Eine tutorielle Mechanik-Aufgabenumgebung mit Maplets

Burkhard Alpers – balper@fh-aalen.de
Fachhochschule Aalen
FB Maschinenbau

Abstract: In diesem Beitrag wird eine Mechanikaufgabenumgebung vorgestellt, die im wesentlichen Statikaufgaben enthält. Die Umgebung ist im CAS Maple implementiert und bietet spezielle tutorielle Prozeduren für das Checken von Zwischenergebnissen und gestufte Fehlerhinweise, so dass der Nutzer Information und Motivation zur eigenständigen Weiterarbeit erhält. Die Realisierung mit Maplets erlaubt eine einfache Nutzung ohne tiefere Kenntnis der Maple-Syntax.

1. Einleitung

Beim Lösen von Aufgaben aus der Technischen Mechanik besteht häufig das Problem, dass die Studenten ins Stocken geraten und dann Musterlösungen zu Rate ziehen, wobei das Trainieren der eigenständigen Problemlösefähigkeit zu kurz kommt. Zwar wird teilweise noch versucht, über ein partielles „Zudecken“ des Lösungswegs nur eine Teilantwort zu erfassen, um dann selbst weiterzudenken, aber dieses Vorgehen ist natürlich unbefriedigend. Der Student sollte vielmehr bei einzelnen Schritten unterstützt werden und dann die Chance und Motivation haben, eigenständig weiterzuarbeiten.

Sogenannte „Intelligente Tutorsysteme“ (ITS, vgl. [4] für den Mathematikbereich) setzen sich zum Ziel, individuelle Unterstützung beim Lernen und Trainieren von Modellen, Konzepten und Verfahren zu bieten. Voll ausgebildete Tutorsysteme bestehen aus einem Studentenmodell, in dem der momentane Wissenszustand des Studenten gespeichert wird, einem Modell der Wissensdomäne, einem Tutormodell, das entsprechende Tutorstrategien implementiert, und einer Kommunikations-schnittstelle zum Benutzer. Die Erstellung eines solchen Systems erfordert einen hohen Ressourceneinsatz. Mit weit geringerem Aufwand lässt sich, beschränkt auf die tutorielle Unterstützung der Aufgabenbearbeitung, eine Lernumgebung realisieren, die das schrittweise Bearbeiten in dem Sinne „intelligent“ unterstützt, dass Studenten je nach Bedarf Vorgehenshinweise abrufen, Zwischenergebnisse überprüfen lassen können und ggf. Fehlerhinweise erhalten.

Der vorliegende Beitrag beschreibt eine solche Umgebung, die mit dem CAS Maple realisiert wurde. Der zweite Abschnitt enthält das grundlegende Konzept für den Aufgabentutor, im dritten Abschnitt wird eine frühere Realisierung auf Worksheet-Ebene vorgestellt, die bereits die wesentlichen tutoriellen Prozeduren beinhaltet, diese aber in recht komplexer Weise zur Verfügung stellt. Wie mit Maplets, dem in Maple 8 bereitgestellten Tool zur Erstellung graphischer User Interfaces, diese Komplexität verborgen werden kann, beschreibt der vierte Abschnitt. Abschließend werden einige weitere Entwicklungsperspektiven diskutiert.

2. Konzept für einen Aufgabentutor

Die Bearbeitung von Mechanikaufgaben kann grob in folgende Schritte oder Phasen unterteilt werden:

- *Modellerkennung*: Der Bearbeiter muss zunächst erkennen, um welche „Art“ von Aufgabe es sich handelt, welche Mechanikkonzepte also zur Anwendung kommen. Geht es beispielsweise um Auflagerreaktionskräfte oder Stabfachwerke, so ist ein statisches Gleichgewichtsmodell zu verwenden. Geht es um Bewegungsaufgaben unter Krafteinwirkung, so ist in der Regel das entsprechende Newton’sche Gesetz anzuwenden.
- *Mathematische Modellaufstellung*: Ist das zu verwendende Anwendungskonzept erkannt, so ist das zugehörige mathematische Modell aufzustellen. Hier handelt es sich beispielsweise um ein lineares Gleichungssystem oder eine Differentialgleichung.
- *Berechnungen im Modell*: Im mathematischen Modell sind dann Berechnungen durchzuführen, um die gewünschten Größen zu ermitteln.
- *Interpretation der Ergebnisse*: Schließlich sind die Ergebnisse im Lichte des Anwendungsproblems zu interpretieren, ggf. das Modell zu hinterfragen oder weitere Varianten zu rechnen.

