
*Automatische Überprüfung
und Hilfestellung zu
Vorlesungs-begleitenden Übungen
(Stand des Projekts im Juni 2004)*

Norman Hendrich, Klaus von der Heide

Universität Hamburg, Fachbereich Informatik (TAMS)

Vogt-Kölln-Str. 30, D 22527 Hamburg

hendrich@informatik.uni-hamburg.de

`tams-www.informatik.uni-hamburg.de/forschung/interaktives-skript/`

Übersicht

- Das interaktive Skript
- Überprüfung und Hilfestellung zu Übungsaufgaben

Darstellung der interaktiven Skripte?

- proprietärer Viewer (mscriptview)
- XHTML-Viewer (ElearningEditorKit)
- HTML+Applet+Server Konzept

Diskussion

- coming soon: Überprüfung von mathematischen Formeln

Ausgangssituation

- T-Lehrstoff gilt als schwer
- geringes Interesse vieler Studenten

"Augen zu und durch"-Ansatz:

- Vorlesung anhören, aber kaum nachbereiten
- sehr schlechte aktive Beteiligung an den Übungen (nur T1/T2)
- Praktikum als Nachhilfekurs nutzen
- Klausur/Prüfung versuchen (man darf ja mehrmals)

=> Übungen direkt in die Vorlesung/Skript integrieren

=> und zwar mit interaktiven Hilfsmitteln ("Applets")

=> sofortige automatische Überprüfung der Lösungen

=> Hilfestellungen zu Lösungsansätzen, soweit möglich

Ziele

- Übungsaufgaben im Skript integriert:

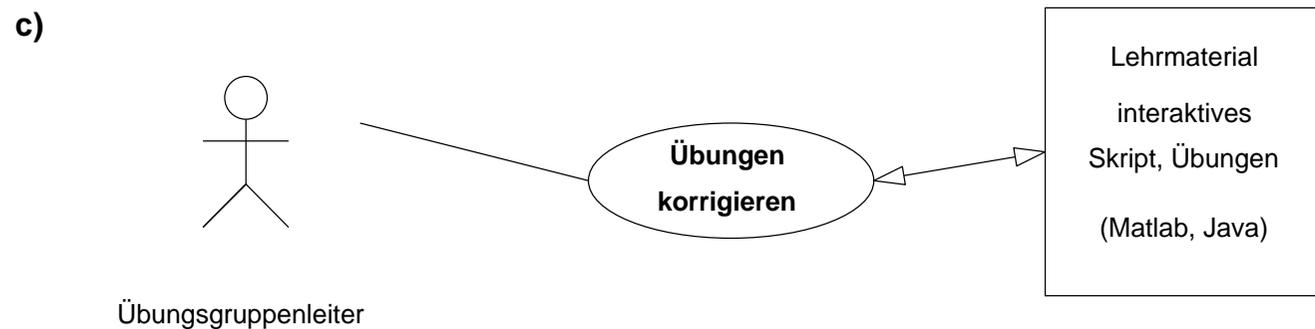
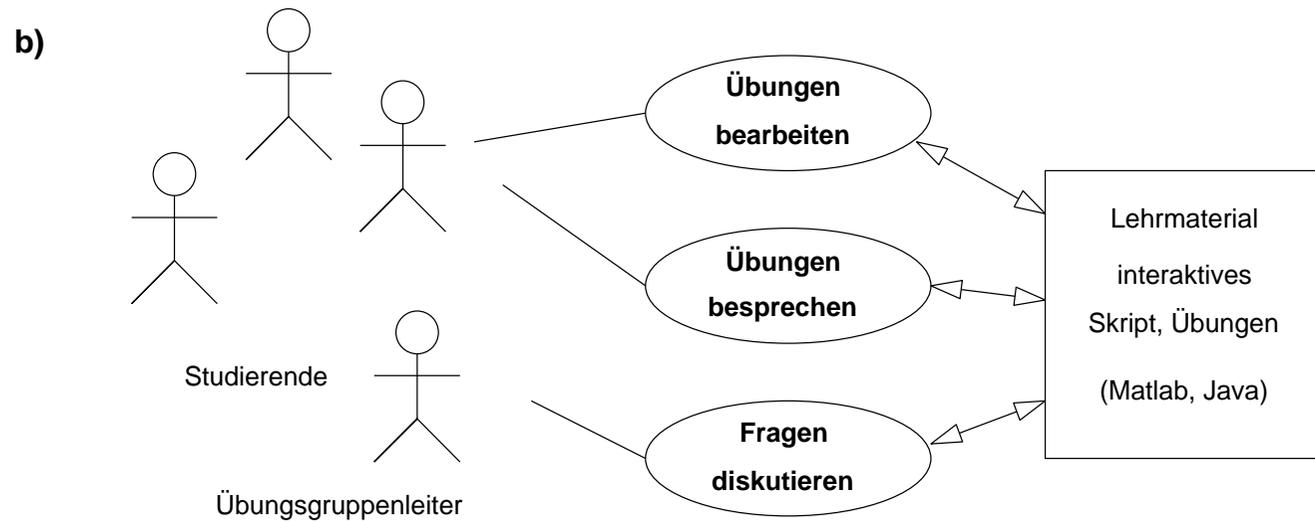
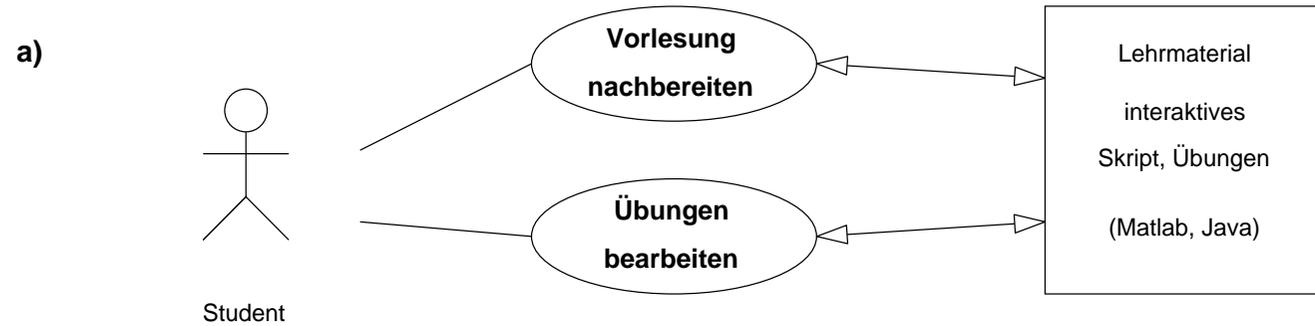
Unterstützung der Studierenden:

