08a72a006276e8cb65af9f65b5189ab7053. 2019. URL: https://gitlab.com/stp-team/systemtestportal-webapp (zitiert auf S. 16, 40).

[Kul19] D. Kulesz. *stp2.0 binary does not work under windows 10*. Online. 2019. URL: https://gitlab.com/stp-team/systemtestportal-webapp/-/issues/1045 (zitiert auf S. 63).

[LEG15] LEGO Education. *Fahrgestell*. Online. 2015. URL: https://education.lego.com/de-de/support/mindstorms-ev3/building-instructions (zitiert auf S. 30).

[LEG19] LEGO Education. *Software-Varianten: PC oder Tablet?* Online. 2019. URL: https://education.lego.com/de-de/support/mindstorms-ev3/software-varianten (zitiert auf S. 54).

[Mey03] H. Meyer. „Zehn Merkmale guten Unterrichts. Empirische Befunde und didaktische Ratschläge“. In: *Pädagogik* 55.10 (2003), S. 36–43 (zitiert auf S. 21).

[Mey15] H. Meyer, Hrsg. *Leitfaden Unterrichtsvorbereitung: [alle Schulformen]*. 8. Aufl. Berlin: Cornelsen, 2015 (zitiert auf S. 24).

[Min19] Ministerium für Kultus, Jugend und Sport Baden-Württemberg. *Bildungsplan für das berufliche Gymnasium der dreijährigen Aufbauform Informatik*. Online. 2019. URL: https://zsl.kultus-bw.de/site/pbs-bw-new/get/documents/KULTUS.Dachmandant/KULTUS/Dienststellen/ls-bw/Bildungspl%C3%A4ne/Berufliche%20Schulen/bg/bg_berufsbezogen/Oberstufe/EG/BG2-AG-EG-SG-WG_Informatik_18_3992k.pdf (zitiert auf S. 24, 75).

[Net20] NetMarketShare. *Operating System Market Share*. Online. März 2020. URL: https://netmarketshare.com/operating-system-market-share.aspx (zitiert auf S. 63).

[Nic08] R. Nickolaus. *Didaktik - Modelle und Konzepte beruflicher Bildung. Orientierungsleistungen für die Praxis*. 3., korr. und erw. Aufl. Studientexte Basiscurriculum Berufs- und Wirtschaftspädagogik. Baltmannsweiler, 2008 (zitiert auf S. 19, 21–23).

[Ott11] B. Ott. *Grundlagen des beruflichen Lernens und Lehrens: ganzheitliches Lernen in der beruflichen Bildung*. Berufs- und Arbeitspädagogik. Cornelsen, 2011 (zitiert auf S. 41).

[Pet92] W. H. Peterßen, Hrsg. *Lehrbuch allgemeine Didaktik*. 3., überarb. u. erw. Aufl. München, 1992 (zitiert auf S. 19, 20).

[Por20] Porteus. *Why choose Porteus?* Online. 2020. URL: http://www.porteus.org/info/why-choose-porteus.html (zitiert auf S. 28).

[Rie01] H. Riedel. *Systemisches Modell zur Differenzierung von Lernsituationen - "Wieviel Freiraum - und wieviel Steuerung braucht der Lernende in welcher Situation?"* Online. 2001. URL: http://bidok.uibk.ac.at/library/riedel-lernsituationen.html (zitiert auf S. 25).

[Rob19] Roberta Initiative. *Open Roberta Lab Server*. Online. 2019. URL: https://www.roberta-home.de/lab/lokale-installation/ (zitiert auf S. 54).

[Roh20] D. Rohnert. *Informatik-Tag*. Online. 2020. URL: https://www.f05.uni-stuttgart.de/informatik/veranstaltungen/informatiktag/ (zitiert auf S. 15, 47, 48).

[SR15] S. Schaper, F. Rau. *Berliner Modell als Entscheidungsmodell (nach Peterßen 2001)*. Online. 2015. URL: https://de.wikibooks.org/wiki/Datei:Berliner_Modell_als_Entscheidungsmodell_(nach_Peter%C3%9Fen_2001).pdf (zitiert auf S. 22).

[SRG03] A. Schelten, A. Riedl, R. Geiger. *Lehr-Lernprozesse in einer konstruktivistischen Lernumgebung für Steuerungstechnikunterricht*. München, 2003 (zitiert auf S. 22).

[STP] STP-Team. *newcomer-workshop*. Online. URL: https://gitlab.com/stp-team/newcomer-workshop (zitiert auf S. 39, 66).

[The14] T. Thesing. *Leitideen und Konzepte bedeutender Pädagogen*. 4. Aufl. Freiburg: Lambertus Verlag, 2014 (zitiert auf S. 39).

[Vis20] Visualisierungsinstitut der Universität Stuttgart. *Informatik-Tag*. Online. 2020. URL: https://www.visus.uni-stuttgart.de/informationen-fuer/schueler/informatiktag/ (zitiert auf S. 15).

[Wei00] F. E. Weinert. „Lehr-Lernforschung an einer kalendarischen Zeitenwende: Im alten Trott weiter oder Aufbruch zu neuen wissenschaftlichen Horizonten?" In: *Unterrichtswissenschaft* 1 (2000), S. 44–48 (zitiert auf S. 22).