Ein Aufgabentutor sollte diese Schritte so weit wie möglich unterstützen. Allerdings liefert eine CAS-Umgebung nur für Teilschritte eine gute Unterstützung bei der Implementierung, und zwar für die Modellaufstellung und die Berechnung, während andere Schritte lediglich durch recht starre textuelle Hinweise berücksichtigt werden. Im Folgenden sind die wesentlichen Hilfe-Konzepte des vorliegenden Tutors kurz skizziert:

- Für die Modellerkennungsphase werden textuelle Hinweise zur Verfügung gestellt, die – wenn möglich schrittweise – Hinweise auf zu benutzende Modellierungskonzepte geben. Diese können vom Studenten selbst je nach Bedarf abgerufen werden. Es gibt also kein „Lernermodell“ wie in ITS, das als Basis für individuelle Hinweise genutzt würde.
- Für die mathematische Modellaufstellung werden Überprüfungsmöglichkeiten für die Korrektheit angeboten; dies ist natürlich zunächst auf einen begrenzten „Satz“ an mathematischen Modellen beschränkt, wie zum Beispiel lineare Gleichungssysteme. Wird vom Studenten genaueres Feedback als nur eine Korrektheitsangabe benötigt, so stehen gestufte Fehlerhinweise zur Verfügung. Damit wird menschliches Tutorverhalten nachmodelliert, denn ein menschlicher Tutor „verrät“ auch nicht unmittelbar die Lösung oder den Fehler, sondern gibt Hinweise, die den Studenten befähigen und motivieren sollen, eigenständig weiterzudenken. Endet er dennoch in einer „Sackgasse“, so kann er immer noch korrekte Zwischenergebnisse abfragen und auf deren Basis weiterarbeiten. Die Aufgaben-, Aufgabenerstellungs- und -bewertungsumgebung „AIM“ (vgl. [5]), die ebenfalls auf dem CAS Maple basiert, erlaubt bei Mathematikaufgaben bereits das Einfügen von Hilfen, der Schwerpunkt liegt dort aber auf der automatischen Bewertung von Aufgabenseiten, wobei die Kriterien flexibel einstellbar sind.
- Was die Berechnungen im Modell betrifft, so stehen natürlich die bekannten Möglichkeiten eines CAS zur Verfügung. Zudem sollen noch spezielle Tools angeboten werden, die auf gewisse Mechanikberechnungen und Konstruktionen zugeschnitten sind, wie zum Beispiel ein Tool zur Erstellung von Seilecken und Kraftecken oder ein Tool zur Berechnung und Darstellung von

Flächenquerschnittgrößen (Trägheitsmomente, Hauptachsen etc.). Solche Tools liefern ebenfalls ein Feedback, indem sie es dem Studenten erlauben, für eine Vielzahl an Konfigurationen die eigenen Berechnungen und Zeichnungen zu überprüfen.

Das Konzept für den Aufgabentutor legt den Schwerpunkt auf die Überprüfung von Eingaben und auf gestufte Fehlerhinweise.

3. Realisierung mit Maple auf Worksheet-Ebene

Das im vorhergehenden Abschnitt beschriebene Konzept wurde zunächst auf Worksheet-Ebene realisiert und in Form von Hyperlinks zu Hilfetexten, Check-Prozeduren für Zwischenergebnisse sowie Tool-Prozeduren für unterstützende Berechnungen dem Benutzer zur Verfügung gestellt. Das Sachgebiet umfasst dabei die Technische Mechanik I (i.w. Statik) und die Festigkeitslehre I.

Ein Aufgaben-Worksheet enthält als erste Sektion die Aufgabenbeschreibung inklusive Zeichnung (falls verfügbar). Dann werden in der nächsten Sektion die zu benutzenden Variablennamen angegeben, da nur bei deren Verwendung eine Überprüfung der Ergebnisse möglich ist. Dies ist sicherlich ein Nachteil, da damit die wichtigen Größen bereits bekannt gemacht werden. Hätte man eine Zeichnungsoberfläche, in die der Student selbst Vektoren einträgt und benamt, könnt der Nachteil vermieden werden. Solch ein Tool ist aber in Maple nicht enthalten. Im nächsten Schritt werden Hyperlinks zu Hilfetexten angeboten, etwa „Hilfe bei der Vorgehensweise“, „Hilfe bei den Gleichgewichtsbedingungen“ oder ähnliches. Schließlich erfährt der Benutzer, welche mathematischen Objekte überprüft und wie die Check-Prozeduren aufgerufen werden können. Gegebenfalls wird ihm auch noch der Zugriff auf Tool-Prozeduren angeboten. Die dritte Sektion enthält die endgültige Lösung.

