Softwaretools für kreatives Lernen im Informatikunterricht

Ralf Romeike

Didaktik der Informatik
Universität Potsdam
A.-Bebel-Str. 89
14482 Potsdam
romeike@cs.uni-potsdam.de

Abstract: Wohl kein Schulfach wird so maßgeblich durch seine Werkzeuge geprägt wie die Informatik. In diesem Artikel wird das Potential von Softwaretools zur Förderung kreativen Lernens im Informatikunterricht untersucht. Hierzu werden Kriterien beschrieben, die für Tools mit diesem Ziel gelten sollten und auf traditionelle und neue im Informatikunterricht verwendete Tools angewandt. Während traditionelle Tools im Informatikunterricht kreatives Lernen bisher kaum fördern, stellen v. a. neue Entwicklungen eine gute Arbeitsumgebung für Kreativität im Informatikunterricht bereit.

Einleitung

Wohl kein Schulfach wird so maßgeblich durch seine Werkzeuge geprägt wie die Informatik. Vermutlich existiert auch für kein weiteres Fach ein so großer Umfang an verfügbaren Tools, die sich als Einstieg in ein Thema, zur Verdeutlichung eines Sachverhalts oder als Unterrichtsgegenstand legitimieren lassen. Dabei haben diese Werkzeuge nicht nur einen starken Einfluss auf die Motivation der Schüler, sondern wirken sich auch darauf aus, inwiefern die Schüler sich kreativ im Unterricht einbringen können. Die Möglichkeit, sich kreativ zu betätigen, kann einen signifikanten Einfluss auf die Motivation der Schüler haben (vgl. [Ro08a]). Betrachtet man die Auswahl eines Tools unter diesem Aspekt, ist es notwendig zu beachten, welchen Einfluss ein Tool auf die Kreativität der Schüler nimmt. So fordert auch Diethelm [Di07], dass Kriterien entwickelt werden müssten, die für die Beurteilung einer Software unter dem Aspekt der Kreativität herangezogen werden können: Es „[...] müssten Kriterien entwickelt werden, die für Lernumgebungen und IDEs für den Unterricht gelten müssen, damit diese nicht nur die hier vorgestellten Unterrichtsmethoden unterstützen, sondern auch die Kreativität der Schüler fördern.“ (S. 156)

Ein Vorschlag für die Entwicklung solcher Kriterien soll in diesem Artikel entwickelt werden. Hierzu wird zuerst die Forschung zu Kreativität im Informatikunterricht zusammengefasst, Forschungsergebnisse zu kreativitätsfördernden Werkzeugen auf die Informatik angewandt und schließlich ein Vorschlag für Kriterien für Softwaretools für

kreatives Lernen im Informatikunterricht vorgestellt. Die Kriterien werden anschließend exemplarisch auf etablierte und neue Tools für den Informatikunterricht angewandt.

Kreativität im Informatikunterricht

Eine Leistung kann als kreativ bezeichnet werden, wenn sie zu persönlich neuen und verwendbaren Ideen, Lösungen oder Erkenntnissen führt. Mit der Informatik existiert ein Unterrichtsfach, dem kreative Prozesse immanent sind und welches nicht nur durch die Anwendung von, sondern insbesondere durch die Auseinandersetzung mit kreativen Prozessen, z. B. Modellierungstätigkeiten, die Anwendung von Kreativität notwendig macht und fördert. Der besondere Beitrag der Informatik liegt hierbei nicht in einer weiteren kreativitätsfördernden Maßnahme, sondern in der spezifischen Ausprägung, in welcher kreative Handlungsformen in Denkprozesse integriert werden und somit von Schülern in ihren Handlungsweisen zu eigen gemacht werden können. Darüber hinaus eröffnen die Werkzeuge und Methoden der Informatik nicht nur Spezialisten der Informatik, sondern auch Laien vielfältige Möglichkeiten der kreativen Betätigung. Diese kreative Beschäftigung mit Informatik an sich scheint ein Grund für viele Schüler zu sein, sich überhaupt der Informatik zuzuwenden (vgl. [KR08]). Tools und die mit ihnen erschaffenen Artefakte spielen hierbei häufig eine zentrale Rolle. Zur Gestaltung eines kreativen Informatikunterrichts wurden in [Ro07] Kriterien aufgestellt und mit dem Ergebnis, dass das kreative Potential häufiger berücksichtigt werden sollte, auf veröffentlichte Unterrichtssequenzen angewandt. Eine umfassende Betrachtung von Kreativität im Informatikunterricht erfolgt in [Ro08b]. Demnach stimuliert kreatives Arbeiten Motivation und Interesse der Schüler, im Informatikunterricht wird Kreativität aufgrund des kreativen Charakters des Fachs Informatik gefördert und IKT (Informations- und Kommunikationstechnologien) fördern Kreativität im Informatikunterricht.

