Prof. Dr. Alfred Schreiber, Universität Flensburg

# Projekt ZERO: Die Entwicklung von Online-Materialien für die Mathematiklehrer-Ausbildung

This paper reports about a project dealing with the conception and production of supplementary learning material for mathematics teachers. It surveys the various types of courseware-modules presented herein online (e.g., dynamic geometry, computer-based-training-like frames, paper-and-pencil-exercises), and discusses their specific purpose and use. Emphasis is put on the problem of how to embody appropriate functions that provide the opportunity to evaluate user inputs - thus enabling an author to give "local" feedbacks to the student. Finally, some questions are raised concerning the form that should be used in the future to represent both data and logical structure of the underlying content.

## 1. Ausgangspunkt und Ziel des Vorhabens

An deutschen Hochschulen kommt es vergleichsweise selten vor, dass ein Dozent seine Vorlesung in weiten Teilen und eng an einem verfügbaren Lehrbuch ausrichtet (außer er hat es selbst verfasst). Entsprechend groß ist Nachfrage der Studierenden nach (autorisierten) Skripten. Ein solches Skript sollte sich leicht auf den neuesten Stand bringen lassen, kostengünstig herzustellen sowie bequem erhältlich bzw. zu verteilen sein. Das heute übliche Verfahren, es als Dokument im Internet bereitzustellen, macht es leichter als je zuvor, diesen Erfordernissen zu genügen.

Im Fach Mathematik gibt es noch weitergehende, spezifische Anforderungen. Für viele Studierende der ersten Semester stellen die mathematischen Begriffe, Methoden und Denkweisen eine beachtliche Hürde dar. Häufig reicht der klassische Übungsbetrieb nicht aus, um die typischen Anfangsschwierigkeiten zu überwinden. Auch wenn – natürlich – die eigentliche Aneignungsarbeit vom Lernenden selbst zu erbringen ist, so lässt sie sich doch durch ein maßgeschneidertes Angebot studienbegleitender und ergänzender Materialien bis zu einem gewissen Grade erleichtern und fördern. Diese Überzeugung liegt dem in ZERO verfolgten Ansatz zugrunde, über die Bereitstellung von Vorlesungsskripten und Literaturlisten hinauszugehen: Vor allem sollen zahlreiche *interaktive Aufgaben unterschiedlichen Formats und Niveaus* den Studierenden Gelegenheit geben, den Lernstoff nach eigenem Bedarf zu üben, zu vertiefen und sich dabei selbst zu kontrollieren. Zweitens kommen Texte, Bilder und bewegliche Figuren hinzu, die in keinem unmittelbaren Zusammenhang mit dem Inhalt des Grundstudiums stehen. Die Idee dahinter ist die, dass ein locker arrangiertes semantisches Umfeld dazu einladen soll, sich einmal ungezwungen und auf Seitenpfaden der Mathematik zu nähern.

## 2. Überblick über den Inhalt

Das Projekt ZERO hat eine eigene Leitseite im Internet, ist aber als Ganzes eingebettet in den Website des Instituts für Mathematik und ihre Didaktik (Universität Flensburg). Auf der Leitseite des Instituts (http://www.uni-flensburg.de/mathe/) verweisen drei Elemente auf ZERO: der blaue Mittelstreifen „Mathematik online ...“ mit dem ZERO-Logo, der darunter angezeigte Link zur

„Aufgabe des Tages“ und der (unregelmäßig wechselnde) gelbe „Denkzettel“ (samt zugehöriger Links).

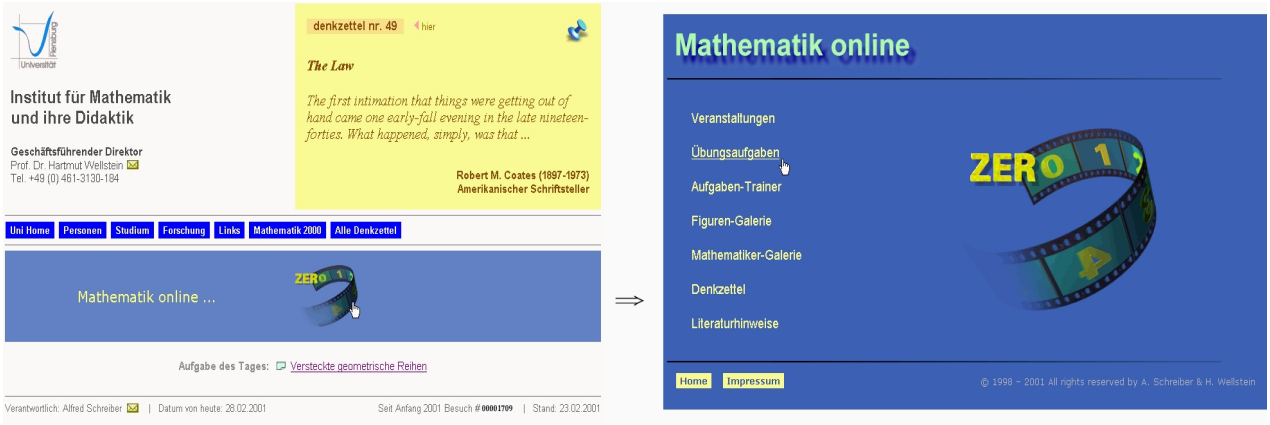


Fig. 1: ZERO Homepage

Auf die Leitseite von ZERO gelangt man durch Anwahl des Logos und findet dort im wesentlichen drei Arten von Lernmaterial:

- *Skripte zu Vorlesungen* (Links: Veranstaltungen, Literaturhinweise)
- *Übungsaufgaben* (Links: Übungsaufgaben, Aufgaben-Trainer)
- *Zusatzelemente* (Links: Figuren-Galerie, Mathematiker-Galerie, Denkzettel).