Check-Prozeduren gibt es momentan für einen begrenzten Satz an mathematischen Objekten: Lineare Gleichungen/Gleichungssysteme, Ausdrücke, Vektoren, Matrizen und Föppl-Funktionen. Diese Objekte können auf ihre Korrektheit hin geprüft werden; gegebenenfalls wird die fehlerhafte Komponente (Gleichung, Vektorkoordinate, Matrixeintrag) angegeben, z.B.: „Die zweite Gleichung ist falsch“. In einem weiteren Schritt kann dann Information über die Fehlerart und den fehlerhaften Teil abgerufen werden. In [3] wurde beim Einsatz eines ITS zur Physik untersucht, welche Fehler am häufigsten auftreten. Dies sind z.B. Vorzeichenfehler, fehlende oder falsche Faktoren, fehlende oder falsche (Teil-)Terme usw. Die Fehlerhinweis-Prozeduren untersuchen gerade solche Fehler und geben, falls möglich, entsprechende Informationen. Dabei werden die symbolischen mathematischen Fähigkeiten von Maple genutzt, indem z.B. Ausdrücke miteinander verglichen werden (Differenz=0?) oder in Datenstrukturen nach fehlenden oder falschen Teilen gesucht wird. Eine genauere Beschreibung der Prozeduren, die noch in generische Prozeduren (etwa für Ausdrücke und Gleichungen) und spezifische Prozeduren (für Vektoren, lineare Gleichungssysteme etc.) unterteilt sind, findet man in [1].

Der Screen-Shot in Abbildung 1 zeigt zunächst eine Aufgabenstellung. Abbildung 2 enthält die zugehörige Bearbeitungssektion, in der die Variablennamen und die Namen der korrekten Objekte festgelegt und die verfügbaren textuellen Hilfen als Hyperlinks angegeben sind.