Zur Unterstützung bei der Auswahl geeigneter Werkzeuge ist eine Evaluation und Kategorisierung von im Informatikunterricht verwendeten Werkzeugen hinsichtlich der Förderung von Kreativität opportun.

Softwaretools zur Förderung von Kreativität

Computer werden von Wissenschaftlern und Künstlern zunehmend als wertvolle Werkzeuge für kreative Tätigkeiten angesehen. Auch ein großer Teil der Forschung im Spannungsbereich von Kreativität und Informatik beschäftigt sich mit den Möglichkeiten zur Förderung von Kreativität durch IKT in professionellen Bereichen und Lernumgebungen [Sh05]. Diese beschränken sich nicht allein auf Möglichkeiten der Förderung von Kreativität innerhalb der Informatik, sondern erstrecken sich auf unzählige Gebiete, auf denen Software kreativitätsunterstützend eingesetzt werden kann. So sieht Trogemann [Tr01] die Aufgabe und Herausforderung der Informatik sogar v. a. darin, neue Kreativ-Werkzeuge bereitzustellen. Shneiderman [Sh07] prophezeit:

> *The current and forthcoming generations of programming, simulation, information visualization, and other tools are empowering engineers and scientists just as animation and music composition tools have invigorated filmmakers and musicians.* (S. 20)

Das impliziert eine steigende Bedeutung kreativitätsunterstützender Software und bedeutet, dass Erfindungen und Innovationen, angeregt durch kreativitätsunterstützende Werkzeuge, in noch größerem Maß als zuvor zunehmen werden. Sie ermöglichen selbst Anfängern, wie Experten zu arbeiten, und Experten, immer beeindruckendere Ergebnisse in vielfältigen Bereichen zu produzieren.

Shneiderman [Sh02] postuliert in einem Kreativitätsframework vier Aktivitäten, die durch IKT unterstützt werden können: das Sammeln und Darstellen von Information, das in Beziehung setzen zu vorhandenem Wissen, z. B. durch Unterstützung der Kommunikation, das Schaffen und Evaluieren neuer Lösungen sowie das Verbreiten der Ergebnisse (vgl. Abbildung 1). Während diese Aktivitäten die potentiellen Möglichkeiten von IKT noch relativ abstrakt beschreiben, konkretisiert Shneiderman [Sh02] die Anforderungen durch Kriterien für kreativitätsunterstützende Softwarewerkzeuge.[1]

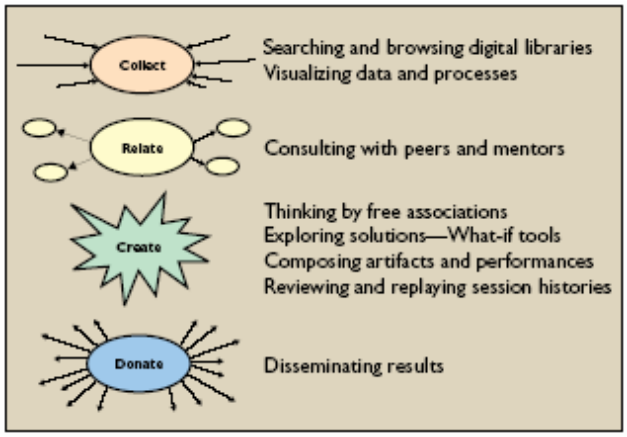

Abbildung 1: Vier Aktivitäten und Aufgaben im Kreativitäts-Framework [Sh02].

IT should support

- *pain-free exploration and experimentation*
- *immediate and useful feedback for one's actions*
- *no big penalties for mistakes, meaningful reward for success*
- *easy way to undo and redo*
- *visualizing data processes*
- *searching for knowledge and inspiration*
- *composing a work step by step*
- *disseminating results to gain recognition.*

[1] Vergleichbare Anforderungen stellen [Gr02; Lu05; RMN05; Lo07].

Diese Anforderungen lassen sich direkt auf Software beziehen, die in der Informatik und teilweise im Informatikunterricht eingesetzt wird. Die Kriterien werden im folgenden erläutert und anhand der Programmiersprache Scratch verdeutlicht.