**Skripte zu Vorlesungen. –** ZERO bietet dem Nutzer einige Dutzend Kapitel aus diversen Vorlesungen in Form herkömmlich aufbereiteter Webseiten (HTML plus CSS), ein Fundus, der ständig gepflegt und weiter ausgebaut wird. Da vor allem Wert auf die logische Struktur(ierung) der Dokumente gelegt wurde, fiel – trotz eingeschränkter Layout-Kontrolle und schwerwiegender Defizite z.B. bei der Darstellung von Formeln – die Wahl auf HTML. Außer Text und Grafik enthalten die Skriptdateien einer Reihe weiterer Elemente: Links zu Literaturangaben (in separaten Listen) und zu diversen Zusatzinformationen (auf ZERO-Seiten und im WWW), MATHEMATICA-Notebooks, sowie bewegliche Figuren auf der Basis von GEOMETRIA/GEOSCRIPT (vgl. [3] und Abschnitt 4).

**Übungsaufgaben.** – Da ich in den folgenden Abschnitten detaillierter darauf eingehen werde, wie die Übungsaufgaben konzipiert und realisiert wurden, mögen hier einige knappe Anmerkungen zu Zweck und Art der Aufgaben genügen. Die bislang (Stand: Februar 2001) knapp 300 Items decken die wichtigsten Themenfelder der ersten beiden Semester ab, z.B. Induktion, Teilbarkeit, Funktionsbegriff, etwas Kombinatorik und Algebra, Elementargeometrie. Es ist geplant, der Sammlung auch fachdidaktische Aufgaben hinzuzufügen. Das Angebot soll in erster Linie den herkömmlichen Übungsbetrieb ergänzen und unterstützen. Es wiederholt dazu den Stoff des Grundstudiums in Standardform und auf unterschiedlichen (elementaren) Niveaus; angehobene Schwierigkeitsgrade bilden die Ausnahme. Zur Minimalausstattung gehören im allgemeinen Lösungshilfen und Musterlösungen.

In technischer Hinsicht lassen sich drei Haupttypen von Aufgaben unterscheiden:

- *klassische Papier-und Bleistift-Aufgabe*:
  Die Problemstellung hat kein vorgegebenes Format. Hilfe und Lösung lassen sich in eigenen Fenstern abrufen (Fig. 2).
- *Lernbaustein mit dynamischer Geometrie*:
  Hier wird eine Figur im Zugmodus und verbunden mit einer Fragestellung dargeboten. In einigen Fällen gibt sie zu Lerneraktionen eine Rückmeldung.
- *Aufgabe im CBT-Format*:
  Die Aufgabe erscheint in einem zyklisch aufgebauten ‚Frame', der die Eingaben des Teilnehmers analysiert und kommentiert.

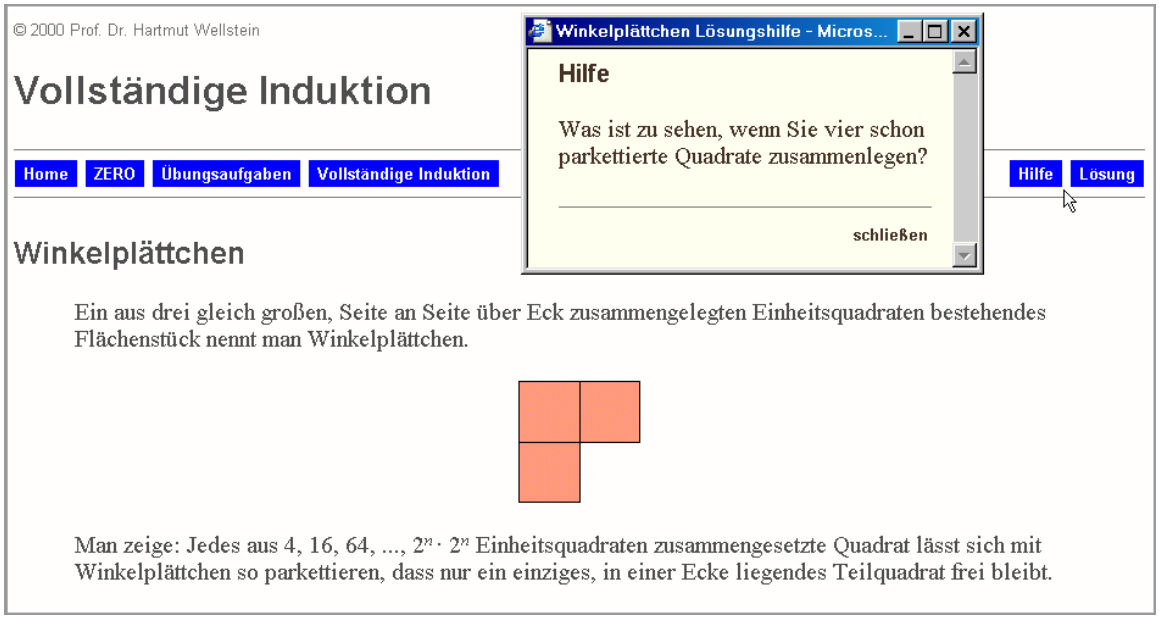


Fig. 2: Beispiel einer Papier-und-Bleistift-Aufgabe

Ich bin mir bewusst, dass keine dieser Aufgabenvarianten frei von Nachteilen ist und dass es weitere Formate (z.B. Simulationen) gibt, die sich gewinnbringend einsetzen ließen. Es erscheint aber akzeptabel, eine vielseitige Mischung anzubieten. Ob das Ergebnis brauchbar ist, hängt zudem wesentlich von der didaktischen Qualität der einzelnen Aufgaben ab.