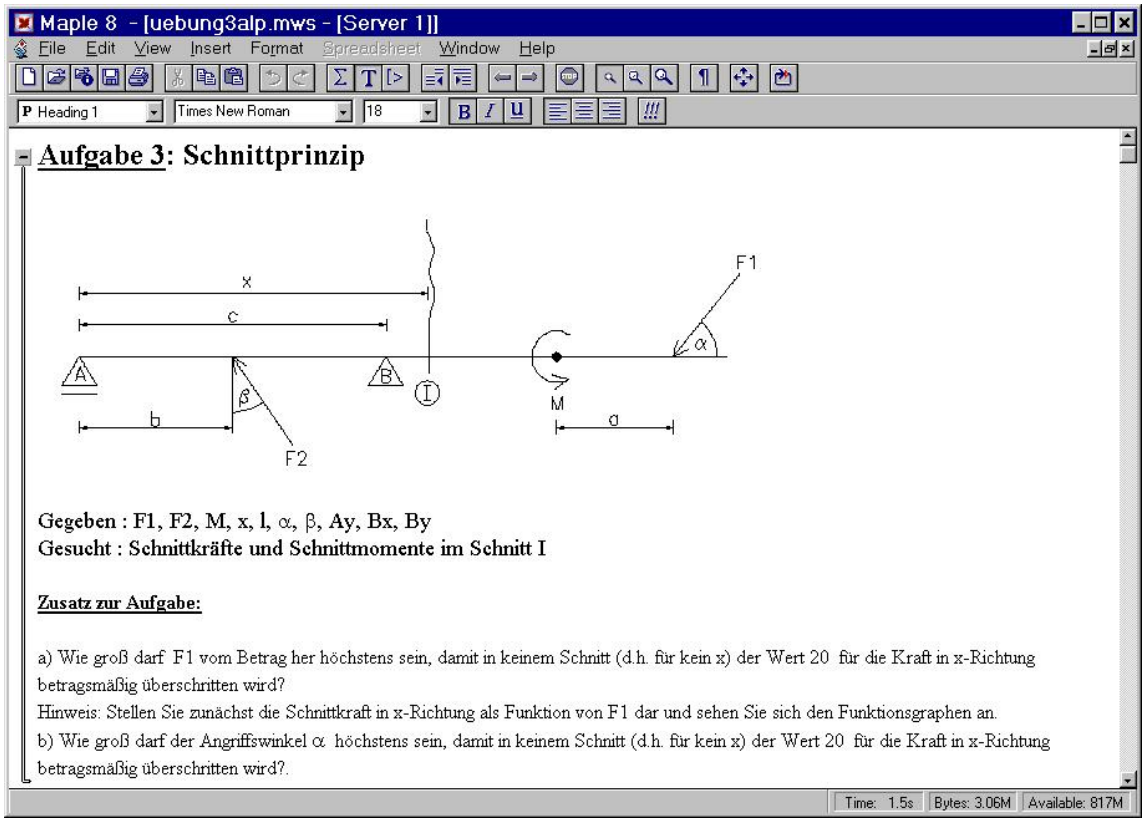

Abbildung 1

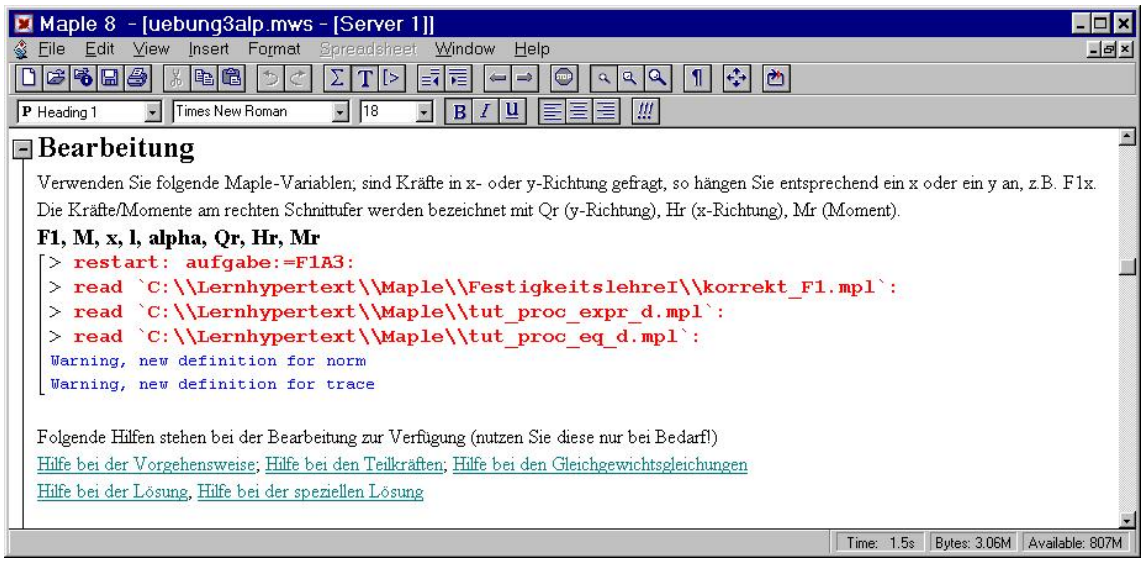

Abbildung 2

Abbildung 3 zeigt schließlich den Teil der Bearbeitungssektion, der die verfügbaren Check-Prozeduren angibt. Zu näheren Erläuterung wird ein Link auf ein Worksheet angeboten, das die Check-Prozeduren im Detail anhand eines Beispiels darlegt.

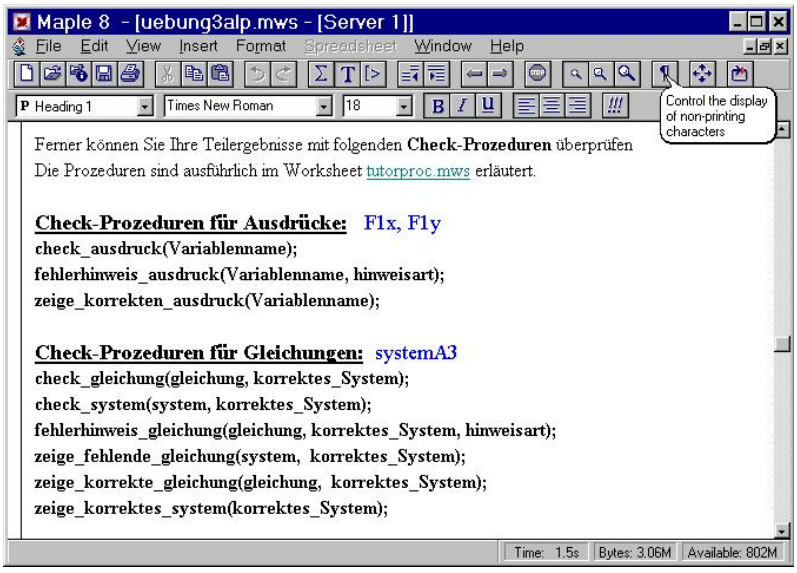

Abbildung 3

Um beispielsweise eine Gleichung zu überprüfen, muss diese zunächst eingegeben werden, etwa in der Form

```
> gleich_x:= 0 = F1x*cos(alpha) +F2x*cos(alpha) + Ax;
```

Dann kann die Gleichung überprüft werden durch Aufruf der entsprechenden Check-Prozedur:

```
> check_gleichung(gleich_x,systemA3);
```

Im Fehlerfall kann dann noch gestuft Fehlerinformation abgerufen werden (Art 1: Nur Fehlertyp; Art 2: Typ und fehlerhafter Teil):

```
> fehlerhinweis_gleichung(gleich_x,systemA3,1);
```

Weitere Beispiele und Angaben zur Erstellung neuer Aufgaben-Worksheets unter Nutzung der bestehenden Prozeduren sind in [1] beschrieben.