- geringere Hemmschwelle zur Bearbeitung der Aufgaben
- automatische Überprüfung der Lösungen
- sofortiger Feedback (nicht erst eine Woche später)
- kontextabhängige Hilfestellungen
- gezielte Gegenbeispiele helfen bei der Fehlersuche

Unterstützung der Übungsgruppenleiter:

- erleichtert das "Ausprobieren" während der Übungsstunden
- automatische (Vor-) Korrektur vieler Aufgaben

Use-Cases



Das interaktive Skript

bzw. "interaktives Lehrbuch"

- erläuternde Texte, eingebettete Formeln
- eingebettete Medien: Graphiken, Animationen, Audio, Video
- Hyperlinks, Verweise im Skript, auf Webseiten, auf externe Programme

- eingebettete aktive Elemente (z.B. Matlab/Jython-Funktionen)
- eingebettete aktive Applets (mit eigener GUI)
- eingebettete Übungen (mit sofortiger Überprüfung)

- jederzeit erweiterbar (auch von den Studierenden)
- auch später im Berufsleben produktiv nutzbar

- einfache Content-Erstellung

Eingebettete Skripte:

- Beispiel-Code im Matlab-Browser: `t1_3_1.m`

```
% Die folgende Funktion bietet eine interaktive Demo für die  
% drei Formate nach IEEE-754:
```

```
demoieee754
```

```
% Werden Operationen durchgeführt mit der Repräsentation für  
% {\fontname{Courier}}NaN}, so ist das Resultat immer ...  
% ...
```

```
inf/(-1+1) % liefert das Resultat +inf
```

```
inf/-(1-1) % liefert das Resultat -inf
```

- markierter Code kann in den Editor kopiert werden
- Experimente mit anderen Parametern
- Erweiterung der bestehenden Funktionen
- einfache Content-Erstellung durch "Kommentar-Trick"

Matlab:

"Matrix Laboratory"

- System für rechnergestützte Mathematik
- entwickelt seit ~1984, www.mathworks.de
- Schwerpunkt auf technischen Anwendungen

- einfache Syntax für Vektor-/Matrixoperationen
- numerische und symbolische Algorithmen

- "workspace" IDE mit Editor und Debugger
- "handle-graphics" objektorientierte Graphik/Plots
- "toolboxes" anwendungsspezifische Erweiterungen
- "simulink" graphische Modellierung
- "system-level design" Codeerzeugung für DSPs / FPGAs

Matlab: Codebeispiele (1)

```
x = 0 : pi/100 : 2*pi;    % Vektor mit 200 Elementen
y = sin( x );            % ditto
plot( x, y );           % 2D-Funktionsplot, autoscale
xlabel( '0 : 2\pi' );    % x-Label, TeX-Annotation
...

% Samplerate, Tonfrequenz, Bits pro Minute
fs = 4000; frequenz = 800; bpm = 160; string = 'ELCHTEST';
morse = morsecode( string, round(fs/10*bpm/60)*[1 3 1 3 4]));
envel = filter( fir1( 60, bpm/f2 ), 1, morse );
tone  = envel .* sin(2*pi*frequenz*(1:length(envel))/fs);
sound( tone, fs );
```

- Demo (morsesound aus t1_4.m)

Matlab: Codebeispiele (2)

```
A = [1,1,1; 1,2,3; 1,3,6]; % Pascal(3) Matrix
b = [3; 1; 4];           % Spaltenvektor
x = A \ b;              % Lösung des Gleichungssystems Ax=b
```

...

```
s = '((x-2).^2 - 5)';    % String
f = inline( s );        % Inline-Funktion
v = feval( f, 0.3 );    % Auswertung f(0.3)
z = fzero( f, 0.1 );    % sucht Nullstelle nahe x=0.1
m = fminbnd( f, 0, 4 ); % Minimum in [0..4]
fplot( f, [0, 6] );    % Funktionsplot
```

...

- alle gängigen Matrixoperationen verfügbar
- numerische und symbolische Operationen

Matlab: Java-Interface

seit Matlab 5.3 auch Zugriff auf Java-Objekte:

```
java on;                                % für Matlab 5.3

generator = java.lang.Random % Konstruktor
x = generator.nextDouble           % Methodenaufruf

editor = hades.gui.Editor          % Hades-Editor
editor.doOpenDesign( 'test.hds', 0 );
editor.getSimulator.runFor( 3.0 ); % simuliert bis t=3.0 sec.
...
```

- läuft auch mit 5.3 Student Edition (einige Einschränkungen)
- Konfiguration über CLASSPATH

Matlab: Akzeptanz-Problem?

bisherige Version der interaktiven Skripte:

- vollständig in Matlab implementiert
 - proprietärer Browser zur Darstellung erforderlich (mscriptview)
 - ansonsten Skripte nur mühsam lesbar
-
- Arbeiten mit dem Material erfordert Matlab und Lizenz
 - nicht genügend Lizenzen (floating-licence) für alle Studierenden
 - unterschiedliche Ausgangslage für Grund-/Hauptstudium
- => Aufbereitung des Materials für "Standard"-Viewer ?!
- => Erstellung neuen Materials auf Basis "freier" Plattformen ?!
-
- => Umsetzung der Skripte nach HTML

Drei Alternativen:

proprietärer, selbstentwickelter Viewer

(mscriptview)

- volle Integration in die Plattform
- Standardfunktionen müssen bereitgestellt werden

(z.B. Matlab)

(z.B. Drucken)

Standard-Web-Browser

(Internet Explorer, Mozilla, ...)