**Zusatzelemente.** – Die bisher entwickelten Beiträge finden sich unter den drei Rubriken „Figuren-Galerie" (eine Sammlung mathematischer Objekte), „Mathematiker-Galerie" (ausgewählte Größen der Mathematikgeschichte, mit prägnanten Knapptexten zu Leben und Werk) sowie „Denkzettel" (eine sporadisch wachsende Kollektion von Glossen, in denen Beziehungen zwischen der Mathematik und dem ‚Rest der Welt' in welcher Form auch immer angesprochen werden können). Vor allem die „Denkzettel" bieten die Möglichkeit, dem bürokratischen Gehege der Mathematik zu entfliehen, aber auch den einen oder anderen allzu betulichen Schonbezirk der Pädagogik einmal von außen zu betrachten. Sie bilden ein lebendiges Forum, das allen offensteht, keinem festen Plan

folgt, keinen bestimmten Stil pflegt und keinem Trend verpflichtet ist. Die Beiträge sind weit davon entfernt, sich in Unterhaltungsmathematik oder populärwissenschaftlichen Darstellungen zu erschöpfen, was nicht ausschließt, dass sie unterhaltsam sein dürfen. Sie können aktuelle Ereignisse kommentieren, interessante Querverbindungen zu Kunst und Kultur aufdecken, unorthodoxe Gedankengänge entfalten, Kritik üben, Filmisches, Literarisches, Satirisches versuchen, etc. Ich glaube, dass ein solcher Experimentalbereich, der auf Etikette keine Rücksicht nehmen muss, Lehrern dabei helfen kann, ein intellektuell und emotionell freieres Verhältnis zu ihrem Fach zu entwickeln.

## 3. Skizze des Aufgaben-Trainers

Der „Aufgaben-Trainer“ ist ein Test- und Übungsprogramm, das zum Download angeboten wird. Es enthält derzeit 120 Aufgaben (Arithmetik und Algebra, keine Geometrie) und ist *offline* zu nutzen. Gut zwei Drittel davon erscheinen als einzeln dargebotene „Aufgaben im CBT-Format“ auch auf den Webseiten von ZERO.

Die Software basiert auf einem abstrakten adaptiven Trainermodul (DUAL), das ab Anfang der 90-er Jahre entwickelt und in [7] und [8] ausführlicher beschrieben wurde. Hier werde ich nur soweit auf Systemmerkmale eingehen, wie es für das Verständnis des Folgenden erforderlich erscheint.

Das System ist aus drei Schichten aufgebaut: 1. einer *externen Datenbasis* (mit den zu präsentierenden Inhaltselementen), 2. einem *Tutorsystem*, das Lerneraktionen verarbeitet und mit den Daten verknüpft, sowie 3. einem *visuellen Front-End*, über das der Lernende mit dem Tutor kommuniziert (Fig. 3 zeigt stark vereinfacht diesen Aufbau). Genau genommen arbeiten in der zweiten Schicht zwei Teilsysteme zusammen:

1. Ein *Makrotutor* entscheidet aufgrund der ihm bis dahin vorliegenden Lerndaten darüber, welche Aufgabe als nächste dargeboten wird. Das zugehörige Item berechnet ein Algorithmus (Lehrstrategie genannt). Über eine Art ‚Einschub‘ lässt sich der Makrotutor mit unterschiedlichen Lehrstrategien bestücken. Der Teilnehmer wird via Passwort identifiziert, sodass die zugehörigen Lerndaten individualisiert verarbeitet werden können.
2. Ein *Mikrotutor* präsentiert jedes vom Makrotutor angeforderte Aufgaben-Item als Frame. Dieser gibt zunächst seinen Typ (diverse Auswahl-, Zuordnungs- und Texteingabeformen) zu erkennen und weist auf die externen Daten hin, die er für die Darbietung benötigt. Anschließend startet der Elementarzyklus des Frames; der Mikrotutor führt ihn (den Vorgaben über Wiederholungsmöglichkeiten entsprechend) zuende und gibt das Ergebnis an den Makrotutor zurück.

*Trennung der Komponenten* war eine Leitidee beim Entwurf und bei der Entwicklung des Systems (vgl. [5], [7]). Sie ließ sich zwar nicht zur Gänze durchhalten, aber doch soweit, dass der Mikrotutor veranlasst werden kann, seine Arbeit auch ohne den Makrotutor zu verrichten. Praktisch bedeutet das: Ein fremdes Programm kann, gleichsam in der Rolle des Makrotutors, einzelne Frames nach Belieben (aus einem oder mehreren DUAL-Projekten) anfordern und dargeboten bekommen.

Das ist ein markanter Unterschied zu CBT-Anwendungen, die Inhaltsdaten und Programmlogik in einer ‚gemischten Struktur' enthalten, wie sie bisher bei den meisten Autorensystemen üblich war.

