Aus der Arbeit im Informatik-Seminar in der 2.Phase der Lehrerausbildung

Eberhard Lehmann

Der folgende Beitrag kann nur einen kleinen Ausschnitt aus der vielfältigen Arbeit im Informatik-Seminar geben. Es wird versucht, die wichtigsten Zielsetzungen zu formulieren. Dann werden einige Beispiele für den Ablauf von Seminarsitzungen gegeben, jedoch auch nur in grober Darstellung. Dazu werden einige weitere wichtige Inhalte der Ausbildung genannt, auf die jeweils ausführlicher einzugehen wäre. Aus einer der Sitzungen wird etwas genauer über erarbeitete Ergebnisse - Demonstrationsprogramme für den Informatikunterricht - berichtet.

Grundsätzlich ist zu beachten, daß die Zielsetzungen in den Informatik-Seminaren (2.Phase der Lehrerausbildung) wohl recht ähnlich sind. Die Inhalte differieren jedoch vermutlich ebenso wie es die Informatik-Lehrpläne tun. Diese Inhalte sind teilweise ziemlich festgeschrieben (siehe Beitrag von Karin Klein (LOGIN 1´96)) oder weitgehend der Erfahrung des Seminarleiters überlassen. Ein Austausch von Erfahrungen aus den zur Zeit noch sehr wenigen Informatik-Seminaren der 2.Phase der Lehrerausbildung über Bundesländergrenzen hinweg findet bislang meines Wissens nicht statt. Ansätze hierzu sind geplant.

1. Aufgaben des Informatik-Seminars

Hauptziele der Ausbildung im Fachseminar Informatik:

- Informatikunterricht sachgerecht vorbereiten, erfolgreich durchführen und anhand von Kriterien analysieren können,
- erziehend wirken und dabei das selbständige Denken der Schüler fördern,
- Schülerleistungen angemessen beurteilen können.

(1) Die Inhalte der Ausbildung sollten sich demzufolge in erster Linie an der Unter-richtspraxis orientieren. Dabei ist zu berücksichtigen, daß der Referendar vom Be-ginn seiner Ausbildung an bereits die volle Komplexität von Unterricht erlebt.

Der Referendar begegnet der Unterrichtspraxis "vor Ort"
a) in gemeinsamen Hospitationsstunden, gegeben vom Seminarleiter oder Mitreferenda-ren als (möglicherweise benotete) Unterrichtsversuche,
b) in Hospitationen bei Kollegen der Ausbildungsschule,
c) im eigenen Unterricht, als Übungsunterricht oder eigenverantwortlich.

(2) Grundlagen für den Unterricht sind insbesondere
a) die im Studium erworbenen Informatik-Kenntnisse,
b) die einschlägigen Lehrpläne oder auch andersartige Ansätze aus Lehrplänen anderer Bundesländer,
c) fachdidaktische Literatur,
d) Schulbücher und andere einschlägige Veröffentlichungen (Zeitungen!),
e) fachspezifische Unterrichtsformen und -methoden,

(3) Die schnelle Entwicklung der Fachwissenschaft Informatik und ihrer Anwendungen bringt besondere Probleme für den Informatikunterricht. Sie führen dazu, daß auch in der Seminararbeit hierüber reflektiert werden muß. Andererseits zwingen die innovativen Entwicklungen dazu, sich auf Kerninhalte zu besinnen, etwa in Form passender Leitideen für die Unterrichtsinhalte und Methoden. Diesen ist dann in der Seminararbeit auch besonderes Gewicht zu geben!