- Einbindung der Skripte über Applets / Plugins
- Anwender haben ihre gewohnte Umgebung
- Vernetzung mit externen Webseiten / Plattformen
- Dokumentenformat liegt fest

(Java, Flash)

(z.B. Bookmarks)

(z.B. WebCT)

(HTML)

modifizierter Web- oder XML-Browser

(Swing HTMLToolkit)

- Erweiterungen des Dokumentenformats möglich

(z.B. <matlab>)

EditorKit: Features

- Java Swing (JDK 1.2+) umfaßt Texteditor-Framework
- javax.swing.text.*
- Packages und Klassen für Unicode / RTF / HTML

- Parser ist HTML 3.2 kompatibel
- auch als Editor nutzbar (HTMLWriter)
- kein JavaScript
- keine Applets oder Plugins

- über Subklassen flexibel erweiterbar
- Einbettung vorhandener JComponents in die Darstellung

- Doku: "Swing in a nutshell" (extra chapter on website)
- Sun: "API is expected to change"

ElearningEditorKit

OO-Konzept erlaubt Erweiterungen von EditorKit:

- abgeleitete, erweiterbare Klassen schreiben
- und über factory-Mechanismus instantiieren
- Beispiel für zusätzliche HTML-Tags

```
<applet src="applet-url" ...> <param /> ... </applet>
```

```
<tex>Euler-Formel:  $e^{i\pi} + 1 = 0$ </tex>
```

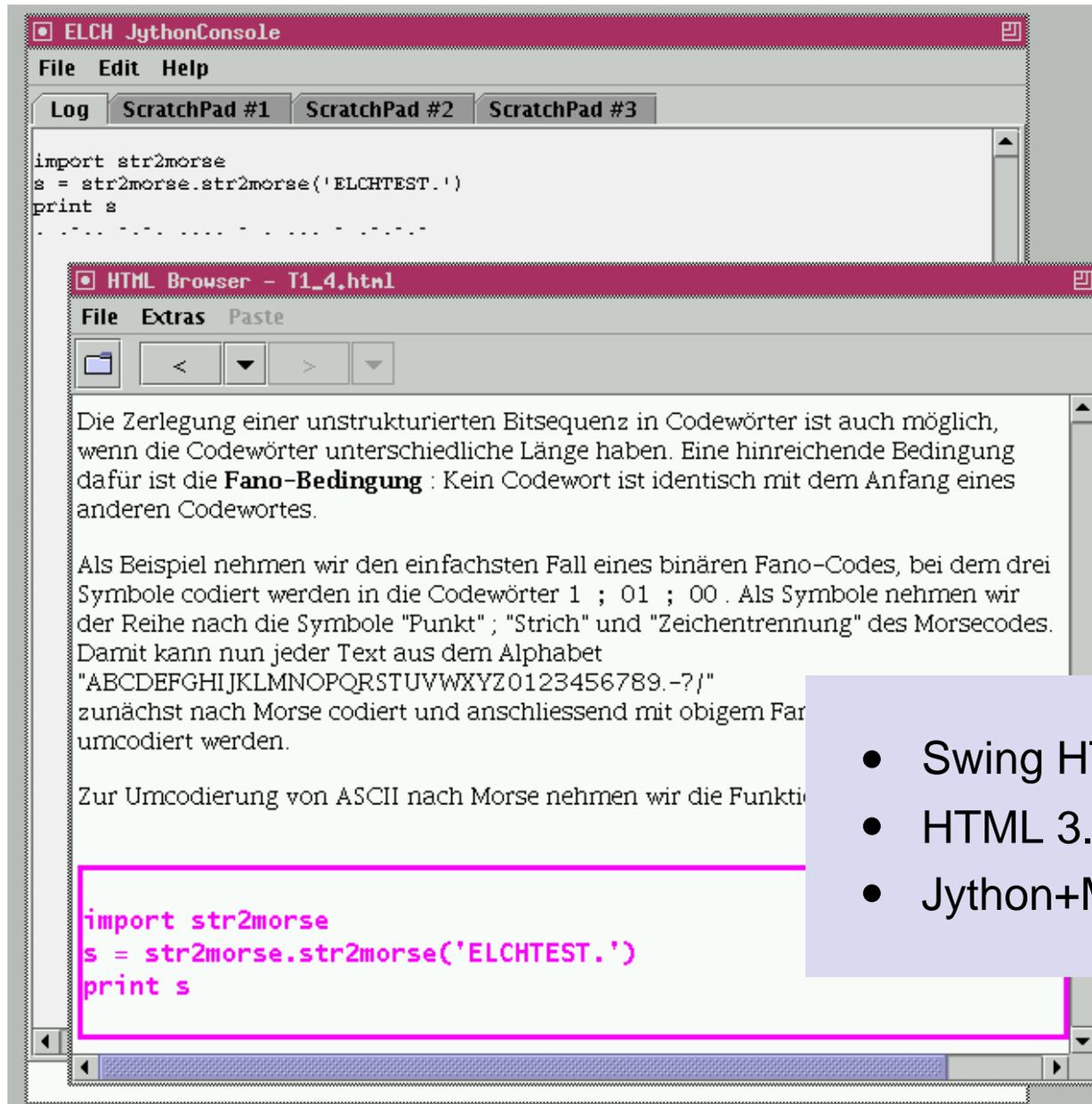
```
<fig src="figure-url" width=600 height=400> </fig>
```

```
<jython> ... </jython>
```

```
<matlab> ... </matlab>
```