Computerbasiertes Experimentieren und Explorieren kann zu neuen Ideen verhelfen und sollte Designen, Simulation, auch Überraschungen und die Erzeugung von Artefakten ermöglichen [MIB03]. Programmierer nutzen diese Möglichkeiten des Experimentierens, die ihnen Programmierumgebungen eröffnen, indem sie Ideen und mögliche Lösungswege austesten. Nach Glass [Gl06] stellt das Anwenden von selektivem Trial und Error einen wesentlichen Teil der Softwareentwicklung dar. Solche „Was passiert wenn"-Szenarien können schnell ausprobiert und variiert werden und sind nur durch die Unterstützung der Entwicklungsumgebung möglich. Programmierumgebungen für Programmieranfänger, wie z. B. Scratch, machen Explorieren sogar noch leichter; Operationen können einfach auf Objekten ausgeführt und das Ergebnis in Echtzeit beobachtet werden. Offensichtlich beinhaltet auch das Programmieren-Lernen das Erkunden der Programmierumgebung („Was sind meine Möglichkeiten?") und das Experimentieren mit den Objekten und Konstrukten, die anwendbar sind („Was passiert, wenn ich dieses so verwende?").

Ein sofortiges und nützliches Feedback auf ausgeführte Aktionen durch den Compiler oder Interpreter unterstützt das Experimentieren und ist eine Besonderheit der Softwareentwicklung. In anderen Bereichen geben die Werkzeuge, mit denen gearbeitet wird, entweder kein qualitatives Feedback automatisch oder können nicht auf Fehler hinweisen.[2] Bei vielen Schülern kann man beobachten, wie solches handlungsbasiertes Feedback[3] ein Gefühl von eigener Kontrolle und selbstreguliertem Lernen ermöglicht.[4]

Die von Shneiderman geforderte „Straffreiheit" wird in Softwareentwicklungsumgebungen insofern realisiert, als dass das Fehlerfeedback die größte „Strafe" darstellt. Im Gegensatz zu anderen Unterrichtsfächern, wo Feedback nur durch den Lehrer oder durch andere Schüler möglich ist, wird dieses beim Programmieren durch eine Maschine in einer nicht einschüchternden Art und Weise übernommen. Ein funktionierendes Programm wartet als Belohnung für einen erfolgreichen Programmierversuch; in Scratch wäre dies bspw. eine Animation oder ein Spiel. Unterstützend wirkt sich beim Experimentieren die Sicherheit aus, dass nichts wirklich „kaputt" gemacht werden kann. So ermöglicht Scratch auf einfachem Weg, letzte Änderungen rückgängig zu machen oder wiederherzustellen. Während diese genannten Punkte kreativen Arbeitens in anderen Fächern wenn dann nur an Simulationen möglich sind, wird beim Programmieren am Produkt selbst experimentiert.

Viele Programmier(lern)umgebungen erfüllen auch weitere von Shneidermans Kriterien, z. B. die Visualisierung von Prozessen und Daten, welche dabei hilft, Ideen und Fakten

---

[2] So müssen bspw. musikalische oder Kunstexperimente durch andere bewertet werden. Nicht-Softwarebasierte Experimente in Physik oder Chemie geben im Fall des Misserfolgs nicht an, wo die Probleme liegen.

[3] Feedback sollte nur informierenden, nicht kontrollierenden Charakter haben.

[4] Programmieren wird bekanntlich häufig sogar autodidaktisch gelernt.

zu organisieren und Beziehungen zu entdecken. Der Trend zu visueller Programmierung spiegelt diesen kreatvitätsrelevanten Aspekt wider. In Scratch werden Programmierkonstrukte (z. B. Schleifen, Operationen, Botschaften) durch bunte Bausteine dargestellt, welche die unterschiedlichen Bausteintypen durch verschiedene Farben kategorisieren. In modellierungsorientierten Ansätzen werden Datenstrukturen und Prozesse als Diagramme mit ihren Beziehungen modelliert und können automatisch in Programmtext übersetzt werden und vice versa (z. B. mit Fujaba). Der Trend zu diesem visuellen, abstrakteren Level der Programmierung verspricht, dass dies in Zukunft wichtiger, und damit auch Kreativität noch besser unterstützt wird.

In einer kreativitätsunterstützenden Umgebung ist es wichtig, Wissen einfach zugänglich zu machen und für Inspiration zu sorgen. Programmierumgebungen ermöglichen das Suchen in einem Hilfesystem und beinhalten häufig eine Dokumentation mit Kommentaren und Beispielen. Scratch bringt zusätzlich inspirierende Beispielprogramme mit, die von anderen Schülern programmiert wurden, und zusätzliche Materialien, an welchen Programmierkonstrukte erklärt und Experimentieren mit diesen Konstrukten angeregt werden.