[Wik20a] Wikimedia Commons. *File:Operating system placement-de.svg — Wikimedia Commons, the free media repository*. Online. 2020. URL: https://commons.wikimedia.org/w/index.php?title=File:Operating_system_placement-de.svg&oldid=392040402 (zitiert auf S. 25).

[Wik20b] Wikipedia. *Liste von Linux-Distributionen*. Online. 2020. URL: https://de.wikipedia.org/wiki/Liste_von_Linux-Distributionen (zitiert auf S. 28).

Alle URLs wurden zuletzt am 27. 03. 2020 geprüft.

A. Anhang

A.1. Ausführliche Evaluation des Workshops am 24.07.2019

Die folgenden Erkenntnisse wurden von Daniel Kulesz nach dem Workshop am 24.07.2019 notiert [Kul].

Was lief gut?

- Gute Atmosphäre, Schüler machen sehr gut mit (vor allem in Anbetracht des Pflichtveranstaltungscharakters)
- Ablauf prinzipiell okay
- Zeitplan haut relativ gut hin
- Besprechung jeweils immer im Stuhlkreis sehr förderlich, verhindert ein Auseinanderlaufen des Workshops
- Ausfüllen der Fragebögen mit dediziertem Slot am Ende hat super geklappt
- Niveau scheint angemessen zu sein (für Zehntklässer aber evtl. einen Ticken zu schwer).
- Infrastruktur-Aufbau hat gut funktioniert (Booten der Rechner, Deployment, Sticks, ...), Ausnahme war nur das WLAN bei zwei Rechnern, das nicht lief.

Was könnte man noch verbessern?

- 17 Teilnehmer (8 Gruppen) grenzwertig, wir bräuchten mehr Betreuer um alle Teilnehmer adäquat zu unterstützen.
- Immer nur das aktuelle Übungsblatt austeilen, evtl. nicht doppelseitig drucken (Schüler blättern nicht um)
- Tausch der Gruppen führte zu Verwirrung da anderer Roboter, der anders bedient werden musste
- Einführungsworkshop zu kurz, ggf. zu abstrakt für Zehntklässler (besser kurze Demo mit Beamer)
- Starten der BrickPIs via dex.local zu kompliziert und zu fehleranfällig (VNC-Konsole startet nur bei Aufruf über IP-Adresse)
- BrickPIs zu instabil, haben sich zu oft zerlegt oder rumgeeiert

- WLAN-Probleme, evtl. wird soziale Komponente durch Tausch der Rechner ohnehin gestärkt (alternativ Verbindung via Crossover-Kabel, falls LAN nicht für BrickPIs benötigt werden würde)
- Abstürze (HTTP 500) im Systemtestportal, Protokoll dadurch nicht gespeichert
- Zuweisen von Labels funktioniert nicht im Systemtestportal, Aufgabe ggf. eh nicht so spannend => ggf. weglassen
- Probleme mit leeren Akkus bei BrickPIs
- Konzept der Testsequenzen evtl. zu fortgeschritten, überfordert manche Teilnehmer. Lieber nur mit losgelösten Testfällen arbeiten.
- Dokumentation wurde nicht genutzt (wir haben sie aber auch nicht beworben), es wäre aber auch kaum Zeit zum Lesen gewesen.
- Getränke für Teilnehmer bereitstellen (Zuständigkeit war unklar, und wir haben leider nichts diesbezüglich unternommen)

A.2. Begründung für die Bewertung der Betriebssysteme

Folgend die ausführliche Begründung für die Bewertung der Anforderungsanalyse in Kapitel 4.1.1:

Anforderung 1: *Einfach und flexibel*
Für beide Betriebssysteme besteht eine grafische Benutzeroberfläche, mit der Anwender intuitiv umgehen können, wobei mit Windows mehr Personen erfahren sind, da es weiter verbreitet ist als Linux [Net20]. Für Linux gibt es eine Vielzahl an verschiedenen Benutzeroberflächen, die teilweise stark voneinander abweichen. Linux ist wesentlich flexibler als Windows, was das System jedoch eher verkompliziert, weshalb Windows mit einer 2,0 und Linux mit einer 2,5 bewertet wird.

Anforderung 2: *Geschützt gegen Veränderung*
Bei dieser Anfoderung wird Linux eindeutig mit einer 1,0 bewertet, da diese Mechanik ein inhäreneter Teil der meisten Linuxdistributionen ist. In Windows kann dies auch konfiguriert werden, wobei der Aufwand hierfür wesentlich höher ist, weshalb Windows mit 2,0 bewertet wird.

Anforderung 3: *Vorkonfiguriert*
Ähnlich wie bei der vorherigen Anforderung können beide Betriebssysteme dies leisten, allerdings ist die Vorkonfiguration bei beiden Systemen mit diversen Unwegbarkeiten verbunden, weshalb in beiden Fällen mit 2,0 bewertet wird.