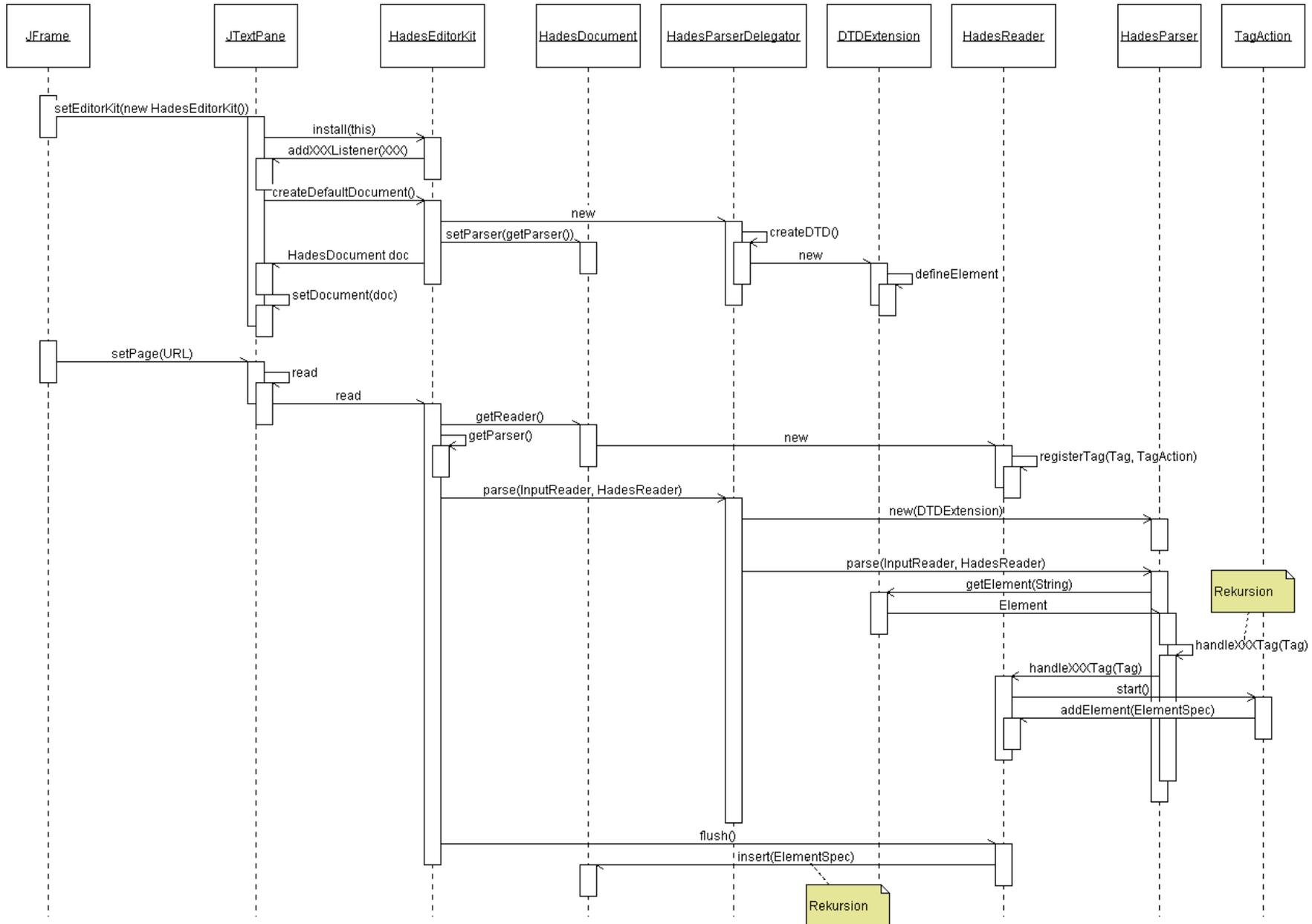
- Prototyp implementiert und getestet
- enorme Komplexität / aber keine brauchbare Doku
- diverse Ärgernisse (Drucken, kein xhtml, kein Javascript, ...)
- Beispiel:

ElearningEditorKit: Prototyp



- Swing HTMLEditorKit
- HTML 3.2
- Jython+Matlab+FIG+TeX

ElearningEditorKit: Parsing ...



ElearningEditorKit: Status und Fazit

Swing-Bibliothek enthält HTML-Komponente

(HTMLEditorKit)

Vorteile:

- HTML 3.2 kompatibel, auch als Editor
- flexibel erweiterbar (aber schlecht dokumentiert)
- unverselle Lösung für alle Plattformen (write once: PDAs?)
- Prototyp unterstützt Applets, Jython und Matlab-Skripte

Nachteile:

- Startup langsam (Seiten werden komplett geparkt)
- Drucken nur eingeschränkt (z.B. keine Seitenumbrüche)
- Probleme mit existierenden Seiten (kein XHTML, kein Javascript)
- wenig fehlertolerant (bei fehlerhaften Seiten)
- keine GUI und Utilities (z.B. Bookmark-Verwaltung)