Die Aufgabenumgebung auf Worksheet-Ebene ist – trotz der großen Schwierigkeiten, die die Technische Mechanik vielen Studenten bereitet – nicht richtig angenommen worden. Eine Untersuchung mit ausgewählten Studenten hat die zugrundeliegenden Probleme ans Licht gebracht:

- Der gesamte Text zu Beginn der Bearbeitungssektion und die relativ komplexe Syntax beim Aufruf der angebotenen Prozeduren sind stark abschreckend. Die Studenten sind es von Windows-Programmen gewohnt, dass Information per Mausklick abgefragt werden kann.
- Die geringe, rein textuelle Unterstützung in der Modellierungsphase, die in der ersten Version auch noch auf die Festigkeitslehre beschränkt ist, hilft zu wenig beim Einstieg in die Aufgaben.

Daneben ist, wie oben bereits erwähnt, ebenfalls problematisch, dass über die Vorgabe der Variablennamen schon die wesentlichen Variablen benannt werden.

4. Realisierung mit graphischer Oberfläche und Maplets

Seit Maple 8 ist in der Standardversion ein Paket namens „Maplets“ enthalten, mit dem man graphische Oberflächen erzeugen kann, die dem Benutzer die von Windows-Programmen gewohnte Oberfläche (Buttons, Auswahlboxen etc.) anbieten (zur genaueren Beschreibung siehe etwa [6]). Damit lässt sich das erste oben genannte Problem, die umständliche und damit abschreckende Bedienung, weitestgehend beseitigen.

Folgende Anforderungen sind an eine Oberflächenrealisierung mit Maplets zu stellen:

- die Eingabe (Ausdrücke, Gleichungen etc.) muss einfach und gut lesbar sein, um Eingabefehler schnell zu erkennen;
- die Überprüfung von mathematischen Objekten und das Abfragen von Fehlerinformation muss per Mausklick ohne Syntaxkenntnisse möglich sein;
- die Zusatztools (graphische Verfahren, Flächenwerte etc.) sollen einfach und ohne Prozeduraufruf gestartet und bedient werden können;
- die Umgebung sollte einfach um neue Aufgaben erweiterbar sein;
- die bereits vorhandenen Maple-Prozeduren zur Modellüberprüfung sollten weiterhin nutzbar sein.

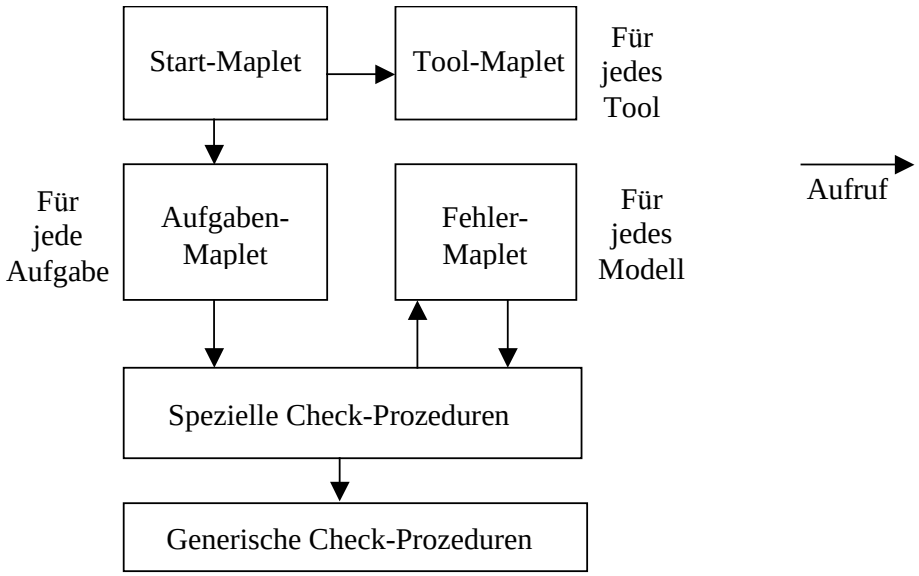

Abbildung 4

Die in Abbildung 4 dargestellte Softwarestruktur trägt diesen Anforderungen teilweise Rechnung. Um die Software übersichtlich zu halten, wurden folgende Arten von Maplets erzeugt:

- Das Start-Maplet bietet die vorhandenen Aufgaben an und ermöglicht ferner den Start eines Tool-Maplets (siehe Abbildung 5).
- Pro Aufgabe gibt es ein Aufgaben-Maplet (siehe Abbildung 6), das folgende Bestandteile enthält:
    - Aufgabenbeschreibung inklusive eingebundener Zeichnungen;

- Beschreibung der zu verwendenden Variablennamen;
- Möglichkeiten der Abfrage von Hilfetext;
- Eingabefelder für mathematische Objekte;
- Buttons zum Aufruf von Check- und Unterstützungsprozeduren sowie zur Anzeige einer PDF-Datei mit der Musterlösung;

- Pro mathematischem Modell (Ausdrücke, Vektoren, lineare Gleichungssysteme etc.) gibt es ein Fehler-Maplet (siehe Abbildung 6), mit dem Fehlerinformation abgerufen werden kann.