Schrittweises Komponieren bedeutet ein langsames Herangehen an ein Problem, während noch viele Entscheidungen offen gelassen werden. Für kreatives Arbeiten ist diese Bottom-Up-Vorgehensweise wichtig, da ein Ergebnis häufig nicht klar von Anfang an bestimmt ist und sich erst später konkretisiert. In Scratch ist diese Möglichkeit gegeben, indem ein Programm zu jedem Zeitpunkt ausgeführt und überprüft werden kann.

Ein Verbreiten der Ergebnisse kann sich positiv auf die Motivation und die eigene Reputation auswirken. Ein Klassenraum bietet zwar nur begrenzte Möglichkeiten, eigene Leistungen zu präsentieren, durch das Internet können diese aber einem breiten Publikum vorgestellt werden. Scratch besitzt eine Funktion, mit der Scratch-Programme einfach auf einen Internet-Server geladen werden können. Dort können sie von anderen Lernenden eingesehen und bewertet werden und dienen ebenso zur Inspiration für andere.[5]

Abschließend betrachtet wird deutlich, dass Scratch, wie auch andere Programmierlernumgebungen, Shneidermans Kriterien für kreativitätsunterstützende Software erfüllt. Die Entwickler von Scratch hatten in so genannten Computerclubhäusern über Jahre Kinder beim Erlernen des Umgangs mit IKT beobachtet und aus ihren Erfahrungen entsprechend Prinzipien für die Entwicklung von Werkzeugen entwickelt, welche die Förderung von Kreativität explizit beinhalten.

Designprinzipien für kreative Lerntools

[5] So wie die einfache Verbreitungsmöglichkeit von Amateur-Geschichten zu einer steigenden Anzahl kreativer Literatur im Internet geführt hat, kann auch das Verbreiten von Software durch das Internet zu kreativem Tun anregen. Auf der Homepage von Scratch sind zum jetzigen Zeitpunkt über 340000 Projekte von 52000 Mitgliedern veröffentlicht und gegenseitig kommentiert.

Da Scratch mit dem Ziel entwickelt wurde, kreatives Lernen zu unterstützen, können die bei der Entwicklung von Scratch zugrunde gelegten Prinzipien auch wichtige Hinweise für die Auswahl von Tools für kreatives Lernen geben (vgl. RKM03)). Es wird deutlich, dass im Lernkontext zusätzliche Bedingungen aufgrund der Lernsituation, wie z. B. die Motivation und Interessen der Schüler, zu berücksichtigen sind. Tabelle 1 (siehe Anhang) zeigt eine Gegenüberstellung der Kriterien Shneidermans mit den Designprinzipien, die der Entwicklung von Scratch zugrunde liegen und erfahrungsbasierten Kriterien für die Entwicklung von „Construction Kits for Kids“ [RS05]. Eine Analyse der Kriterien macht deutlich, dass diese nicht nur die Kreativität der Schüler unterstützen sollen, sondern auch in Kernpunkten mit Shneidermans Kriterien übereinstimmen. Zusätzlich berücksichtigen Sie die Besonderheiten einer Lernsituation, wie Motivation und Interessen der Schüler.

Lassen sich diese Kriterien nun nur auf Programmiersprachen und Construction-Kits anwenden? [RS05] schreiben bezogen auf die Designkriterien: „While these principles apply especially to the development of construction kits, we believe that they could be useful for everyone who designs new technologies for kids.” Bzgl. der Anwendung der Kreativitätskriterien auf die Auswahl von Werkzeugen für den Informatikunterricht kann ähnliches gesagt werden: Die Kriterien sind besonders wertvoll bei der Auswahl von Modellierungstools, lasses sich aber vermutlich auch auf andere Tools anwenden.

Kriterien für Softwaretools für kreatives Lernen im Informatikunterricht

Die genannten Kriterien lassen sich hinsichtlich ihres Wirkungsbereichs in drei Kategorien einordnen und unter Berücksichtigung des *Faktorenmodells für kreatives Lernen im Informatikunterricht* (vgl. Romeike 20008) präzisieren. Auf neu hinzugekommene Kriterien wird dabei kurz eingegangen.