=> Ansatz (erstmal) nicht weiter verfolgt

HTML+Applets+Server: Idee

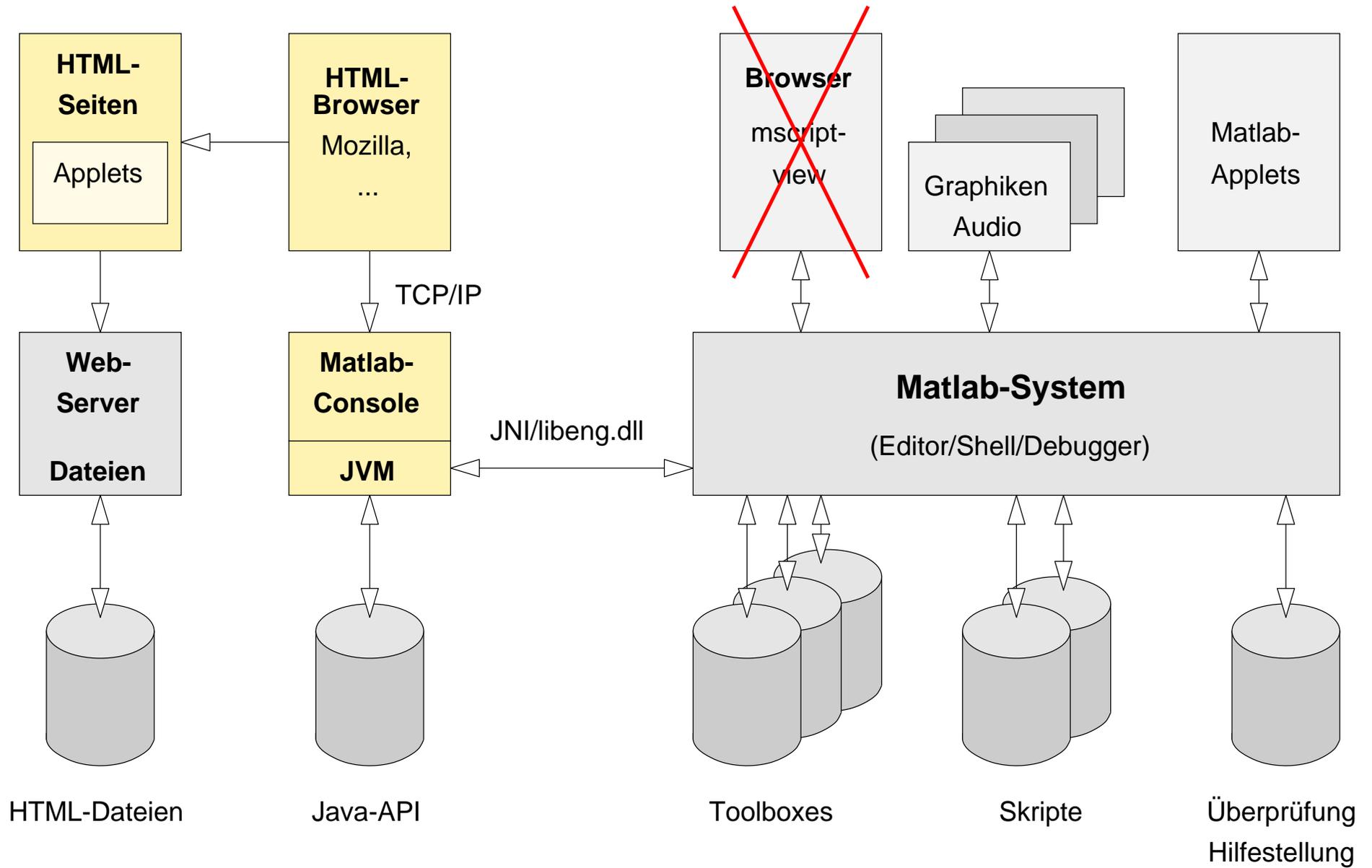
- Skript-Texte nach HTML umsetzen
- interaktive Codeblöcke als eingebettete Java-Applets
- Ausführung über zusätzlichen Serverprozess ("Console")
- wichtig: Texte sind auch ohne Matlab gut lesbar

Aufgaben der Applets:

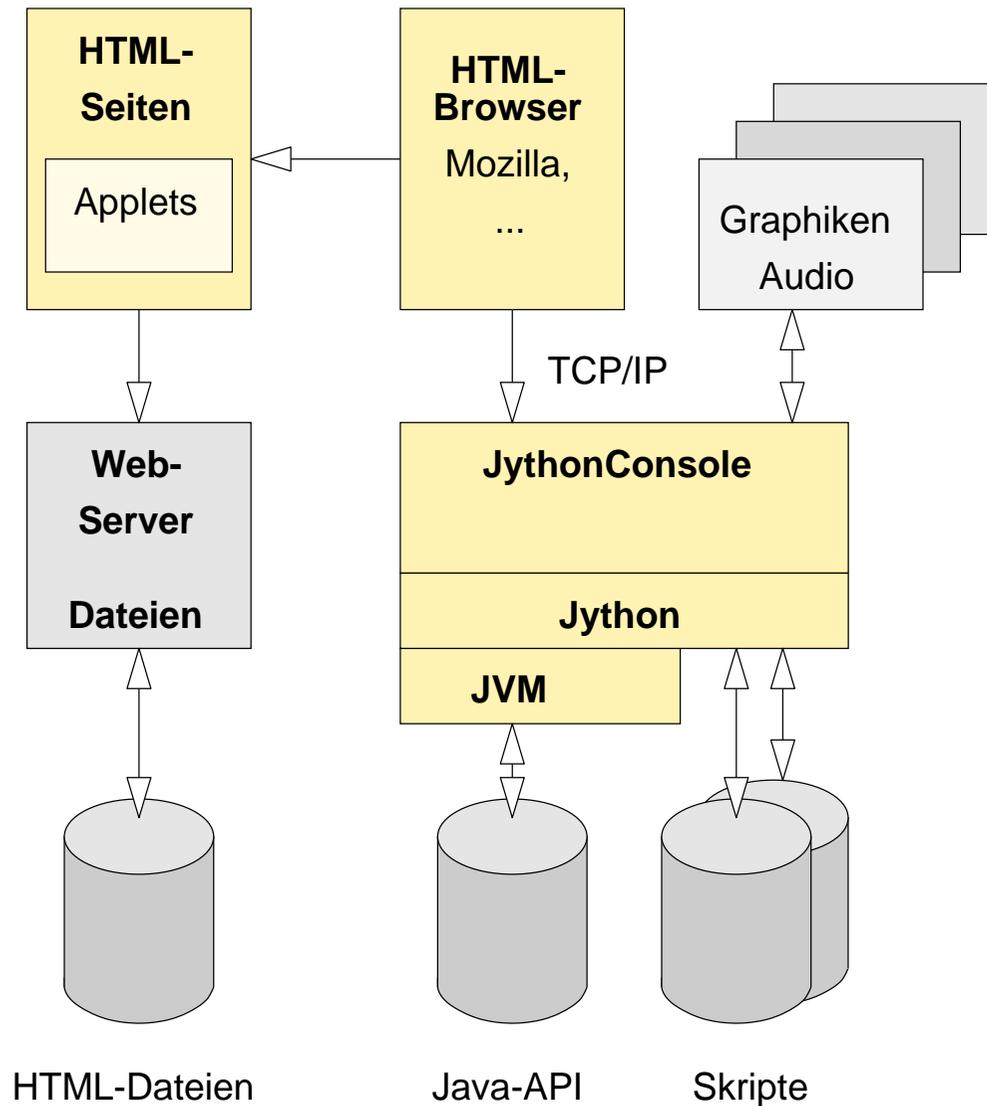
- Darstellung des ausführbaren Codes
- Ereignisbehandlung: Hervorheben, Ausführen, Editieren
- Übergabe des Codes an die jeweilige Console
- Applet benötigt nur Netzwerkzugriff auf localhost