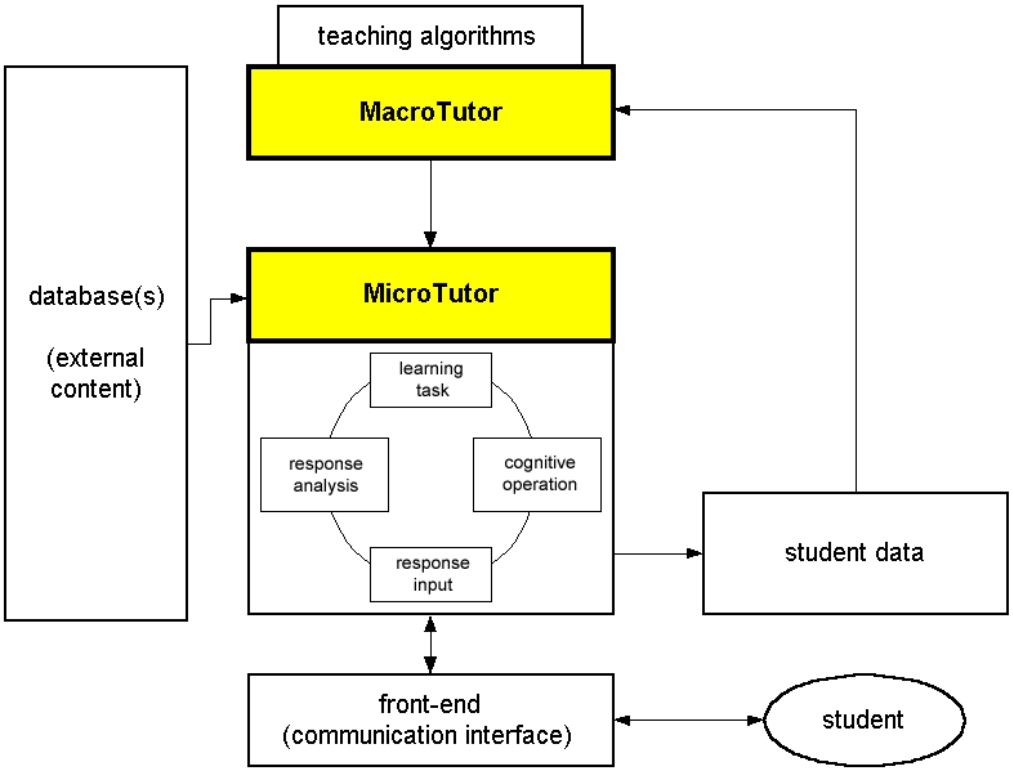


Fig. 3: Schematischer Aufbau des tutoriellen Rahmensystems DUAL (A. Schreiber / T. Merino)

## 4. Aufgaben im CBT-Format

Etwa 70 Prozent der im „Aufgaben-Trainer" enthaltenen Items sind in den Sparten der ZERO-Übungsaufgaben einzeln wählbar und online ausführbar. Um dies zu erreichen, wurde der Mikrotutor auf die Java-Plattform portiert. Das betreffende Applet zeigt die Aufgabe innerhalb einer Webseite oder auf Wunsch in einem eigenen Applet-Fenster.

Das neue Mikrotutor-Applet unterstützt praktisch alle früheren Aufgabentypen. (Derzeit noch unberücksichtigt sind generative Rechenaufgaben, die eine individuelle Problemstellung samt Musterlösung über das Modell einer Sachsituation zufallsgesteuert kompilieren.) Im einzelnen sind dies: Ja-Nein-Entscheidungen, Multiple-Choice-Aufgaben mit randomisiert dargebotenen Auswahlitems (jeweils in den zwei Varianten: ‚1 aus *n*' und ‚*k* aus *n*'), injektive und nicht-injektive Zuordnungen von Merkmalen zu Objekten (vgl. ein Beispiel in Fig. 4), Lückentexte und Aufgaben mit freier alphanumerischer Eingabe.

Der Inhalt der Aufgaben befindet sich in typspezifischen Datenbanken, auf die der Mikrotutor zugreift. Jeder Datensatz enthält zudem Angaben, die den Ablauf des Dialogs steuern, z.B. Höchstzahlen für Antwortversuche (je nach Bewertung der Antwort) sowie diverse Vorschriften für die

Prüfung von Antworten. Bei Aufgaben vom Auswahl- oder Zuordnungstyp gibt es, je nach Variante, außer Richtig und Falsch noch Zwischenergebnisse vom Typ ‚gemischt' oder ‚unvollständig'.

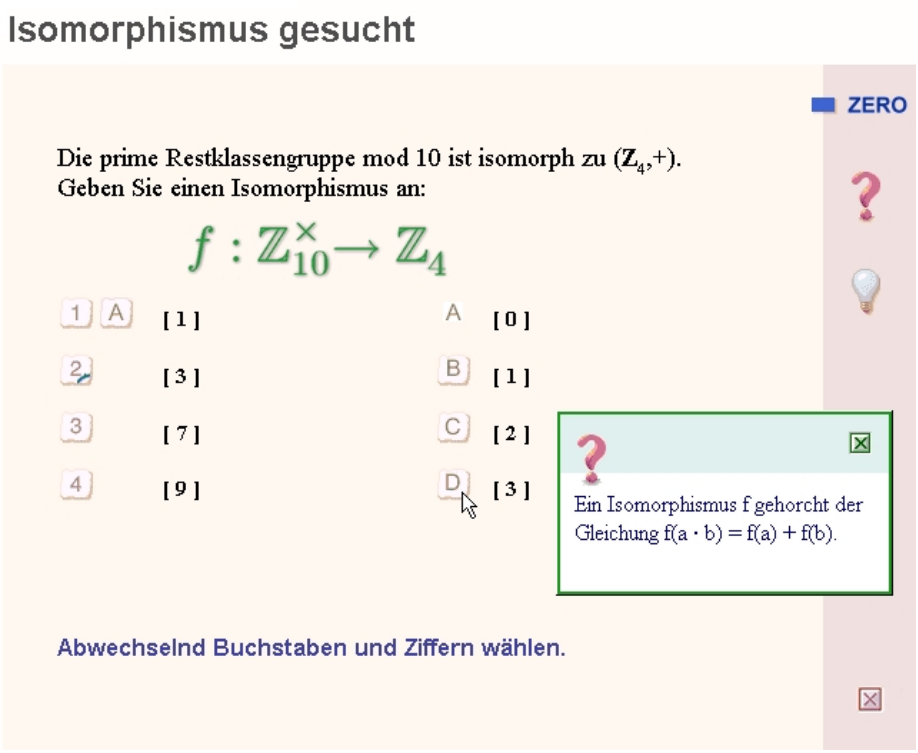


Fig. 4: Beispiel CBT-Aufgabe (Typ: injektive Zuordnung)