2. Abläufe von Informatik-Seminarsitzungen

Die oben genannten Bemerkungen zu den Aufgaben des Fachseminars lassen erkennen, daß es diverse Möglichkeiten für die Planung und Durchführung von Informatik-Seminarsitzungen gibt.
Im folgenden werden einige Ablauffiguren von Informatik-Seminarsitzungen angegeben, wie sie sich in der Praxis bewährt haben. Eine bestimmte Reihenfolge der Sitzungen wird hier nicht intendiert! Wie anderswo gilt auch hier:

Eintönige Organisationsformen und Abläufe sollten vermieden werden!
- Keine kurzschrittige Planung,
- keine Überfrachtung mit Theorie oder Referaten,
- keine einseitige Gestaltung der Sitzungen

vielmehr wird gefordert:
- Flexibilität gegenüber kurzfristig entstehenden Bedürfnissen der Referendare,
- Berücksichtigung aktueller, möglicherweise unterrichtsrelevanter Themen,
- Beobachtung der fachdidaktischen Zeitschriften, Bücher und Tagungsberichte,
- ständige Verfolgung der schnellen Entwicklung der Informatik mit Reflexion der für den Unterricht daraus erwachsenen Bedürfnisse und Möglichkeiten

aber auch längerfristig geplante Referate und Betrachtungen
- zur Planung von Unterricht
  kurzfristig - die einzelne Stunde
  mittelfristig - die Unterrichtsreihe
  langfristig - das Unterrichtshalbjahr

- zu wichtigen didaktisch-methodischen Themen,
- zur didaktischen Standardliteratur,
- zu den fachspezifischen Methoden der Informatik

Sitzung A (Dauer 4 Schulstunden)

1) Vorführstunde des Seminarleiters mit dem Aspekt:"Auswertung eines Zeitungsartikels".
Die Referendare erhalten eine Anlage mit den vorgesehenen Lernzielen und dem Beobachtungsauftrag: "Wurden die Lernziele erreicht? Wodurch?"

2) Besprechung der Stunde, insbesondere unter dem Lernzielaspekt. Allgemein gültige Ergebnisse werden ins Protokoll übernommen.

3) Diskussion über die Verwendbarkeit von Zeitungs- oder Zeitschriftenartikeln im Unterricht und ihrer didaktisch-methodischen Aufbereitung.
a) Stillarbeit in 3 Gruppen,
b) Diskussion,
c) Festhalten der wichtigsten Ergebnisse im Protokoll.

4) Hausarbeit: Übung an einem weiteren Zeitungsartikel: Es sollen Fragen zu dem Artikel
an die Schüler formuliert werden.

5) Organisation der nächsten Unterrichtsversuche (Terminabsprachen)

Sitzung B (Dauer 4 Schulstunden)

1) Vorführstunde eines Referendars, die anderen Referendare erhalten einen Beobachtungsauftrag, z.B. "Impulsgebung".

2) Besprechung der Stunde
a) Der unterrichtende Referendar äußert sich,
b) Diskussion aller Seminarmitglieder, u.a. besondere Stärken oder Schwächen des Unterrichts, Verbesserungsvorschläge; im Vordergrund steht die Impulsgebung.
c) Abschließende Stellungnahme des Seminarleiters, ggf. Note, Alternativen. Hinweise
zu im Stundenablauf sichtbar gewordenen, häufiger auftretenden methodischen Problemen (unterrichtspraktische Tips).

3) Vortrag eines Referendars zum Thema: Medien im Informatikunterricht (außer Computer)

4) Hinweis auf Fortbildungsveranstaltungen

Sitzung C (Dauer 4 Schulstunden)
Der Seminarleiter hat bei einem seiner Unterrichtsbesuche ein Demonstrationsprogramm zu einem Sortierverfahren gesehen, das der besuchte Referendar in überzeugender Weise eingesetzt hat.

1) Der Referendar berichtet kurz über die Einbettung des Demo-Programms in seinen Unterricht. Er demonstriert das Programm.

2) Auftrag an die Seminarteilnehmer: "Sinn und Unsinn des Einsatzes von Demo-Programmen im Informatik-Unterricht (zur Unterstützung fachlicher Intentionen)". Arbeit in Gruppen. Ergebnisse werden kopierbar fixiert (ggf. am Rechner).

3) Besprechung der Ergebnisse, Ergänzungen des Seminarleiters.

4) Zusammenstellung von leicht zugänglichen Demo-Programmen (z.B. aus den Schulen
der Seminarteilnehmer).