Die Check-Prozeduren sind ausgelagert und werden von den Aufgaben-Maplets aus aufgerufen. Sie starten ihrerseits die Fehler-Maplets oder schlicht eine „OK-Box“. Die Prozeduren sind wie schon bei der Worksheet-Version in generische und modellspezifische unterteilt (vgl. dazu [1]).

Abbildung 5: Start-Maplet

Das folgende Maplet zeigt, wie das Fenster in Abbildung 6 erzeugt werden kann (... ... steht für eine Auslassung):

```
maplet_TM1B1A1 := Maplet(

        BoxRow([[Label("Blatt 1, Aufgabe 1",'halign'=right, 'font' = Font("arial", bold, 16))],

        BoxLayout(BoxRow('border'=true,([      "Aufgabenbeschreibung:",
                                               "Ein Balken ist nach Abb. 1 mit Kräften belastet.",
                                               "a) Mit Hilfe des Seileckverfahrens ist graphisch die Resultierende",
                                               "   der gegebenen fünf Kräfte und ihr Abstand a zum Lager A zu
                                                       bestimmen.",
                                               "   Hierbei sollten eine unterschiedliche Reihenfolge der Kräfte
                                                       sowie",
                                               "   verschiedene Lagen von Pol und Anfangspunkt des Seilecks
                                                       gewählt",
                                               "   werden. Wie groß sind die Lagerkräfte A und B ?"
```

```
                                                    " ",
                                                ... ... ...
                                                ],

                                                [Label(Image("Bilder\\imageB1A1.gif"))]
                                                )
                                    )
                        ),

            BoxLayout(BoxRow('border'=true,(["Hinweis für zu benutzende Variablennamen:",
                                                "F1, F2, F3, F4, F5, A, B, Bx, By"]
                                                )
                                    ),
                                    Button("Hilfe", Shutdown())
                        ),

            BoxLayout(BoxRow('border'=false,(["Geben Sie unten die Systemgleichungen ein und benutzen Sie dabei die",
                                                "oben eingegebenen Variablennamen. Mit dem Check-Button können Sie",
                                                "eine eingegebene Gleichung prüfen und ggf. Hilfen anfordern."]
                                                )
                                    )
                        ),
            BoxLayout(BoxRow('border'=true,[  TextField['Loesungssystem'](value="systemA1", visible=false),
                                                TextField['Gleichungsfelder'](value="{A1,A2,A3}", visible=false),
                        BoxLayout(BoxRow('border'=false,(["Summe aller Kräfte in x-Richtung"],
                                                [TextField['A1'](25, 'font' = Font("arial",12))],
                                                [Button("Check Gleichung",

            Evaluate(function="check_gleichung",Argument('A1'),Argument('Loesungssystem')))]


                                                )
                                    )
                                    ),

                                    BoxLayout(BoxRow('border'=false,(["Summe aller Kräfte in y-Richtung"],
                                                [TextField['A2'](25, 'font' = Font("arial",12))],
                                                [Button("Check Gleichung",
                                    Evaluate(function="check_gleichung",Argument('A2'),Argument('Loesungssystem')))]
                                                )
                                    )
                                    ),

                        BoxLayout(BoxRow('border'=false,(["Summe aller Momente              "],
                                                [TextField['A3'](25, 'font' = Font("arial",12))],
                                                [Button("Check Gleichung",
                                    Evaluate(function="check_gleichung",Argument('A3'),Argument('Loesungssystem')))]
                                                )
                                    )
                                    ),
            [Button("Checke System",
                        Evaluate(function="check_system",Argument('Gleichungsfelder'),Argument('Loesungssystem'))),
                        Button("Zeige fehlende Gleichung",

            Evaluate(function="zeige_fehlende_gleichung",Argument('Gleichungsfelder'),Argument('Loesungssystem')))],
                        [Button("Löse System",
                        Evaluate(function="loese_system",Argument('Gleichungsfelder'),Argument('Loesungssystem'))),
                         Button("Zeige korrektes System",
                        Evaluate(function="zeige_korrektes_system",Argument('Gleichungsfelder'),Argument('Loesungssystem')))]
                                                ]
                                    )
                                    ),

            BoxLayout(BoxRow('border'=false,([Button("Lösung",
                        Evaluate(function="zeige_loesung"))],
                                                [Button("Zurück zur Auswahl", Shutdown())]
                                                )
                                    )
                        )

            ],
            hscroll=as_needed,vscroll=as_needed)
):
```