Der Mikrotutor verfährt dabei nach einem universellen Schema, das auf alle Aufgaben dieser Art anwendbar ist. Sollen hingegen alphanumerische Eingaben ausgewertet werden, liegen die Dinge komplizierter. Der Autor einer solchen Aufgabe muss sich vorab überlegen, wie er welche möglichen Antworten bewerten möchte. Er kann dabei (mit Hilfe von booleschen und diversen Toleranz-Operatoren) gleich ganze Antwortklassen beschreiben und in den Datensatz der Aufgabe als sog. Prüfschlüssel eintragen (für Einzelheiten vgl. [4], [8] und [9]). Liegt eine Antwort in keiner der vordefinierten Antwortklassen, so bleibt für sie nur die Bewertung als ‚nicht-identifiziert' oder ‚unverständlich'. Das ist eine prinzipielle Schwachstelle jeder Art von Musterabgleich (selbst dann, wenn semantische Verfahren eingesetzt werden, da schließlich kein Nutzer von Lernsoftware davon abgehalten werden kann, absichtlich Unsinniges einzugeben). Die beste Empfehlung, die ich aus eigener Erfahrung geben kann, lautet: mit sinnvollen Eingaben rechnen und diese soweit wie möglich durch den Kontext der Aufgabenstellung determinieren bzw. eingrenzen.

## 5. Lernbausteine mit dynamischer Geometrie

Heute bieten die Programme für dynamische Geometrie zumeist auch die Möglichkeit, ihre Objekte ins Web zu exportieren. Zu den ersten Systemen, die von vornherein auf das Internet ausgerichtet waren, gehört das GEOMETRY-APPLET von D. Joyce. Ein Teil der darin geleisteten Arbeit diente als

Grundlage eines Vorhabens [3], das 1997 an der Universität Flensburg begonnen wurde und zum Ziel hatte, eine möglichst große Klasse flexibel erzeugbarer geometrischer Objekte mit dynamischem Verhalten für das Internet bereitzustellen. Ich will summarisch und an wenigen Beispielen darlegen, durch welche Akzentuierungen sich dieser Ansatz auszeichnet:

1. Eine Figur entsteht durch eine Folge von Anweisungen in einer (aus dem GEOMETRY-APPLET heraus weiterentwickelten) skriptartigen Sprache (GEOSCRIPT); die Befehlszeilen sind in einer separaten Datei abgelegt. Im Vordergrund stehen demnach die Beschreibung einer Figur durch einen Autor und das aus ihr resultierende Erzeugnis (bewegliche Figur in einem Lehrtext, interaktives Arbeitsblatt, etc.). Ein Java-Applet (mit dem Namen GEOMETRIA) sorgt für seine Ausführung im Browser.

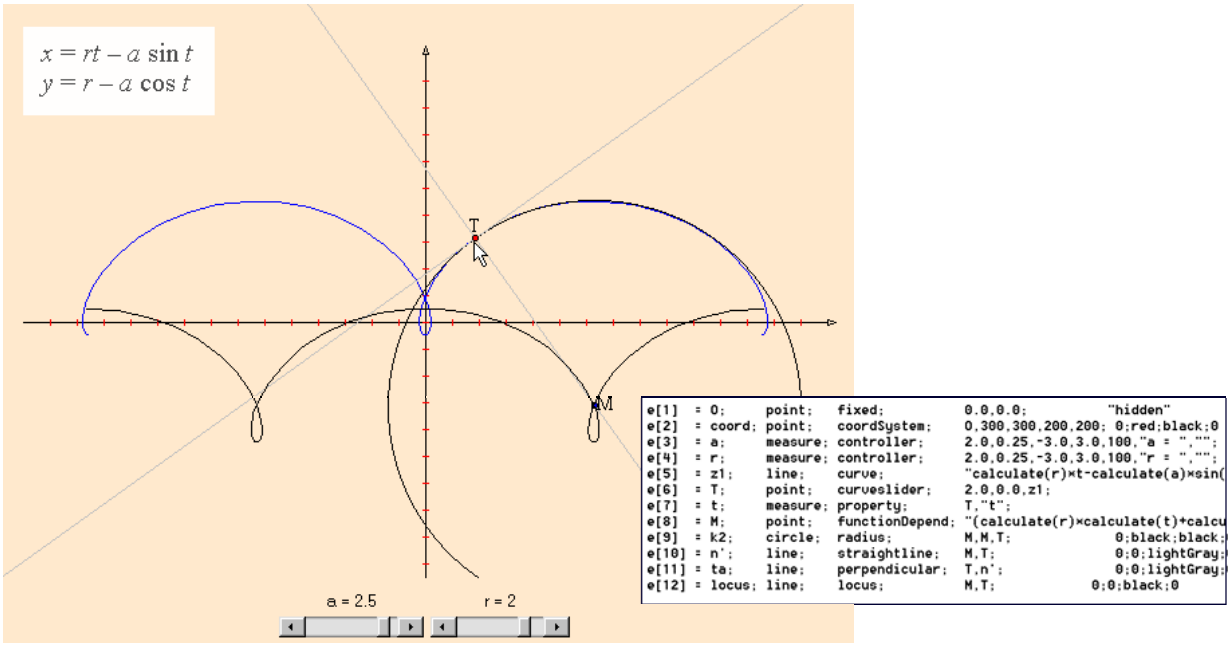


Fig. 5: Evolute der Zykloide (GeoScript: T. Ehmke)

2. Die Möglichkeiten von GEOSCRIPT lassen sich nach Belieben durch eigene Funktionsbibliotheken erweitern (ohne dass dazu der Quellcode von GEOMETRIA neu übersetzt werden müsste). Die betreffende Programmierschnittstelle ist an die zentrale Klasse `Measure` gebunden, mit der sich die Zustände von Objekten kontrollieren lassen. Außer systeminternen Funktionalen steht dem Figurenautor der volle Sprachumfang von Java zur Verfügung; auch JavaScript kann verwendet werden.