Beispiele für weitere Themen in der Seminararbeit

(auf Vollständigkeit wird kein Wert gelegt)

Aspekte bezogen auf Einzelstunden oder Unterrichtsreihen (mehr lokale Aspekte)

- Festlegung von Stundenzielen
- Kriterien für Stundenanalysen
- Anfertigung von Stundenentwürfen
- Unterrichtsformen bei der Arbeit mit Unterrichtsmaterial
- Möglichkeiten der Ergebnissicherung
- Möglichkeiten der Erfolgskontrolle
-

Übergeordnete (mehr globale) Aspekte

- Ziele des Informatikunterrichts, Abgrenzung gegen Informationstechnische Grundbildung in der Sekundarstufe 1
- Leitlinien, z.B.
  - Benutzung, Analyse und Konstruktion komplexer Systeme
  - Spiralprinzip
- Gestaltung des Anfangsunterricht
- Konstruktion und Bewertung von Klausuren
- Grundzüge der Projektarbeit im Informatikunterricht
- Schülerkompetenz im Informatikunterricht

-

Konkrete Unterrichtsinhalte

- Einführung der Rekursion
- Grundlagen des Software-Engineering
-

3. Demonstrationsprogramme als Hilfsmittel zur Erreichung informatorischer Lernziele

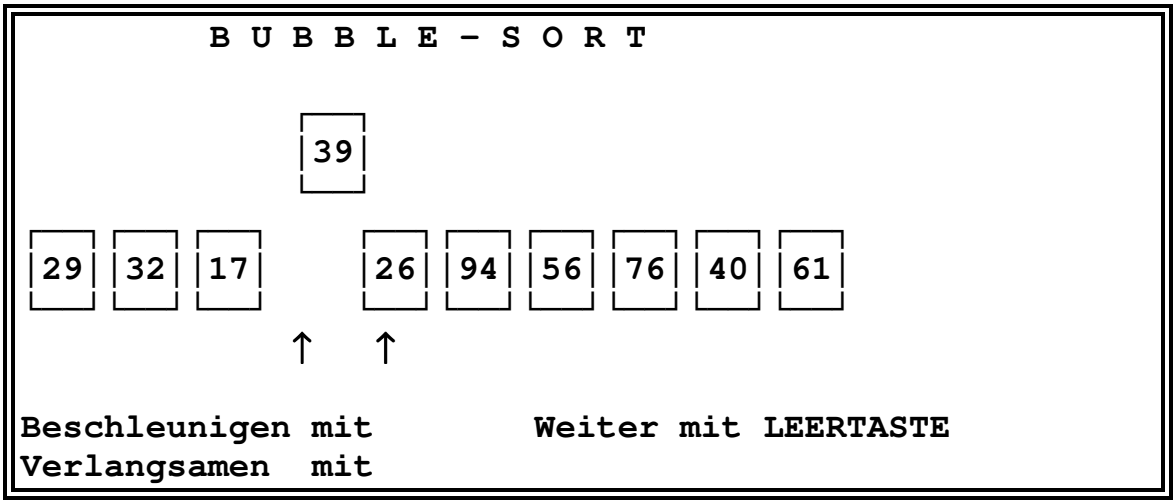
Zur Konkretisierung einer der Seminarsitzungen (Sitzung C) wird hier das Thema "Demonstrationsprogramme" etwas ausführlicher dargestellt. Ausgangspunkt war ein von einem Referendar eingesetztes Programm zur Demonstration von Sortierverfahren.

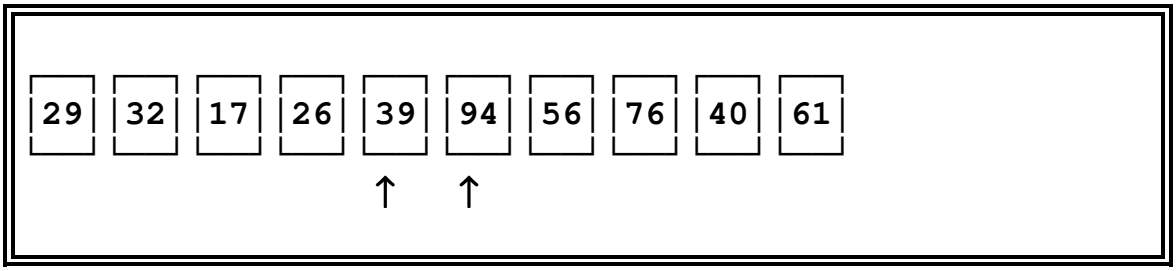
Beispiel:
Erarbeitung von Bubble-Sort mit Hilfe eines Demo-Programms