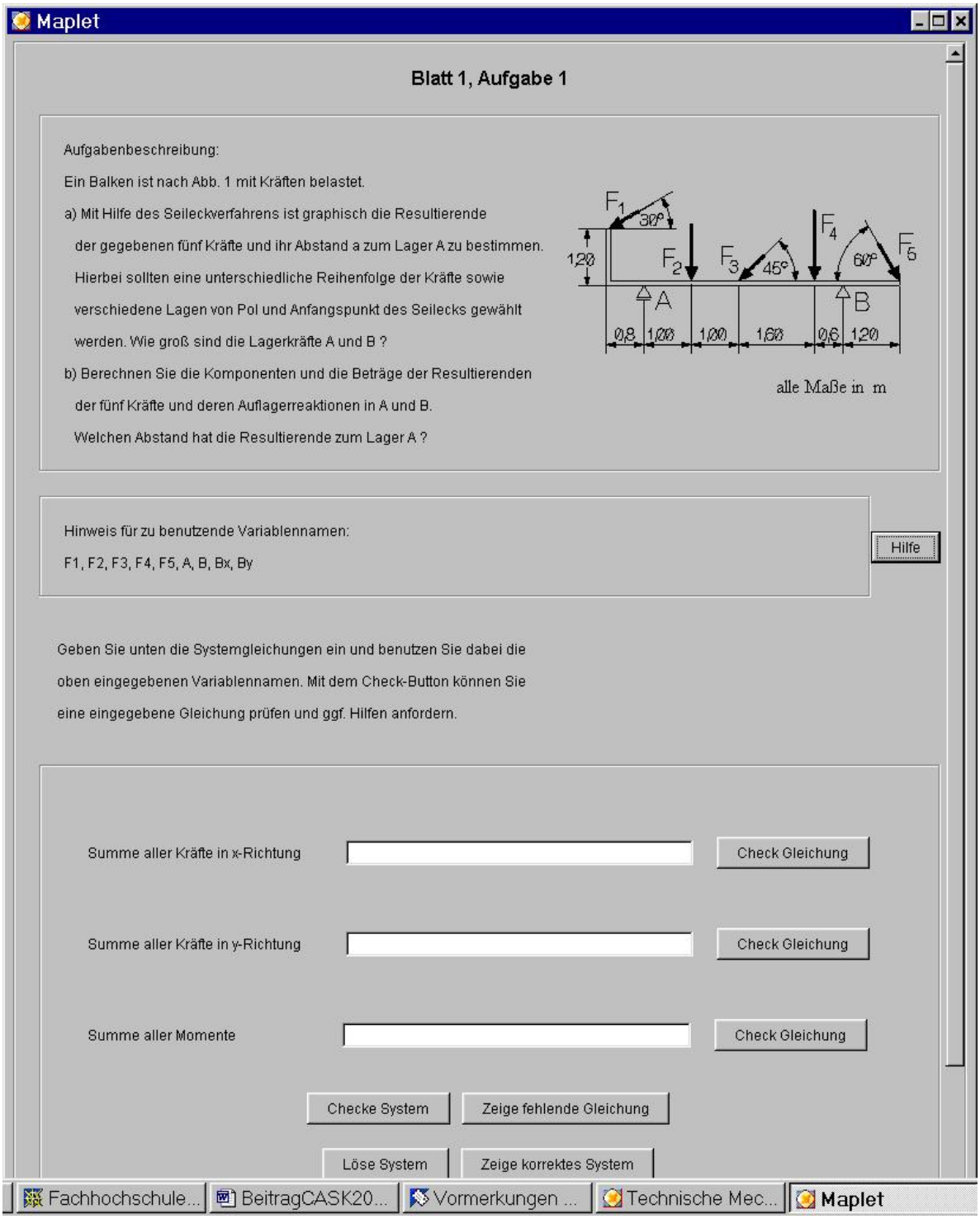

Abbildung 6: Aufgaben-Maplet

Abbildung 7: Fehler-Maplet

Soll eine neue Aufgabe eingefügt werden, die lediglich bereits unterstützte mathematische Modelle verwendet, so sind nur folgende Schritte durchzuführen:

- Erstellung des entsprechenden Aufgaben-Maplets
- Eintrag in die entsprechende Liste im Start-Maplet

Kommen noch nicht berücksichtigte mathematische Modelle vor, so sind entsprechende Check-Prozeduren zu programmieren, was natürlich sehr aufwändig sein kann.