Arbeitsweise

In diesem Bereich werden Kriterien angegeben, die den Prozess des kreativen Arbeitens unterstützen:

Das Tool
- ermöglicht konstruktionistisches Lernen[6]
- bietet und unterstützt verschiedene *Explorations- und Experimentiermöglichkeiten*

[6] Papert greift in seiner Lerntheorie des Konstruktionismus das Prinzip des Lernens als Konstruktionsprozess auf, fügt aber die Idee hinzu, dass Lernen durch das eigenhändige Konstruieren eines Artefakts unterstützt wird [PH91]. Im Gegensatz zu Piaget hält Papert es für wichtig, dass Lernende direkt in Situationen eintauchen, statt sie von außen zu betrachten. Hierzu zählt auch die Verbundenheit mit dem Lerngegenstand: Der Schlüssel zum Lernen liegt im Einswerden mit dem betrachteten Phänomen, was wiederum zu bedeutungsvollem Lernen führt. Vorteil des Schaffens eines realen Produkts ist die Möglichkeit, dieses auszuprobieren, zu zeigen, zu diskutieren und zu bewundern [Pa93].

- erlaubtes einfaches *Undo/Redo*
- *visualisiert* Prozessabläufe und Datenflüsse
- erlaubt *schrittweises Entwickeln* und das frühe Erzeugen von *Prototypes*
- hat „high ceilings“, d. h. es ist *erweiterbar* und erlaubt unterschiedliche *Komplexitätslevel*
- unterstützt *schrittweises und inkrementelles Lernen*

Lernen durch Designen ist ein zentrales Konzept des konstruktionistischen Lernansatzes, auf dem auch kreatives Lernen basiert: Schüler lernen dann am besten, wenn sie sich aktiv gestaltend mit einem für sie bedeutungsvollen Objekt beschäftigen. Erweiterbarkeit und die Unterstützung schrittweisen und inkrementellen Lernens sind wichtige Kriterien für in Lernszenarien eingesetzte Tools. Kreatives Lernen bedeutet immer auch selbst gesteuertes Lernen, eine zu hohe Komplexität eines Tools (z. B. Fujaba) kann dabei abschreckend wirken. Sind die Grenzen eines Tools dagegen zu schnell erreicht, wird der Schüler um seine kreativen Möglichkeiten beschnitten. Erfüllen lassen sich diese Kriterien u. a. durch verschiedene Experten/Novizen-Ebenen, Plugins und/oder eine Konfigurierbarkeit der Benutzeroberfläche.

Motivationale Kriterien

Intrinsische Motivation ist eine Grundvoraussetzung für Kreativität. In diesem Bereich werden Kriterien beschrieben, welche die Motivation und Interessen der Schüler beeinflussen.

Das Tool
- ist *intuitiv* und hat *geringe Einstiegshürden*
- ermöglicht das *Erschaffen von Produkten/Artefakten* und damit *schnelle Erfolgserlebnisse*
- lässt *Wert und Potential* des Tools schnell *deutlich* werden
- gibt nützliches, aussagekräftiges und sofortiges *Feedback* auf Handlungen
- ermöglicht *Fehlermachen* ohne „Strafe“
- bietet *Verbreitungsmöglichkeiten* um Anerkennung zu erlangen
- macht *Spaß* und soll mit seinen Aktivitäten als „cool“ angesehen werden und die *Interessen der Schüler* ansprechen (z. B. durch Multimedia)
- spricht Jugendliche *unterschiedlichen Hintergrunds* an

Intuitivität erfordert bei Lerntools eine einfache Einarbeitungsphase (insbesondere wichtig für den Schulunterricht) und eine gute Verständlichkeit der verfügbaren Funktionen. Durch Präsentationsmöglichkeiten (z. B. durch WebSharing wie bei Youtube, Flickr etc.) können die Schüler sich präsentieren und Rückmeldungen zu ihren Kreationen erhalten. Eine direkte Sichtbarkeit und Erfahrbarkeit von Erfolgen und Leistungen ermöglicht schnelle Erfolgserlebnisse und motiviert damit zu weiterem kreativen Lernen.

Inspirationsorientierte Kriterien

Vor allem wenn Schüler selbst bestimmt arbeiten kann ein Tool bei der Ideenfindung und -anwendung unterstützen. Hierfür relevante Kriterien werden in diesem Bereich beschrieben.

Das Tool
- ermöglicht *Aktivitäten unterschiedlicher Art* („wide walls“)
- lässt grundlegende *Ideen hervorstechen* (ohne sie aufzudrücken)
- ist *Nachschlagequelle* für grundlegendes Wissen und Anregungen
- liefert *Beispiele* und *Ideen*
- beinhaltet *Tutorials* und/oder *Aufgaben* zur Anregung
- bietet *Kollaborationsmöglichkeiten*

Ist der Einsatzbereich eines Tools sehr begrenzt, besteht die Gefahr, dass das Interesse der Schüler relativ schnell versiegt. Ermöglicht ein Tool dagegen Aktivitäten unterschiedlicher Art, kann das Interesse an der Erkundung und Ausnutzung des Tools Ideen anregen und zu neuen Zielen inspirieren. Ein umfangreiches Repertoire an Beispielen, Ideen und ggf. Tutorials verbreitert das Spektrum der möglichen Schüleraktivitäten mit einem Tool. Zu Kollaborationsmöglichkeiten gehören Ideenaustausch und die Möglichkeit, gemeinsam an Projekten zu arbeiten.