3. Ein Objekt lässt sich nicht allein mit euklidisch-konstruktiven oder abbildungsgeometrischen Mitteln aufbauen. Punkte, Vektoren, Kurven, usw. können ohne Umweg (d.h. Hilfskonstruktionen) direkt durch analytische Vorschriften erzeugt werden (s. Fig. 5). Selbst beliebige – auf der Zeichenfläche darstellbare – Punktmengen können als Bezugsobjekte für ziehbare Punkte dienen. Durch die externe Schnittstelle kommen weitere Methoden in Betracht, z.B. Rekursion oder Limesbildung (etwa für fraktale Objekte, s. Fig. 6).

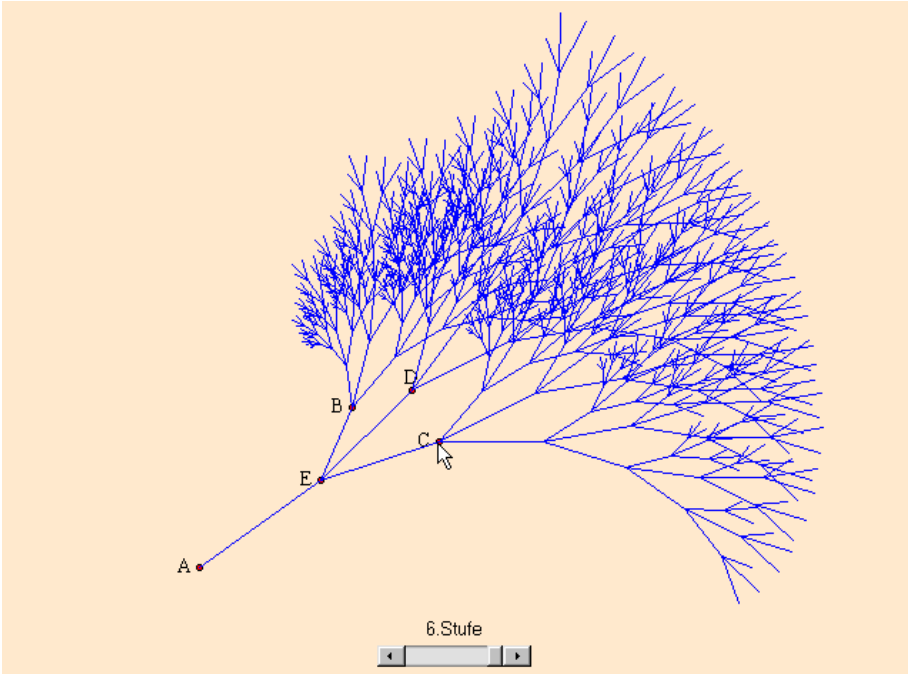


Fig. 6: Ziehbarer fraktaler Baum (GeoScript: T. Ehmke)

4. CBT-Funktionalität: Ein GEOMETRIA-Objekt lässt sich (dank der Möglichkeiten, welche die `Measure`-Klasse bietet) mit einer Art Frame-Struktur versehen. Der Figurenautor kann Zustände von Teilobjekten abfragen und auf diese Weise die Eingaben eines Schülers differenziert bewerten und kommentieren. Eine Eingabe kommt durch eine Aktionssequenz zustande, welche die Figur in einen bestimmten Zustand versetzt (Fig. 7).