5. Zusammenfassung und Ausblick

Der vorliegende Beitrag beschreibt einen Aufgabentutor für die Technische Mechanik I (inklusive Festigkeitslehre I). Bei der Erstellung einer solchen Umgebung spielen drei Themenkomplexe eine wesentliche Rolle:

- Didaktik: Wie kann man menschliches Tutorverhalten nachbilden? Welche Unterstützung benötigen die Studenten, um sowohl befähigt als auch motiviert zu sein, eigenständig an einer Aufgabe weiterzuarbeiten?
- Mathematik: Wie kann man Fehler in mathematischen Objekten erkennen? Hier kommen gerade die Fähigkeiten von CAS zum Tragen, da man einfach symbolische Ausdrücke vergleichen und symbolische Datenstrukturen „durchsuchen“ kann.
- Softwaretechnik: Wie kann man eine flexible, leicht nutzbare und erweiterbare Aufgabenumgebung schaffen?

In allen drei Bereichen sind noch viele offene Fragen zu klären. So liegt der Aufgabentutor liegt bislang nur in einer Testversion vor, die Akzeptanz bei den Studenten ist also noch nicht sicher. Mit der Umstellung auf Maplets wurde der Bedienkomfort deutlich verbessert, ob aber die angebotene Unterstützung ausreicht, ist noch unklar.

Abschließend noch einige Bemerkungen zur Nutzung des CAS Maple und der Maplets im besonderen für die Erstellung eines Aufgabentutors: Maple liefert eine gute Unterstützung für das Checken mathematischer Objekte. Während in intelligenten Tutorsystemen (ITS), die meist auf KI-Konzepten und KI-Software basieren, das Fehlerfinden sehr aufwändig ist, geht dies mit CAS-Unterstützung

deutlich einfacher. Hier wäre auch der Einsatz von CAS als Komponente in ITS denkbar (vgl. dazu [2]). Was die Kopplung mit anderen Programmen betrifft, so könnte man auch als Zeichenoberfläche ein dynamisches Geometrieprogramm (DGS) einsetzen, wenn eine Kommunikation zwischen CAS und DGS angeboten wird.

Die Maplets sind ohne geeignete Softwareentwicklungsumgebung etwas mühsam in der Handhabung, wie aus dem oben dargestellten Beispielcode unschwer zu erkennen ist. Sie erlauben aber die Erstellung ansprechender User Interfaces. Eine Abhilfe kann hier ein „Meta-Maplet“ schaffen, das zur Erzeugung von Aufgaben-Maplets dient. Der Aufgabenautor bräuchte dann nur die im Maplet erscheinenden Bestandteile anzugeben (Text, Bild, Check-Möglichkeiten), während die ganzen Klammerstrukturen mit Hilfe der „String-Tools“ in Maple automatisch erzeugt würden.

Momentan braucht jeder Nutzer der Aufgabenumgebung eine Maple-Lizenz, es handelt sich also nicht um eine „Stand-Alone-Anwendung“. Maplets bestehen aber im wesentlichen aus Java-Code und mit dem „Maple-Server“ MapleNet® könnte man auch einen entfernten Zugang über einen normalen java-fähigen Browser anbieten.

Literatur:

[1] Alpers, B.: Intelligent Assignment Environments for Mechanical Engineering with Computer Algebra, Int. Journal of Computer Algebra in Math. Education (IJCAME), Vol. 8, No. 4 (2001), 309-328.

[2] Alpers, B.: Using Computeralgebra for Rapid Development of ITS Components in Engineering, in: Gauthier, G., Frasson, Cl., VanLehn, K. (Eds.): Intelligent Tutoring Systems, Proc. ITS 2000, Springer: Lecture Notes in Computer Science Vol. 1839, Heidelberg - New York 2000

[3] Gertner, A.S.: Providing feedback to equation entries in an intelligent tutoring system for Physics, in: Goettl, B.P. et al.: Intelligent Tutoring Systems, Proc. of 4th Int. Conf. ITS '98, Springer: Berlin (=LNCS 1452) 1998, pp. 254-263.

[4] Holland, G.: Intelligent Tutorial Systems, in: Biehler, R. et al.: Didactics of Mathematics as a Scientific Discipline, Kluwer: Dordrecht 1994, pp. 213-223.

[5] Klai, S., Kolokolnikov, Th., Van den Bergh, N.: Using Maple and the Web to Grade Mathematics Tests, in: Kinshuk, J. C. & Okamoto T. (Ed.): Advanced Learning Technology: Design and Development Issues, Los Alamitos, CA: IEEE Computer Society 2000 (IWALT 2000).

[6] Kofler, M., Bitsch, G., Komma, M.: Maple – Einführung, Anwendung, Referenz, 5. vollständig überarbeitete Auflage, Pearson Studium: München 2002.