Zusätzlich zu den genannten Bereichen ließen sich noch technische Kriterien (z. B. Kosten, einfache Installation u. a.) angeben, die im Wesentlichen aber so unspezifisch sind, dass hier auf diese verzichtet wird.

Anwendung der Kriterien

Die Kriterien sind offensichtlich nicht alle in gleichem Maße erfüllbar, ja schließen sich teilweise sogar aus. So steht z. B. die Vielfalt an Möglichkeiten (Komplexität) zunächst in einem direkten Widerspruch zur Intuitivität eines Tools. Während die meisten der Kriterien von der schon zuvor erwähnten Programmierlernumgebung Scratch erfüllt werden, stoßen trotz des Design-Kriteriums „high ceilings“ gerade ältere Schüler schnell an die Grenzen von Scratch – wesentliche Konzepte der Programmierung sind nicht implementiert – was auch einen Hauptkritikpunkt an Scratch darstellt. Hier ist ein Kompromiss zu schließen. Demgegenüber steht bspw. Etoys/Squeak, ebenfalls eine Programmierumgebung für Kinder, die allerdings eine vollständige Programmiersprache darstellt bzw. dahingehend erweiterbar ist, aber nur sehr bedingt als intuitiv bezeichnet werden kann.

Traditionelle Tools zum Programmierenlernen

Zu den im Informatikunterricht am häufigsten verwendeten Tools zum Programmierenlernen zählen – neben professionellen Programmiersprachen – wohl Kara, Robot Karol und der Java Hamster. Diese Tools haben gemeinsam, dass in erster Linie mit einer Art Roboter vorgegebene „Probleme“ gelöst werden sollen. Soweit diese Tools auch ver-

schiedene wichtige Kriterien erfüllen, wie *schrittweises Entwickeln*, *Experimentieren*, *geringe Einstiegshürden* oder *aussagekräftiges Feedback*, so sind die Möglichkeiten kreativen Ausdrucks und Lernens mit ihnen doch stark beschränkt. Die mangelnde Möglichkeit, Artefakte zu erschaffen, verhindert konstruktionistisches Lernen. Ebenso sind diese Tools in ihren Betätigungsmöglichkeiten recht einseitig, sie erreichen auch im Bezug auf ihre Komplexität recht schnell ihre Grenzen. Hinsichtlich der motivationionalen Kriterien bleibt fraglich, inwiefern die Interessen der Schüler angesprochen werden, sind sie doch aus ihrer täglichen Umgebung deutlich anspruchsvollere Animationen gewohnt (vgl. [Gu02]). Wert und Potential der Tools können vermutlich bei diesen expliziten Lerntools jenseits der Einsicht, dass das Gelernte „später" einmal hilfreich sein könnte, gar nicht deutlich werden. Trotz teilweise hervorragend ausgearbeiteter Tutorials und Aufgaben (z. B. bei Kara) beschränkt die geringe Aktivitätsbandbreite das Entfalten von Schülerideen – zumindest jenseits der Problemlösungen für gegebene Aufgaben – deutlich.

Neue Tools zum Programmierenlernen

In den letzten Jahren sind immer mehr Tools entwickelt worden, die bereits Programmieranfängern das Entwickeln ansprechender eigener Software ermöglichen und damit versprechen, Kinder und Jugendliche frühzeitig für das selbständig kreative Gestalten von IKT zu beigeistern. Zu den Werkzeugen, zu denen bereits Unterrichtserfahrungen dokumentiert sind, gehören u. a. Scratch, Alice, GameMaker, Squeak/Etoys, StarLogo und Greenfoot. Auf die einzelne Betrachtung der Tools muss an dieser Stelle aus Platzgründen verzichtet werden, dennoch sollen ihre für kreatives Lernen relevanten Gemeinsamkeiten hervorgehoben werden.

Allen diesen Tools ist gemeinsam, dass sie das kreative Erstellen von Spielen, Animationen und anderen Programmen in den Vordergrund stellen und dabei eine weite Bandbreite verschiedener Projekte ermöglichen. Bei der Entwicklung der Tools standen häufig die Interessen der Schüler im Vordergrund, welche auch wieder i. d. R. eine weite Bandbreite verschiedener möglicher Projekte erfordern. Einen anderen Weg geht bspw. das Rapunsel-Projekt [PFH05]: Hier wurde untersucht, wie die Interessen von Mädchen explizit angesprochen werden und resultierend daraus ein Tool explizit für Mädchen entwickelt.