Anforderung 4: *Welchseldatenträger*
Beide Betriebssysteme bieten die Möglichkeit von einem USB-Stick ausgeführt zu werden, allerdings ist diese Funktionalität bei Linux in den meisten Distributionen inbegriffen, während im Falle von Windows Zusatzprogramme notwendig sind, wodurch es zu Einschränkungen kommt [Egg19]. Linux wird daher mit 1,0 bewertet, während Windows eine 4,0 erhält.

Anforderung 5: *Funktionsfähig auf alten Endgeräte*
Im Falle von Windows müsste man sich überlegen, welche Betriebssystemversion man verwenden möchte. Eine ältere Version, wie z.B. Windows XP würde auf älterer Hardware höchstwahrscheinlich gut funktionieren, könnte dann allerdings Probleme mit den Treibern für die Roboter haben. Dass ältere Windows Versionen keine Updates mehr erhalten, spielt für die Bewertung keine Rolle, da nicht vorgesehen ist, dass die Endgeräte mit dem Internet verbunden werden. Linux dagegen bietet Distributionen an, die speziell für ältere Hardware entwickelt werden und daher auch zuverlässig und flüssig auf dieser funktioniert. Windows wird so mit einer 3,0 und Linux mit einer 1,0 bewertet.

Anforderung 6: *STP*
Das Systemtestportal ist sowohl auf Windows als auch auf Linux funktionsfähig, allerdings besteht auf Windows in der aktuellen Version 2.0 ein Bug, wodurch der Pfad der Datenbank des Portals immer in das Stammverzeichnis des Laufwerkes gelegt werden muss, auf dem das Systemtestportal ausgeführt wird [Kul19]. Aufgrund dieser Einschränkung wird Windows mit 1,5 und Linux mit 1,0 bewertet.

Anforderung 7: *Programmierung*
Sowohl die LEGO Mindstorms als auch die Dexter Industry BrickPis sind unter Windows problemlos mit der Werks-Firmware programmierbar. Bei Linux hingegen ist es schwer bis unmöglich die LEGO Mindstorms ohne einer benutzerdefinierte Firmware zu programmieren, während die BrickPi Roboter tadellos funktionieren, da hier die Programmierung mittels VNC direkt auf dem Roboter stattfindet. Windows wird daher mit 1,0 und Linux mit 3,0 bewertet.

Aufgrund dieser Ausführungen kommt die Endnote für Windows (2,21) und Linux (1,64) zustande.

A.3. Transkript Experteninterview

Legende:
I: Interviewer
E: Expertin

I: Hier ist der Ablaufplan für den Workshop. Mir geht es darum, du hast ihn gerade durchgelesen, dass du einfach als didaktische Expertin hier erzählst wo du denkst, dass es vielleicht Schwierigkeiten geben könnte oder wo man besonders vorsichtig sein sollte als Referent, damit der Ablauf auch so läuft, wie er laufen soll.

E: Ich denke das größte Problem ist, dass du eine Differenzierung in alle Richtungen brauchst. Du brauchst eine Differenzierung nach unten und auch eine Differenzierung nach oben. Die Differenzierung nach oben hast du ja gemacht mit den Extra-Aufgaben, aber noch eine Differenzierung nach unten, also noch mehr Hilfestellung gegebenenfalls, die die Teilnehmer zur Hilfe nehmen können.

I: [Mehr Hilfestellung] vom Referenten her oder vom inhaltlichen? Die Referenten sind ja da, so dass die in solchen Fällen angesprochen und unterstützen können oder denkst du mehr an das inhaltliche?

E: Es ist leichter, wenn du es schon vorbereitet hast, weil du dich dann aus der Situation raus nehmen kannst und es ist wieder etwas, was der Teilnehmer selbst erarbeiten kann.

I: Hast du ein Beispiel für so etwas? Vielleicht sogar im Kontext [des Workshops] oder wenn nicht ein allgemeines Beispiel.

E: Dass du zum Beispiel irgendwo etwas hinterlegst, dass denTuT auf die Sprünge helfen kann oder eine Lösung oder ähnliches. Das muss auch nicht unbedingt am Platz sein, sondern wenn sie überhaupt nicht weiter kommen, dass sie dann vor kommen oder zu einem bestimmten Platz kommen um sich da die Lösung oder einen Gedankenanstoß, ein Impuls nochmal holen können.

I: Zum Beispiel, wenn du jetzt gerade das Programmieren nimmst, dass man quasi einen Teil von dem Programm gelöst, zum Beispiel auf Papier, wo sie anschauen können oder einen Ausschnitt davon, der ihnen weiterhilft.