- Serverprozess läuft als Applikation mit vollen Rechten
- Matlab-"Engine", Jython+JVM, Webserver, Datenbanken, ...

HTML+Applets+Server: Matlab



HTML+Applets+Server: Jython



- Codeblöcke als Jython-Skripte und Funktionen
- voller Zugriff auf Java-API inklusive Audio-/Video
- Java-Variante der Matlab "HandleGraphics" Funktionen
- bisher nur wenig Content

HTML: Codebeispiel

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<a href="t1_3_3.html"></a> ...

<h1><font color="Silver">
4. Codierung
</font></h1>

<p>
<b>Def.</b> Unter <b>Codierung</b> versteht man das Umsetzen
einer vorliegenden Repräsentation A in eine andere Repräsentation B .
Die Interpretation von B nach A muss eindeutig sein.
Ist sie auch umkehrbar eindeutig, so spricht man von
einer <b>Umcodierung</b> .
</p>

<p>
Ein einschrittiger Code mit 12 Wörtern ist z.B. der folgende
</p>
<p>
<applet code="de.uni_hamburg.informatik.tams.elearning.applets.MatlabApplet" width="510" ...>
<param value="n = 12;" name="line1">
<param value="binmatout(num2binmat(einschritt(n)',ceil(log2(n))))" name="line2">
<param value="2" name="nlines">
</applet>
</p>

</body>
</html>
```



```
n = 12;
binmatout(num2binmat(einschritt(n)',ceil(log2(n))))
```

T1-Skript in HTML (1)

The image shows two screenshots of a Mozilla browser window. The left screenshot displays a table of contents for a document titled 'Technische Informatik I' by Klaus von der Heide. The right screenshot shows a specific page titled '3.3 Rundungsfehler' (Rounding Errors), which discusses the non-associative and non-distributive properties of floating-point arithmetic. A callout box on the right side of the second screenshot lists HTML features used in the document: formatted text, hyperlinks, navigation, and formulas.

Technische Informatik I
Klaus von der Heide
Universität Hamburg, Fachbereich Informatik, e-mail: v.d.heide@on-line.de

[Vorwort](#)

- [1. Einführung, Grundbegriffe](#)
- [2. Repräsentation von Zahlen](#)
- [3. Gleitkommazahlen](#)
- [4. Codierung](#)
- [5. Kanal- vs. Quellen-Codierung](#)
- [6. Algebra](#)
- [7. Schaltfunktionen](#)
- [8. Schaltnetze, Zeitverhalten](#)
- [9. Schaltwerke](#)
- [10. Flip-Flops](#)
- [11. Getaktete Schaltwerke](#)
- [12. Grundbausteine der Register-Transfer-Ebene](#)
- [13. Von-Neumann-Architektur](#)
- [14. Maschinen- und Assembler-Programme](#)

3.3 Rundungsfehler

Die Operationen mit Gleitkommazahlen sind im allgemeinen nicht informationstreu. Überdies gelten weder das Assoziativ- noch das Distributivgesetz. Wir zeigen dies der Anschaulichkeit halber wieder in dezimaler Schreibweise mit 4-stelliger Mantisse und nehmen hier an, dass bei der Normalisierung und Denormalisierung überzählige Stellen einfach ohne Rundung fortfallen.

Assoziativ-Gesetz

Mit den drei Zahlen

$$a = 9.857 \cdot 10^{-3}$$
$$b = 9.743 \cdot 10^2$$
$$c = -9.742 \cdot 10^2$$

ergibt sich

$$a + b = b$$

weil wegen des Exponenten die Man

$$(a + b) + c = b + c = 0.001 \cdot 10^2$$

- Text (formatiert)
- Hyperlinks <href>
- Navigation <href>
- Formeln (MathML?)

T1-Skript in HTML (2)

Previous Up Next

4. Codierung

Def. Unter **Codierung** versteht man das Umsetzen einer vorliegenden Repräsentation A in eine andere Repräsentation B. Häufig liegen beide Repräsentationen A und B in der selben Abstraktionsebene. Die Interpretation von B nach A muss eindeutig sein. Ist sie auch umkehrbar eindeutig, so spricht man von einer **Umcodierung**.

Die folgende Tabelle zeigt eine Reihe mehr oder weniger gebräuchlicher binärer Codierungen für Dezimalziffern.

Ziffer	BCD	Gray	Exzess3	Gray-	Aiken	biquinär	1-aus-10	2-aus-5	CCIT-2
0	0000	0000	0011	0010	0000	000001	0000000001	11000	01101
1	0001	0001	0100	0110	0001	000010	0000000010	00011	
2	0010	0011	0101	0111	0010	000100	0000000100	00101	
3	0011	0010	0110	0101	0011	001000	0000001000	00110	
4	0100	0110	0111	0100	0100	010000	0000010000	01001	
5	0101	0111	1000	1100	1011	100001	0000100000	01010	
6	0110	0101	1001	1101	1100	100010	0001000000	01100	
7	0111	0100	1010	1111	1101	100100	0010000000	10001	
8	1000	1100	1011	1110	1110	101000	0100000000	10010	01100
9	1001	1101	1100	1010	1111	110000	1000000000	10100	00011

Jede Wandlung von einem Code dieser Tabelle in einen anderen der Tabelle ist eine Umcodierung. Welcher Code vorliegt, geht nicht aus den Codewörtern hervor.

Def. Werden zur Repräsentation B Wörter eines Zeichenvorrats Z verwendet, so bezeichnet man diese als **Codewörter**.

- Tabellen: `<table>`
- Abbildungen: ``
- Zeilenumbrüche automatisch
- Stylesheets optional

T1-Skript in HTML (3)

Kanalkapazität

Die Informationstheorie wurde entwickelt als Theorie zu deren Fehlerverhalten sich aber in einem stochastischen Kanal ohne längliche Herleitung die sog. Kanalkapazität C an.

$$C = 1 - H(P)$$

Hierin ist $H(P)$ die Entropie des Fehlerverhaltens.

Der binäre symmetrische Kanal sei festgelegt durch folgende Parameter:

- (1) Die Wahrscheinlichkeit der beiden Symbole 0 und 1
- (2) Die Wahrscheinlichkeit P , dass aus einer 0 eine 1 wird
- (3) Die Wahrscheinlichkeit eines Fehlers an der Binärstelle

Damit ist $H(P) = -P \log_2(P) - (1-P) \log_2(1-P)$

Die Kanalkapazität des binären symmetrischen Kanals ist:

```
P = 0:0.001:1; % Abszisse mit 1001 Punkten
q = [P; 1-P]; % Matrix der Wahrscheinlichkeiten
C = 1 - sum( q .* log2( 1./q )); % Vektor der zugehörigen Kanalkapazitäten
fh = fullscreen('Kanalkapazität'); % Erzeugung eines Graphikfensters
plot(P,C)
title('Kapazität des binären symmetrischen Kanals')
xlabel('Bitfehlerwahrscheinlichkeit')
ylabel('Bits pro Binärstelle')
grid on
```

Offenbar ist die Kanalkapazität 0 bei $P = 0.5$.
Zwei verschiedene Bitsequenzen unterscheiden. Dass die Kanalkapazität
Inversion aller Bits eine Sequenz erzeugt, für die die Kanalkapazität
Die Kanalkapazität ist eine obere Schranke, die eine praktisch fehlerfreie Übertragung möglich wird.

Previous Up Next

Applet de uni_hamburg.inf

Kapazität

Bits pro Binärstelle

Bitfehlerwahrscheinlichkeit

Bitfehlerwahrscheinlichkeit (P)	Bits pro Binärstelle (C)
0.0	1.0000
0.1	0.7219
0.2	0.4710
0.3	0.2473
0.4	0.1073
0.5	0.0000
0.6	0.1073
0.7	0.2473
0.8	0.4710
0.9	0.7219
1.0	1.0000

- aktive Elemente als Applets
- Ausführen über Doppelklick
- MatlabConsole als Editor
- nur Windows: Matlab Workspace

DSP-Skript in HTML (4)

Mozilla

file:///tmp/eclipse/workspace/elch-xxx/applets/dsp/dsp_13_7.html

Previous Up Next

13.7 Vermessung von Multiratenfiltern

Amplituden- und Phasenfrequenzgang ger Multiratenfilter. Insbesondere möchte man die Unterdrückung der Aliassignale haben Inputfrequenz das Spektrum des Ausgabe über jeder Inputfrequenz (x-Achse) das Sp Möchte man nur wissen, ob die Spezifikat längs der Geraden $y = x$. Dadurch wird rechts und links davon, weil ihre Frequenz allerdings verwirrend, da zwei Achsen über

Wir betrachten zunächst als einfaches Be

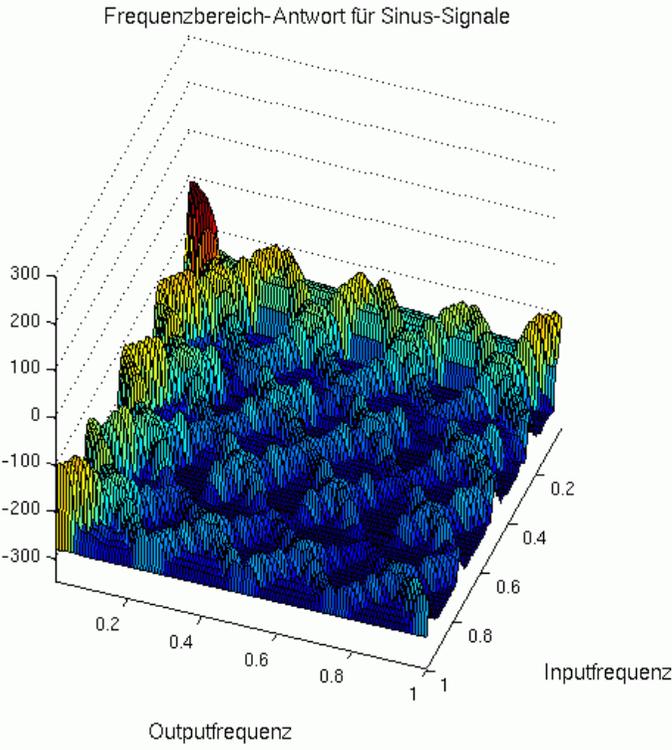
```
[koef,srate,dly] = mrdesign(0.05,0.03,1,0.03,1);  
freqscan('mrfilter(p{1},p{2},x)',{koef;srate;dly});
```

Der Blick kann durch die Maus und die Tastatur durch das Aliasdiagramm von oben.

Im ersten Beispiel wurde absichtlich der ungewöhnliche Dezimationsfaktor 0.03 verwendet, um praktisch nur die Wirkung der Aliasdämpfung der Interpolationsfilter gleich zu betrachten.