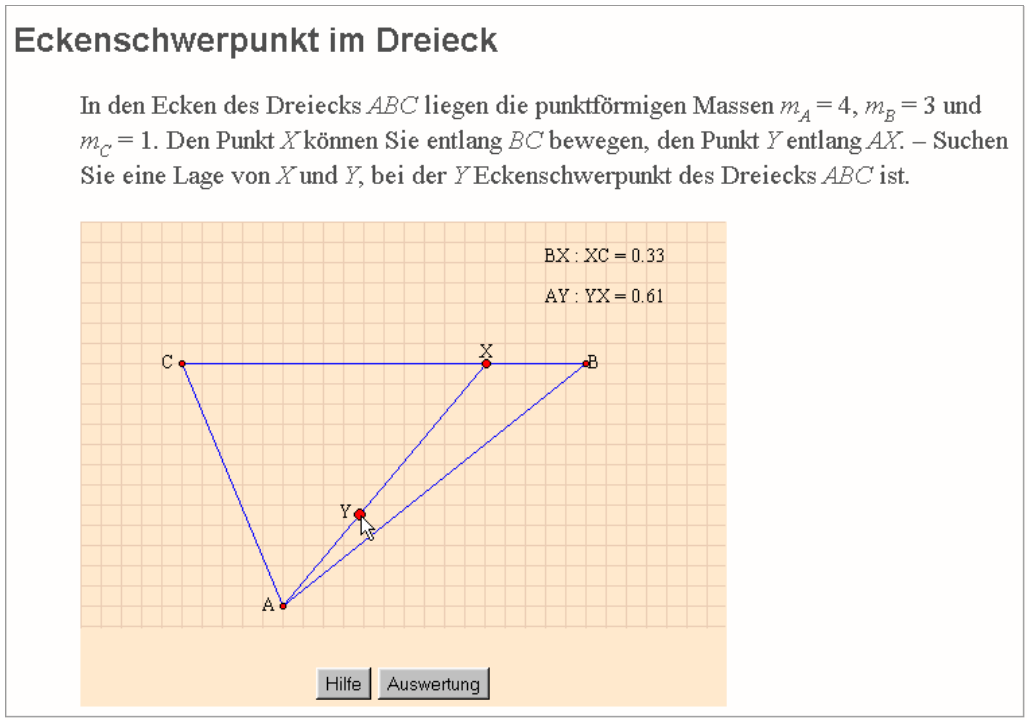


Fig. 7: Dynamische Figur mit Lösungshilfe und differenzierter Antwortauswertung

Aus den Punkten 1-3 leuchtet ein, dass die Konzeption der Klassenhierarchie und die eigentliche Applet-Entwicklung zunächst Vorrang vor der Entwicklung eines Konstruktionseditors hatten. Freilich soll diese nicht ausgeschlossen werden. Inzwischen kann EUKLID DYNAGEO (von R. Mechling) einen Großteil seiner Figuren nach GEOSCRIPT exportieren; die erweiterten Funktionen lassen sich dann nachträglich via GEOSCRIPT und/oder die Programmierschnittstelle nutzen.

## 6. Modellierung des Inhalts: Probleme und Vorschläge

Wenn ich in diesem Stadium von ZERO auf die bisherige Projektentwicklung (von mehr als 3 Jahren) zurückblicke, so rückt unter allen Problemen, die bei Unternehmungen dieser Art zu meistern sind, vor allem eines in den Vordergrund*: die Repräsentation (Modellierung) des Inhalts*. – An drei Beispielen möchte ich die Dringlichkeit und Tragweite dieses Themas erläutern:

a) Um formelhaltige Texte akzeptabel auf Webseiten präsentieren zu können, wurde der Quellentext zunächst in MATHEMATICA als Notebook-Datei geschrieben und anschließend nach HTML konvertiert. Eine so hergestellte Datei muss in der Regel noch gründlich überarbeitet werden. So ist das Layout an die verwendete Stilvorlage (samt Navigation) anzupassen; ferner sind die Schriftgrößen von Text und Formeln (GIF-Grafik) anzugleichen und zu fixieren. – Dieses Verfahren hat den *gravierenden Nachteil*, dass es die aktuelle Fassung des Textes von seiner Quelle abschneidet. Beschränkt man sich stattdessen darauf, den Quellentext lediglich umzuwandeln, so hat man zwar ein Dokument in mustergültiger Typografie (wie z.B. bei der Konversion von TEX nach PDF), jedoch keine echte Einbettung in die Zielumgebung.

b) Der Lehrstrategie-Algorithmus im Aufgaben-Trainer operiert auf einer ausgelagerten Menge von Items, und auch für den Mikrotutor sind die Daten dieser Items extern. Man sollte daher erwarten dürfen, dass z.B. ein in Java geschriebenes Mikrotutor-Applet *mit denselben Daten* arbeiten kann. Doch in der Entwicklerpraxis erfüllen sich Ideale höchst selten! Die Grafiken waren an neue Bildformate anzupassen, und die Texte waren im Hinblick auf veränderte typografische Erfordernisse zu überarbeiten. Immerhin, die Portierung der Daten gelang zu ca. 80 Prozent. Bei dieser Gelegenheit wurden noch einige inhaltliche Fehler korrigiert und die eine oder andere Formulierung verbessert. Im Endeffekt gibt es also, wie bei a), zwei Quellen, oder – pessimistisch ausgedrückt – keine.

c) Die in a) und b) skizzierten Probleme entstehen dadurch, dass der Entwicklungsprozess zu einer Modellierung des Inhalts führt, die sich nicht als Quelle weiterer Bearbeitungen eignet. Ein anderes Problem ist *die Vielzahl der Modellierungen selbst*. Daran kranken nicht nur herkömmliche Autorensysteme für CBT; auch geometrische Konstruktionswerkzeuge speichern ihre dynamischen Objekte in proprietären (und untereinander unverträglichen) Formaten ab. Einige Systeme verfügen über eine Parameter-Schnittstelle (z.B. das GEOMETRY-APPLET von D. Joyce, JAVASKETCHPAD von N. Jackiw) oder definieren ihre Figuren, etwas bequemer, über eine Skriptsprache (z.B. GEOMETRIA/GEOSCRIPT von T. Ehmke [3]). Dadurch wird das zugrunde liegende Modell immerhin etwas durchsichtiger. Die wirklich interessierende Frage scheint mir jedoch die nach einem aner-

kannten Repräsentationsstandard zu sein. Es mag unrealistisch sein, zu erwarten, in der Dynamischen Geometrie werde man sich bald über ein einheitliches Beschreibungsverfahren verständigen können. Vielleicht lassen sich aber (höchstens) zwei oder drei Konkurrenzmodelle finden; auch das wäre ein Fortschritt.