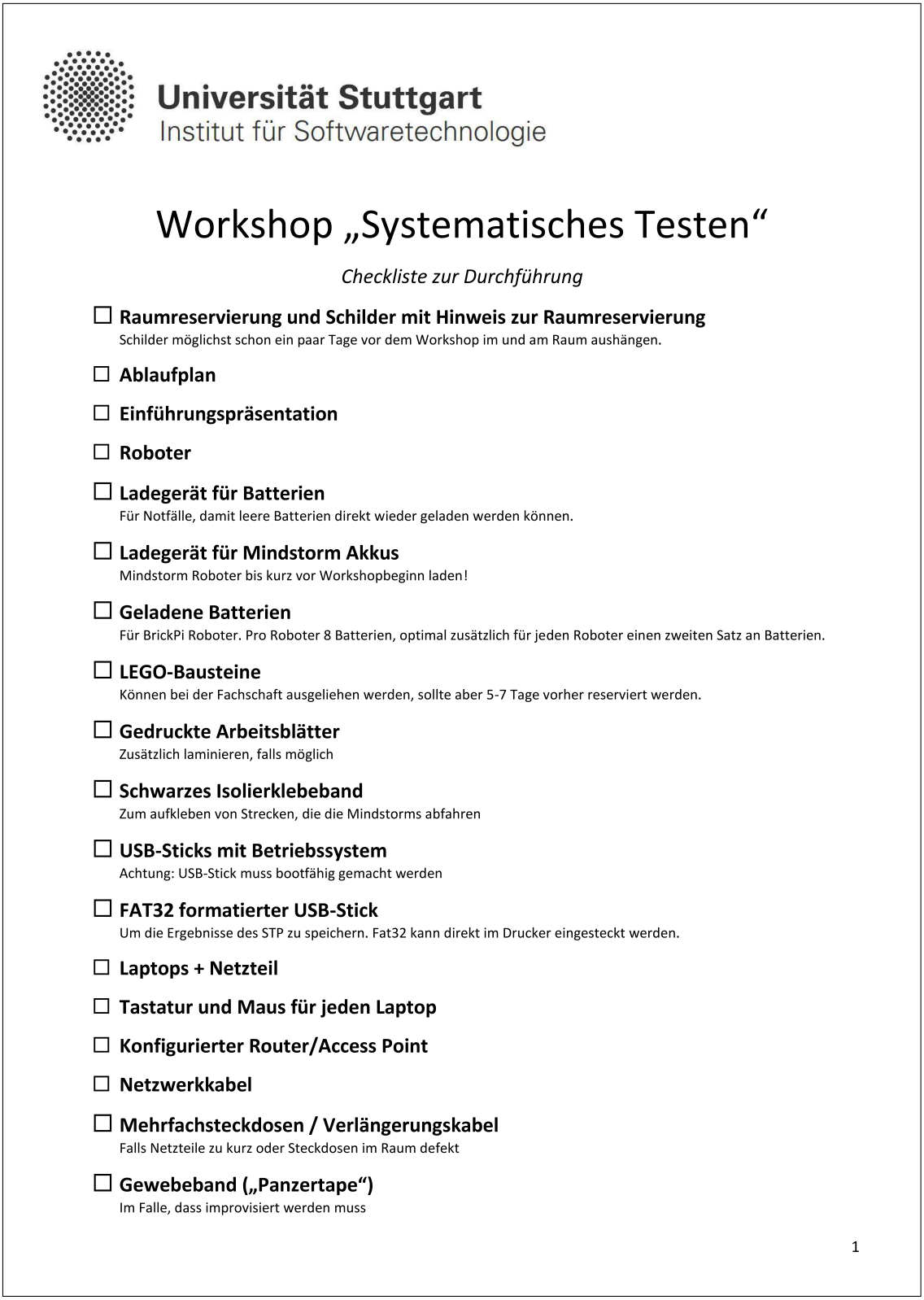
E: Ja, genau, so dass sie das dann auch nachvollziehen können und es als Anstoß nehmen können weiter zu denken und dann doch auf die richtige Lösung zu kommen.

— kurze Pause —

E: Worauf du natürlich auch immer achten must, gerade wenn du mit Technik zu tun hast oder in Prüfungssituationen, dass du noch einmal alles in der Hinterhand hast. Das heißt noch einmal genau dieselbe Anzahl von deinen kleinen Robotern in der Hinterhand, falls einer ausfällt.

A.4. Abbildungen

Die folgenden Abbildungen sind für berechtigte Personen im Gitlab-Account des STP-Teams verfügbar[STP].



Universität Stuttgart
Institut für Softwaretechnologie

Workshop „Systematisches Testen“

Checkliste zur Durchführung

☐ **Raumreservierung und Schilder mit Hinweis zur Raumreservierung**
Schilder möglichst schon ein paar Tage vor dem Workshop im und am Raum aushängen.

☐ **Ablaufplan**

☐ **Einführungspräsentation**

☐ **Roboter**

☐ **Ladegerät für Batterien**
Für Notfälle, damit leere Batterien direkt wieder geladen werden können.

☐ **Ladegerät für Mindstorm Akkus**
Mindstorm Roboter bis kurz vor Workshopbeginn laden!

☐ **Geladene Batterien**
Für BrickPi Roboter. Pro Roboter 8 Batterien, optimal zusätzlich für jeden Roboter einen zweiten Satz an Batterien.

☐ **LEGO-Bausteine**
Können bei der Fachschaft ausgeliehen werden, sollte aber 5-7 Tage vorher reserviert werden.

☐ **Gedruckte Arbeitsblätter**
Zusätzlich laminieren, falls möglich

☐ **Schwarzes Isolierklebeband**
Zum aufkleben von Strecken, die die Mindstorms abfahren

☐ **USB-Sticks mit Betriebssystem**
Achtung: USB-Stick muss bootfähig gemacht werden

☐ **FAT32 formatierter USB-Stick**
Um die Ergebnisse des STP zu speichern. Fat32 kann direkt im Drucker eingesteckt werden.