Seit vielen Jahren gibt es ein gut verwendbares Demonstrationsprogramm von CUM (s.u.) für einige Sortierverfahren: Aus dem Vorrat der vorhandenen Sortierverfahren kann eins ausgewählt werden. Die zu sortierende Zahlenfolge kann in unterschiedlicher Weise festgelegt werden. Im folgenden Beispiel wurde Bubble-Sort gewählt (paarweiser Austausch), eine Zufallszahlenfolge von 10 Zahlen sollte aufsteigend sortiert werden.

CUM, Computer als Unterrichtsmedium, Berichte zum Modellversuch 3, im Auftrage des Kultusministeriums Rheinland-Pfalz, S.42: Sortieralgorithmen, Autor Reimut Schmitt,Gymnasium Kirn.

Ein Ausschnit aus dem Programmlauf zeigt die Idee:
1) Vergleich zweier benachbarter Zahlen, die durch Pfeile markiert werden.
2) Wenn die linke Zahl kleiner ist als die rechte Zahl erfolgt ein Austausch der beiden Zahlen, optisch durch Wanderung der beiden Zahlenkästchen.
3) Die beiden Pfeile wandern um eins nach rechts bis sie am rechten Ende angekommen sind. Dort steht dann die für einen Durchgang jeweils größte Zahl.
4) Nach (hier) 9 Durchläufen ist Schluß, vorher geht es stets bei 1) weiter.
Farbige Markierungen unterstützen die Abläufe.





Da das Verfahren leicht verständlich ist, wird man in diesem Fall die Demonstration nicht im Nachhinein zum besseren Verständnis einsetzen, sondern **das Verfahren ohne besondere Kommentare an Hand der Demonstration selbständig erarbeiten lassen**. - Der Leser überlege sich, ob ein solches Vorgehen erfolgreich ist.

Zielsetzung der Besprechung im Seminar

Die Referendare (Lehrer) sollen

- Vor- und Nachteile des Einsatzes von Demo-Programmen in dem gerade betrachteten Gebiet einschätzen können,
- Demo-Programme angemessen zur Erreichung von Lernzielen einsetzen können,
- unter Beachtung gewisser Kriterien Demo-Programme selbst konstruieren können.

Weitere Aspekte der Seminararbeit beim Thema "Demo-Programme"

Die Bearbeitung des Themas beginnt mit einer Brainstorming-Phase in Gruppen. Danach werden gemeinsam Fragen in Zusammenhnag mit Demo-Software formuliert.

Wir machen zunächst eine Einschränkung auf Demonstrationsprogramme in Bezug auf Lernziele des Informatikunterrichts und meinen ganz grob solche Programme, die in Demonstrationsform die Erreichung solcher Lernziele fördern. Beispiel: Ein Programm, das dem Schüler hilft, das Backtracking-Verfahren zu verstehen. Eine Abgrenzung gegenüber "Simulationsprogrammen" wird nicht versucht.

Bei einigem Nachdenken ergibt sich eine umfangreiche Fragenliste:

- Wer konstruiert die Demo-Programme?
- Wie sollten Demo-Programme aufgebaut sein?
- Für welche Informatik-Gebiete sind sie besonders geeigent?
- In welchen Fällen sollte man sie einsetzen?
- Wie geht man methodisch vor?
- Sollen Schüler selbst Demo-Programme herstellen?
- usw.

Wir wenden uns der Frage zu:

Wie sollten Demo-Programme aufgebaut sein?

Zur Beantwortung der Frage ist es hilfreich, die Oberfläche einiger bereits existierender **Demo-Programme** zu **analysieren** und sie auf Einsatzfähigkeit hin zu prüfen.