```
[koef,srate,dly] = mrdesign(0.05,0.03,1,0.03,1);  
freqscan('mrfilter(p{1},p{2},x)',{koef;srate;dly});
```

Die drei Bilder zeigen im Vergleich die Wirkung der Aliasdämpfung. Ein Beispiel für ein Multiratenbandpassfilter ist das folgende:



ELCH MatlabConsole

File Edit View Help

Temp. koef srate dly

Co1

Applet de.uni_hamburg.informatik.tams.elearning.applets.MatlabApplet started

Woher kommen die HTML-Seiten?

- Skripte liegen als Matlab-Dateien vor
- Vorlesungen DSP, NT, T1, T2, (Teile) Praktikum T4

=> Konverterprogramm erzeugt daraus HTML-Dateien:

Text mit Formatierungen:

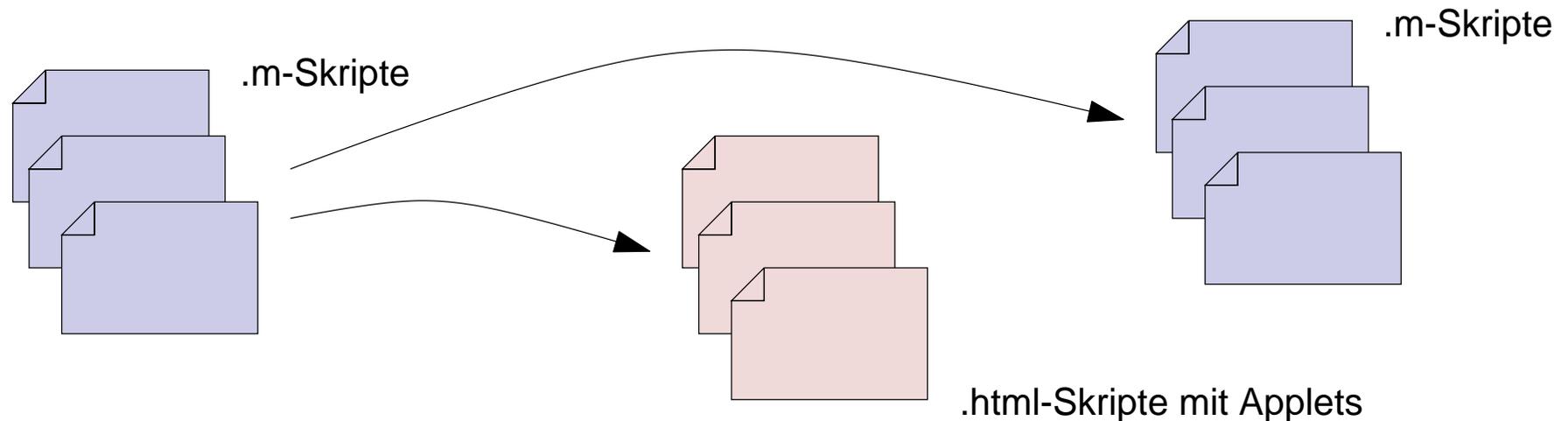
(Schriftart, Schriftgröße, Farben)

- Formeln, Sub/Superskripte, Symbole (LaTeX-Notation)
- Navigation, Hyperlinks (Inhalt, prev/next/up)
- Einbettung von Medien: Bilder, Videos, Animationen

Java-Applets als interaktive Elemente:

- Ausführung über MatlabConsole und Workspace
- andere Skriptsprachen möglich
- keine besonderen Sicherheitseinstellungen nötig

Matlab2HTML Konverter:



- je eine HTML-Datei pro Matlab Skriptdatei
- je ein Applet pro Matlab-Codeblock
- Hyperlinks zur Navigation
- Dateibaum mit .m-Skripten, Matlab Pfad-Initialisierung
- Heuristiken zur Erkennung und Umsetzung von Tabellen sowie Formeln als HTML oder Applets

- Details: Vortrag von A.Ruge

(OS TAMS, 22.06.2004, 16:15, F334)

HTML Content-Erstellung

vorhanden: Konverter von Matlab nach HTML (s.u.)

aber: Applet+Server-Konzept ist universell

jeder andere HTML-Editor reicht aus, um Skripte zu erstellen:

- geeignete Texteditoren
- HTML-Editoren (emacs, vi, nedit)
- Content-Management-Systeme (Mozilla, HoTMetaL)
- ...

weitere / vorhandene Konverter ebenfalls nutzbar:

- Beispiel: DA Y.Sünneli (OS TAMS, 29.06.2004, 16:15, F334)
- LaTeX -> Docbook -> HTML+Applets

Zusammenfassung:

- Konzept für HTML-Version der interaktiven Skripte
- Zwei Architekturen als Prototypen erprobt:
 - ElearningEditorKit auf Basis von Java2/Swing
 - Standard-HTML mit eingebetteten Applets
- HTML+Applet+Server-Konzept funktioniert gut
- auch für andere Plattformen als Matlab (Jython, PHP, ...)
- Integration in E-Learning Portale möglich (WebCT, ...)

- Konverter erlaubt vollautomatische Umsetzung
- bisher konvertiert: Vorlesungen T1, DSP, NT
- Installer in Vorbereitung (install4j)

- Fragen, Wünsche, Anregungen?