☐ **Laptops + Netzteil**

☐ **Tastatur und Maus für jeden Laptop**

☐ **Konfigurierter Router/Access Point**

☐ **Netzwerkkabel**

☐ **Mehrfachsteckdosen / Verlängerungskabel**
Falls Netzteile zu kurz oder Steckdosen im Raum defekt

☐ **Gewebeband („Panzertape“)**
Im Falle, dass improvisiert werden muss

1

Abbildung A.1.: Checkliste zur Workshopdurchführung

Universität Stuttgart
Institut für Softwaretechnologie

Workshop „Systematisches Testen“

Roboter testen

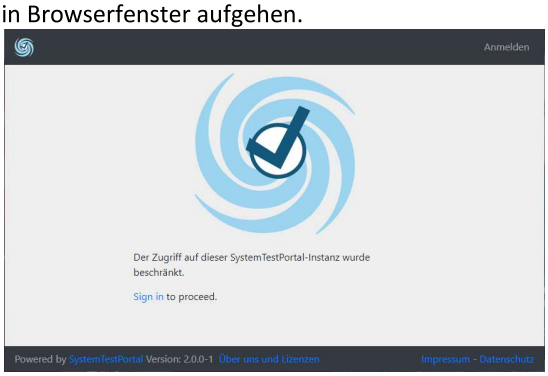
Testen in der Softwareentwicklung

Das Testen von Soft- und Hardware gehört zu den wesentlichen Bausteinen einer modernen Produktentwicklung. Das Systemtestportal bietet eine einfache Möglichkeit um manuelle Tests zu organisieren, auszuführen, zu dokumentieren und zu analysieren.

Die Ergebnisse eurer Tests und eurer Roboterprogrammierung werden wir am Ende in einer gemeinsamen Runde präsentieren.

Systemtestportal öffnen

- Klickt auf dem Desktop auf die Verknüpfung **`Starte Systemtestportal`**
- Klickt nach ca. 10 Sekunden auf die Verknüpfung **`Öffne Systemtestportal`**. Es sollte ein Browserfenster aufgehen.

→ 

- Loggt euch durch Klick auf **`Anmelden`** rechts oben mit dem Benutzernamen **`tester`** und dem Passwort **`tester`** ein.

Testsequenz durchführen

- Klickt auf das Projekt **`iste/Teste LEGO Mindstorm Roboter`**.
- Klickt rechts auf 

um den Schieberegler auf **`Testsequenzen`** umzustellen.
- Wählt die zu eurem Roboter passende Testsequenz aus. Eure Roboterbezeichnung findet ihr links unten auf der Steuereinheit (NXT oder EV3).

(Bitte wenden) →

Informatiktag 2020 1

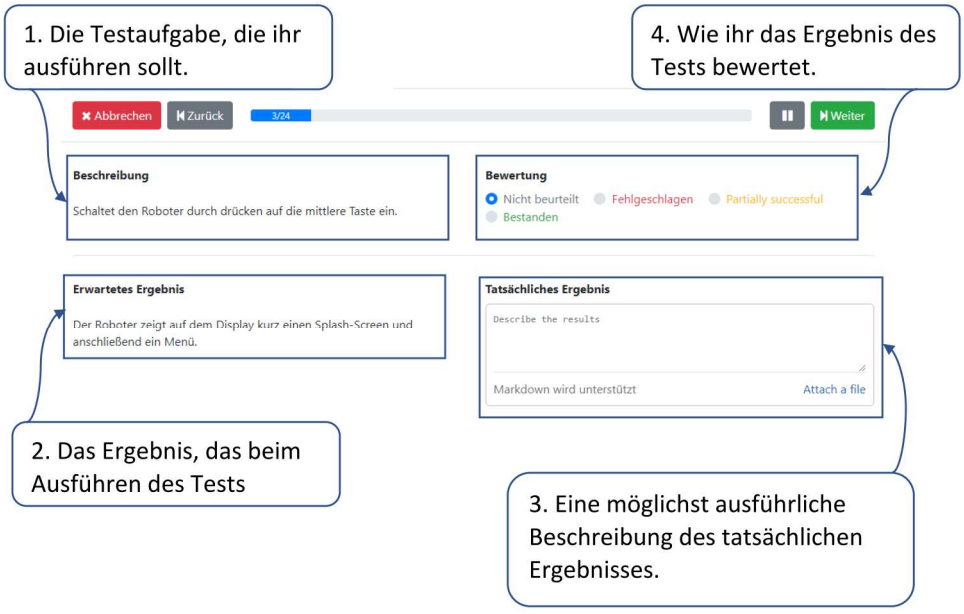
Abbildung A.2.: Aufgabenblatt 1: Systematisches Testen; führt die TuT an die Systematik des systematischen Testens heran.

- Ihr seid nun im Startfenster der Testsequenz

- Startet die Testsequenz indem ihr auf ▶ Start drückt.
- Ihr werdet nun durch mehrere vordefinierte Tests geführt:
  - Zuerst werden meist Vorbedingungen geprüft, z.B.