Der zu erlernende Inhalt muß passend in Teile zerlegt und *in anschaulischer Form* präsentiert werden. Ein wichtiges Prinzip ist das der *Zwischenausgaben* während des Ablaufs. Die Veranschaulichung kann erfolgen durch *Graphiken*, durch (dynamische, filmartige) *Darstellung von Abläufen*, durch *passenden Text* z.B. für Zwischenausgaben, durch (wenige) *Farben*. Achtung: *Keine Überfrachtung* des Bildschirms! Es ist wichtig, sich auf den *Kern der Sache* zu konzentrieren. Häufig ist es nützlich, wenn *Abläufe wiederholbar* sind. Falls mehrere Bildschirmseiten (oder Aktionen) benutzt werden, sollte man *zur vorhergehenden Seite (Aktion) zurückkehren* können. Es bleibt zu prüfen, ob weitere *Eingriffe der Benutzer* erfolgen dürfen, die den Demo-Ablauf steuern.
Der Einsatz von Demo-Programmen ist besonders dann sehr kritisch zu hinterfragen, wenn der gleiche Inhalt auch von Hand demonstriert werden kann.

Es ist hilfreich einige Demo-Programme auf die formulierten Kriterien hin zu untersuchen und ihre Qualitäten einzuordnen.

Für welche Informatik-Gebiete sind Demo-Programme besonders geeignet?

In der Regel sind es Gebiete, die einen gewissen Schwierigkeitsgrad aufweisen. Schließlich muß erst das Bedürfnis nach besserem Verständnis und Veranschaulichung geweckt werden. Die Demonstration wird sich dann in den meisten Fällen auf Beispiele zur allgemeinen Problemstellung beziehen! Einige Beispiele für "Demo-freundliche Gebiete":

- Backtracking-Algorithmen, z.B. 8-Damenproblem,
- Rekursionen, z.B. Größter gemeinsamer Teiler,
- Sortierverfahren, z.B. Quicksort,
- Abläufe im Computer, z.B. Modellrechner,
- Verschlüsselung nach dem RSA-Verfahren,
- Automaten, z.B. Simulation eines Getränkeaustomen.

Mit welchen Zielsetzungen kann man ein fertiges Demo-Programm einsetzen?

- Es soll helfen, schwierige Algorithmen / Vorgänge selbständig zu erarbeiten oder besser zu verstehen.
- Das Demo-Programm kann dazu dienen, Ideen zu erzeugen bzw. solche zu überprüfen.
- Das Demo-Programm dient der Übung zu einem schon bekannten Inhalt.

- Das Demo-Programm kann auf seine Qualität hin untersucht werden, wenn die Inhalte vorher schon über andere Wege erarbeitet wurden (Lernziel: Bewerten können!).
- Das Demo-Programm kann dazu dienen, andersartige Darstellungsformen kennenzulernen.
- Es kann Vorstufe zur eigenen Programmierung eines Demo-Programmen sein, beispielsweise für spätere Schülerjahrgänge.

Zusammenfassung:

Die oben teilweise mehr plakativ formulierten Aussagen sollten genügen, um auch Außenstehenden einige Aktivitäten in Informatik-Seminaren zu verdeutlichen. Die LOGIN-Rubrik "Praxis und Methodik - Werkstatt" wird weiterhin über solche Aktivitäten berichten!

Wir wenden uns hier der letzten Frage zu.

Ziele bei der Konstruktion von Demo-Programmen durch Schüler für (andere) Schüler

1) In diesem Fall müssen die Schüler selbstverständlich die Inhalte, die demonstriert werden sollen, kennen.
2) Diese Forderung schließt nicht aus, daß die Schüler ihr eigenes Wissen durch die Beschäftigung mit der Demonstration vertiefen oder gewisse Sachverhalte besser verstehen als vorher.
Die Schüler
2) müssen entscheiden, welche Inhalte ihre "Kunden" (z.B. Schüler eines Kurses im nächsten Schuljahr) begreifen sollen.
4) lernen Modellbilden und Wesentliches vom Unwesentlichen zu trennen.
5) finden Methoden der Veranschaulichung von Sachverhalten.
6) lernen den Entwurf einer auf einen bestimmten Zweck hin ausgerichteten Benutzeroberfläche
6) erweitern bei der Realisierung ihres Planes ihr Wissen in der verwendeten Programmiersprache.