Lassen sich die in a)-c) geschilderten Schwierigkeiten überwinden? Ich glaube, ja, und ich bin ferner davon überzeugt, dass dies durch die konsequente Anwendung von XML (Extensible Markup Language) möglich ist. XML ist eine (inzwischen vom W3 Consortium standardisierte) Metasprache zur Definition bereichsspezifischer Auszeichnungssprachen. Zu ihren ersten Teilerfolgen zählt die Spezifikation von Mathematical Markup Language (MathML), mit der sich mathematische Formeln beschreiben lassen, ferner die konkurrierenden XML-Anwendungen (wie SVG, VML) zur Beschreibung zweidimensionaler Vektorgrafiken. Es liegt nahe, in ähnlicher Weise Fachsprachen für eine Reihe von Gebieten zu spezifizieren, die eine bedeutende Rolle für die *Entwicklung und Darbietung interaktiver didaktischer Materialien* spielen, z.B.

- *Dynamische Geometrie*
  Hierbei lässt sich von vornherein eine größere Allgemeinheit erzielen, innerhalb der dann spezielle Objektklassen, z.B. die mit Zirkel und Lineal konstruierbaren Figuren, abgegrenzt werden können. Interessant erscheint auch die Möglichkeit, bestimmte ‚instruktionale' Funktionen zu definieren, etwa Feedback-Mechanismen (zu denen es verschiedenartige Ansätze z.B. in Cinderella oder in GEOMETRIA/GEOSCRIPT gibt).
- *Modellbildung / Simulation*
  Zweck einer geeigneten Fachsprache wird u.a. sein, die definierenden Relationen bestimmter Arten von Systemen (z.B. diskrete dynamische Systeme) sowie Optionen ihrer raumzeitlichen Wiedergabe (Simulation) festzulegen.
- *Unterrichtsformen*
  Die vom CBT her bekannten Aufgabentypen (vgl. [1], [8]) lassen sich als Sonderfälle formal beschreibbarer Unterrichtsformen auffassen. Einen allgemeinen (nicht computer-zentrierten) Ansatz dieser Art hat vor mehr als einer Dekade K. Eckel mit seiner „Didaktiksprache" [2] vorgelegt. Die Grundidee lässt sich zumindest gewinnbringend auf CBT anwenden [8]. Die darüber hinausgehenden Versuche sind noch einigermaßen unausgegoren [6], führen aber auf eine Fülle interessanter Forschungsfragen.

Welche Vorteile dürfen wir uns von geeigneten XML-Anwendungen auf solchen Gebieten erhoffen? – Zunächst einmal kämen wir in den Genuss transparenter und qualifizierter Quellendaten: Sie sind für jedermann lesbarer Klartext, besitzen eine logische Struktur, enthalten dokumentierende Metainformation und eignen sich als robustes *Austauschformat*. Auch der wissenschaftliche Nutzen ist beachtlich, wenn man bedenkt, dass nun alle Beteiligten angehalten sind, über offene Standards zu diskutieren statt über irgendwelche Software-Features (auf der Basis womöglich unbekannter Modelle). Im übrigen werden damit die begrifflichen und konstruktiven Grundlagen dem eigentlichen Software-Engineering logisch vorgeordnet und besser von ihm getrennt. Entwicklern bleibt

dann immer noch die ehrgeizige Aufgabe, Darstellungsmodule (z.B. für Web-Browser) zu schaffen oder leistungsfähige Editoren, die ihre Daten in das jeweilige XML-Derivat exportieren. Diese Tätigkeit ist umso lohnender, desto mehr im Vorfeld Sicherheit und Konsens in Fragen des Modellaufbaus hergestellt wurde.

Inhaltliche Beiträge zu ZERO stammen auch von Hartmut Wellstein (der Auszug aus der Vorlesung „Elementargeometrie“, zahlreiche Übungsaufgaben, eine Reihe von Denkzetteln); Timo Ehmke steuerte das GEOMETRIA-APPLET und vor allem Lernbausteine zur Figuren-Galerie bei; Josef Schmid-Egger schrieb ein Dutzend Denkzettel. Ihnen und allen hier nicht genannten Personen, die mir Texte, Kritik oder Anregungen haben zukommen lassen, möchte ich an dieser Stelle herzlich danken.

## Literatur

[1] Alessi, S. M. / Trollip, S. R.: *Computer-Based Instruction. Methods and Development*. Prentice Hall: Englewood Cliffs, New Jersey 1985, 2nd ed. Allynand Bacon

[2] Eckel, K.: *Didaktiksprache*. *Grundlagen einer strengen Unterrichtswissenschaft*. Köln, Wien: Böhlau 1989. Amerikanische Ausgabe: *Instruction Language*. *Foundations of a Strict Science of Instruction*. Educational Technology Publications, Inc.: Englewood Cliffs, New Jersey 1993

[3] Ehmke, T.: *Eine Klasse beweglicher Figuren für interaktive Lernbausteine zur Geometrie*. Diss. Universität Flensburg 2001

[4] Nesbit, J. C.: Approximate String Matching in Response Analysis. In: Journal of Computer-Based Instruction 12 (1985), 71-75

[5] Schreiber, A.: Bausteine für Lernprogramme: Beschreibung und Implementierung. In: Informatik- Fachberichte Nr. 259. Springer: Berlin, Heidelberg, New York 1990, 333-348

[6] Schreiber, A.: Rezension von [2]. In: Zentralblatt für Didaktik der Mathematik 22 (1990), 114-116

[7] Schreiber, A.: Eine Didaktik-Umgebung für Adaptives Lernen (DUAL). In: Grundlagenstudien aus Kybernetik und Geisteswissenschaft / Humankybernetik 33 (1992), 25-31

[8] Schreiber, A.: *CBT-Anwendungen professionell entwickeln*. Springer: Berlin, Heidelberg, New York 1998

[9] Schreiber, A.: Die Analyse von Schülerantworten in Lehrprogrammen. In Vorbereitung (für Grundlagenstudien aus Kybernetik und Geisteswissenschaft / Humankybernetik).