    ☐ Ein Lego-Mindstorm Roboter des Typs EV3 ist vorhanden und bereit getestet zu werden.

    Durch Klick auf den Button wählt ihr aus, ob die Vorbedingung erfüllt ist (grün) oder teilweise erfüllt (gelb) ist.
  - Mit einem Klick auf ⏭ Weiter kommt ihr zum nächsten Testschritt.
  - Testschritte sind wie folgt aufgebaut:



Viel Spaß beim Testen 😊

Informatiktag 2020 2

Abbildung A.2.: Aufgabenblatt 1: Systematisches Testen; führt die TuT an die Systematik des systematischen Testens heran.

Universität Stuttgart
Institut für Softwaretechnologie

Workshop „Systematisches Testen“

Anleitung für Programmierung

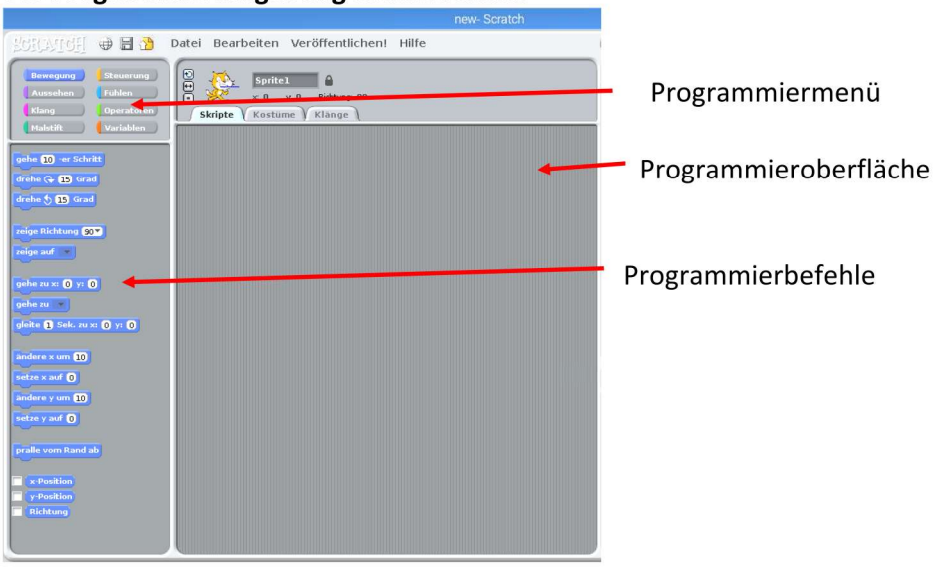
Roboter anschalten

Schaltet den Roboter unten über den Schiebeschalter ein, wartet ca. 1 Minute und ruft dann auf dem Desktop die Verknüpfung „BrickPi 11“ auf.
(Sollte die Verknüpfung nicht funktionieren, könnt ihr im Browser auch **`dex11/`** eingeben.)

Programmierumgebung starten

- Klickt im Browser auf VNC
- Öffnet die Programmierumgebung mit Scratch for Robots und Start Programming

Die Programmierumgebung kennenlernen



Informatiktag 2020 1

Abbildung A.3.: Aufgabenblatt 2: Programmieren; durch die Informatinen auf dem Arbeitsblatt wird den TuT die Grundlage in Scratch erklärt, so dass diese selbst die gestellte Aufgabe implementieren können.

Programmierung starten

Im Programmiermenü könnt ihr auswählen, was der Roboter machen soll. Da der Roboter fahren soll, wählt den Menüpunkt **Steuerung**.

Die Symbole mit dem geschwungenen Dach dienen als Startpunkt. Alle folgenden Symbole sind Anweisungen zur Steuerung des Roboters.

Mit der 

-Anweisung können die Motoren des Motors gesteuert werden. Statt **`READY`** wird der Name eines Motors eingetragen. Die Motoren haben die Namen **`MB`** und **`MC`**.

Baut nun das Beispielprogramm rechts. Die Zahlen hinter der Motorenbezeichnung stehen für die Geschwindigkeit. Experimentiert mit den Zahlen und drückt nach jeder Änderung die **`Leertaste`**.

Versucht nun mehrere Anweisungen selbst zu programmieren, so dass der Roboter in alle Richtungen fährt und auch wieder gestoppt werden kann. Im Dropdown-Menü, in dem **`Leertaste`** steht, könnt ihr auch andere Tasten wählen.

Falls ihr euch noch einen weiteren Vorteil verschaffen wollt, könnt ihr über 

die 

„Augen“ des Roboters nutzen, um den Abstand zum nächsten Hindernis abzufragen. Die Augen sind ein Sensor und haben den Namen **`EV3US`**. Die Distanz wird in cm. ausgegeben. Tragt zuerst den Namen ein und drückt dann das Kästchen.

Nun könnt ihr euch an den Parkour wagen.

Parkour gemeistert? Schafft ihr es den Roboter so zu programmieren, dass er selbst den Weg findet?

Abbildung A.3.: Aufgabenblatt 2: Programmieren; durch die Informatinen auf dem Arbeitsblatt wird den TuT die Grundlage in Scratch erklärt, so dass diese selbst die gestellte Aufgabe implementieren können.