Durch Visualisierung werden bei den Tools Prozessabläufe visualisiert, bestenfalls geschieht dies durch Programmieren mit visuellen Bausteinen, welches keine Syntaxfehler zulässt und gleichzeitig den Umfang programmiertechnischer Möglichkeiten vor Augen hält. Bereits nach wenigen Minuten erstellt ein Schüler bei diesen Tools einen ersten funktionstüchtigen Prototypen, der dann verändert, erweitert oder verbessert werden kann. In diesem konstruktionistischen Vorgehen werden die zugrunde liegenden Ideen schnell implizit deutlich und mit dem Ziel, etwas Eigenes und Interessantes zu erschaffen, bereitwillig und mit Spaß gelernt.

Diskussion

Müssen Werkzeuge der Informatik Kreativität unterstützen? Sollte ein zweckorientiertes Tool wie z. B. Kara denn Kreativität zulassen? Studien zur Kreativität im Informatikunterricht haben gezeigt, dass Kreativität ein wichtiges Motivationsmerkmal im Informatikunterricht darstellt. Ebenso kann eine kreative Herangehensweise an Probleme der Informatik durchaus als typisch für die Informatik bezeichnet werden. Kara, auch wenn primär mit anderen Zielen entwickelt, wird nach eigenen Erfahrungen von Schülern durchaus kreativ – soweit es die Software zulässt – genutzt. Würde der Marienkäfer z. B. die Gestaltung eigener persönlich relevanter Spiele zulassen, wäre eine deutlich höhere Motivation der Schüler, sich mit Kara zu beschäftigen, zu vermuten.

Software kann durchaus auch unkreativ eingesetzt werden, was bei der Betrachtung insbesondere älterer „Lern-Software“, welche vor allem einem Drill-and-Practice Ansatz folgte, deutlich wird. So warnt Hartmann [Ha04], dass Computer und Internet allein im Unterricht kaum zu mehr Kreativität und Effizienz führen werden:

> *Gefragt ist eine effiziente und effektive Nutzung dieser Werkzeuge, für eine kreative und inspirierende Schulumgebung bleiben aber weiterhin die Lehrpersonen verantwortlich.*

IKT fördern Kreativität in der Informatik so, wie Musikinstrumente Kreativität in der Musik beflügelt haben. Dabei sind IKT im Unterricht nicht als „Wundermittel“ zu verstehen, deren alleinige Existenz für die Entwicklung von Kreativität garantiert. Die sorgfältige Auswahl des Werkzeugs und die passenden Aufgabenstellungen sind hierfür grundlegend. Werden vom Informatiklehrer die kreativitätsrelevanten Anforderungen an die Werkzeuge berücksichtigt, können diese einen Teil der kreativen und inspirierenden Schulumgebung ausmachen. Nicht zuletzt weil Computer einen festen Bestandteil der Lernumgebung von Informatikunterricht darstellen, sollte auf einen kreativen Einsatz der Technik geachtet werden. Die vorgestellten Kriterien sind möglicherweise nicht vollständig, bieten aber einen ersten konkreten Anhaltspunkt zur Auswahl eines Tools für den Informatikunterricht. Für die Entwickler von Informatiktools können sie eine Orientierungshilfe wünschenswerter Eigenschaften sein.

Basierend auf den Kriterien ist geplant, unter der Internetadresse www.informatiktools.de einen übersichtlichen Katalog für Softwaretools für den Informatikunterricht bereitzustellen, zu bewerten und damit praktizierenden Informatiklehrern Überblick und Auswahl zu erleichtern.

Literaturverzeichnis

[Di07] Diethelm, I.: Strictly models and objects first - Unterrichtskonzept und -methodik für objektorientierte Modellierung im Informatikunterricht. Dissertation, Universität Kassel, Kassel, 2007.

[Gl06] Glass, R. L.: Software creativity 2.0. developer .* Books, Atlanta, 2006.

[Gr02] Greene, S. L.: Characteristics of Applications that Support Creativity. In Communnications of the ACM 45(10), 2002; S. 100-104.

[Gu02] Guzdial, M.; Soloway, E.: Teaching the Nintendo generation to program. In Commun. ACM 45(4), 2002; S. 17-21.

[Ha04] Hartmann, W.: Computer und Internet im Unterricht: Zu hohe Erwartungen! In Gymnasium Helveticum 1/04 1(44-45), 2004.