Universität Stuttgart
Institut für Softwaretechnologie

Umfrage zum Workshop „Systematisches Testen autonom fahrender Fahrzeuge"

Wir würden gerne wissen, ob Euch der Workshop gefallen hat und was wir ggf. noch verbessern könnten. Deshalb bitten wir Euch, diesen Fragebogen auszufüllen. Die Resultate werden evtl. veröffentlicht, aber selbstverständlich nur in einer anonymisierten Form, die keinen Rückschluss auf Eure Person erlaubt. Herzlichen Dank für Eure Mithilfe!

Klasse	
Alter	

	Stimme voll zu	**Stimme etwas zu**	**Neutral**	**Stimme kaum zu**	**Stimme nicht zu**
Der zeitliche Rahmen (Beginn und Dauer) war angemessen.					
Der Termin ließ sich gut mit anderen Verpflichtungen des Schuljahres vereinbaren.					
Der Veranstaltungsort war gut erreichbar.					
Die Organisation des Workshops entsprach meinen Vorstellungen und Wünschen.					

	Stimme voll zu	**Stimme etwas zu**	**Neutral**	**Stimme kaum zu**	**Stimme nicht zu**
Der Ablauf des Workshops (behandelte Themenbereiche) entsprach meinen Vorstellungen und Wünschen.					
Der Inhalt wurde gut vermittelt.					
Die in der Veranstaltung vermittelten Inhalte waren eine Bereicherung.					
Die zuständigen Mitarbeiter/-innen waren gut vorbereitet.					
Der Workshop hat mir gefallen					

	Stimme voll zu	**Stimme etwas zu**	**Neutral**	**Stimme kaum zu**	**Stimme nicht zu**
Ich hatte schon Vorkenntnisse zum Thema.					
Ich habe schon mal einen systematischen Test durchgeführt.					
Ich war schon immer an Informatik-Themen interessiert					
Mein Interesse für einen informatiknahen Studiengang wurde geweckt					

(Bitte wenden)

Informatiktag 2020

Abbildung A.4.: Feedbackbogen zur Evaluation eines durchgeführten Workshops.

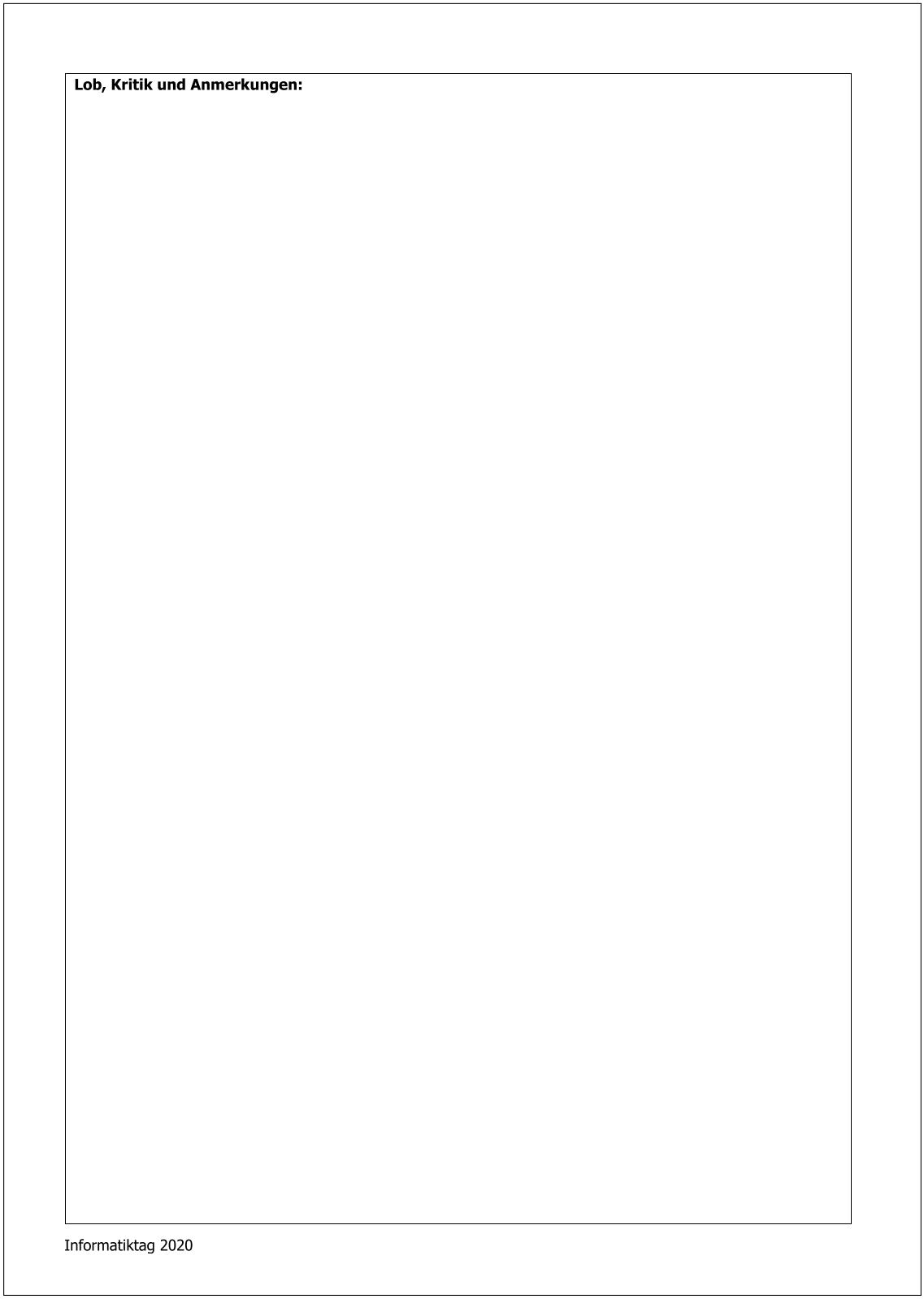


Abbildung A.4.: Feedbackbogen zur Evaluation eines durchgeführten Workshops.

A.5. Tabellen

Operator	Erläuterung	Zuordnung AFB
anwenden	einen bekannten Sachverhalt, eine bekannte Methode auf eine neue Problemstellung beziehen	I, II
auswerten	Daten, Einzelergebnisse oder Sachverhalte zu einer abschließenden Gesamtaussage zusammenführen	II, III
begründen	für einen gegebenen Sachverhalt einen folgerichtigen Zusammenhang zwischen Ursache und Wirkung herstellen	II
benennen, nennen	Sachverhalte, Strukturen und Prozesse begrifflich aufführen	I, II
berechnen	mittels charakteristischer Merkmale einen Sachverhalt genau feststellen und beschreiben	I, II
beschreiben	Strukturen, Sachverhalte oder Zusammenhänge strukturiert und fachsprachlich richtig mit eigenen Worten wiedergeben	I, II
bestimmen	einen Zusammenhang oder einen möglichen Lösungsweg aufzeigen und das Ergebnis formulieren	II, III
beurteilen	den Stellenwert von Sachverhalten oder Prozessen in einem Zusammenhang bestimmen, um kriterienorientiert zu einem begründeten Sachurteil zu gelangen	III
bezeichnen	Sachverhalte, Strukturen und Prozesse erkennen und zutreffend formulieren	I
darstellen	Zusammenhänge, Sachverhalte, Methoden etc. in strukturierter Form grafisch oder gegebenenfalls fachsprachlich wiedergeben	I, II
definieren	einen Begriff exakt bestimmen, um ihn von anderen abzugrenzen	II, III
diskutieren	zu einem Sachverhalt, zu einem Konzept oder zu einer Problemstellung eine Argumentation entwickeln, die zu einer begründeten Bewertung führen	II, III
dokumentieren	alle notwendigen Erklärungen, Herleitungen und Skizzen darstellen	II, III
einordnen	einen Sachverhalt oder eine Aussage mit erläuternden Hinweisen in einen Zusammenhang stellen	II, III
Entwerfen, planen	zusammenstellen von Funktionalitäten unter Berücksichtigung vorgegebener Daten	II, III
entwickeln	zu einem Sachverhalt oder zu einer Problemstellung ein konkretes Lösungsmodell oder ein Lösungskonzept begründend skizzieren	II, III
erklären, erläutern	Strukturen, Prozesse und Zusammenhänge von Erscheinungen erfassen, in Einzelheiten verdeutlichen und durch zusätzliche Informationen verständlich machen	I, II

Operator	**Erläuterung**	**Zuordnung AFB**
ermitteln	einen Zusammenhang oder eine Lösung finden und das Ergebnis formulieren	I, II
erstellen	Darstellen von Sachverhalten gemäß vorgegebener Syntax	II
erweitern	eine vorgegebene Struktur um Bestandteile ergänzen	II, III
identifizieren, kennzeichnen	das Wesentliche und Typische benennen	II
implementieren	Algorithmen und Datenstrukturen in eine Programmiersprache umsetzen	II
kommentieren	kausale Zusammenhänge anhand gegebener oder eigener Ergebnisse präzise vorstellen	II, III
modellieren	zu einem Ausschnitt der Realität ein informatisches Modell anfertigen	II, III
skizzieren	die wesentlichen Eigenschaften eines Objektes, eines Sachverhaltes oder einer Struktur grafisch darstellen	I, II
Stellung nehmen	unter Heranziehung von Kenntnissen differenziert eine eigene begründete Position beziehen	III
überprüfen, testen	Sachverhalte, Probleme, Fragestellungen nach bestimmen fachlich üblichen Kriterien untersuchen	II, III
übertragen	einen bekannten Sachverhalten, eine bekannte Methode auf eine neue Problemstellung beziehen	II, III
vervollständigen	Sachverhalte, Ausdrücke oder Aussagen nach bereits vorliegenden Kriterien mit zusätzlichen Informationen versehen	I, II
zeichnen	eine anschauliche und hinreichend exakte grafische Darstellung gegebener Strukturen anfertigen	I, II

Tabelle A.1.: Liste von Operatoren zur Gestaltung von kompetenzorientiertem Unterricht und deren Einordnung in AFBs nach [Min19].

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Schornhof, 27.03.2020 M. Schaff

Ort, Datum, Unterschrift