[KR08] Knobelsdorf, M.; Romeike, R.: Creativity as a Pathway to Computer Science. Proc. 13th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2008), Madrid, ACM Press, 2008.

[Lo07] Loveless, A.: Literature Review in Creativity, New Technologies and Learning. In NESTA Futurelab series. Report 4, 2007.

[Lu05] Lubart, T.: How can computers be partners in the creative process: classification and commentary on the special issue. In Int. J. Hum.-Comput. Stud. 63(4-5), 2005; S. 365-369.

[MIB03] Mitchell, W. J.; Inouye, A. S.; Blumenthal, M. S.; (U.S.), N. R. C.; Creativity, C. o. I. T. a.: Beyond productivity: information technology, innovation, and creativity. National Academies Press, Washington, DC, 2003.

[Pa93] Papert, S.: The children's machine: rethinking school in the age of the computer. BasicBooks, New York, 1993.

[PH91] Papert, S.; Harel, I.: Situating Constructionism. In (Papert, S.; Harel, I., Hrsg.): Constructionism, Norwood, N.J., Ablex Publishing,1991.

[PFH05] Perlin, K.; Flanagan, M.; Hollingshead, A.: The Rapunsel Project. In (Subsol, G., Hrsg.): Virtual Storytelling, Berlin, Springer,2005; 251-259.

[RKM03] Resnick, M.; Kafai, Y.; Maeda, J.: A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers in Economically-Disadvantaged Communities. In Proposal to the National Science Foundation 2003.

[RMN05] Resnick, M.; Myers, B.; Nakakoji, K.; Shneiderman, B.; Pausch, R.; Selker, T.; Eisenberg, M.: Design Principles for Tools to Support Creative Thinking. In (Shneiderman, B.; Fischer, G.; Czerwinski, M.; Myers, B.; Resnick, M., Hrsg.): Creativity Support Tools. Workshop Report on Creativity Support Tools, Washington, DC, National Science Foundation,2005; 25-35.

[Ro07] Romeike, R.: Kriterien kreativen Informatikunterrichts. Proc. 12. GI-Fachtagung "Informatik und Schule - INFOS 2007", Siegen, Germany, Köllen, 2007.

[Ro08a] Romeike, R.: Applying Creativity in CS High School Education - Criteria, Teaching Example and Evaluation. Proc. 7th Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finnland, Conferences in Research and Practice in Information Technology, 2008.

[Ro08b] Romeike, R.: Kreativität im Informatikunterricht. Dissertation, Universität Potsdam, Potsdam, 2008.

[Sh02] Shneiderman, B.: Creativity support tools. In Commun. ACM 45(10), 2002; S. 116-120.

[Sh07] Shneiderman, B.: Creativity support tools: accelerating discovery and innovation. In Commun. ACM 50(12), 2007; S. 20-32.

[Sh05] Shneiderman, B.; Fischer, G.; Czerwinski, M.; Myers, B.; Resnick, M.: Creativity Support Tools. Workshop Report on Creativity Support Tools. National Science Foundation, Washington, DC, 2005.

[Tr01] Trogemann, G.: Computer und Kreativität. Proc. „Mensch und Computer", Stuttgart, Teubner, 2001.

Kriterien nach Shneiderman [Sh02]	**Kriterien der Entwicklung von Scratch** [RKM03]	**Prinzipien für Construction Kits** [RMN05]
• Pain-free exploration and experimentation	• Youth can create a first project with the tool quickly and easily	• Low Floor and Wide Walls
• Immediate and useful feedback for one's actions	• Youth see the value and potential of the tool right away	• Make it as Simple as Possible – and Maybe Even Simpler
• No big penalties for mistakes, meaningful reward for success	• Youth can learn features of the tool gradually and incrementally	• Support Many Paths, Many Styles
• Easy way to undo and redo	• Youth can create "products" that they can show off to others (pride of authorship)	• Design for Designers
• Visualizing data processes	• The tool supports a wide range of different types of activities	• Choose Black Boxes Carefully
• Searching for knowledge and inspiration	• The tool/activities appeal to youth of different backgrounds and cultures	• Make Powerful Ideas Salient – Not Forced
• Composing a work step by step	• The activities fit into the social dynamic of the Clubhouse	• A Little Bit of Programming Goes a Long Way
• Disseminating results to gain recognition.	• Youth see the tool/activities as "cool," resonating with their interests and passions	• Give People What They Want – Not What They Ask For
	• Youth can continue to use the tool in ever more complex ways over time	• Invent Things That You Would Want to Use Yourself
		• Iterate, Iterate – then Iterate Again

Tabelle 1: Übersicht über verschiedene Kriterien für kreativitätsunterstützende Softwaretools.