Using Open-Source Software to Develop Interactive Web-based Assignments for Engineering Students

Michael Griffis
University of Florida Mechanical and Aerospace Dept (email: mwg@ufl.edu).

Abstract

The purpose of this work was to use readily available open-source software to create a software development environment that permits an instructor to quickly develop interactive, individually tailored, web-based assignments for engineering students. It was desired that each student (out of many) would access his or her own individually styled assignment through a Canvas[1] or Sakai[2] type of system.

Keywords

Interactive web-based assignments, open-source software, webserver, engineering, homework

Introduction

Today's engineering student is oftentimes challenged for time and does not expend sufficient effort to do homework. Quality of written or scanned homework is typically not satisfactory (illegible or just plain wrong). Calculation errors are frequent, and students rarely return to correct them. In large measure, students have gravitated away from self-correction of traditional homework based on posted solutions. Basic concepts stressed in class are not completely understood. An additional concern is that students work on homework in groups, and a student X can submit answers that only student Y understands.

To address these concerns, the author saw a need for a method used by an instructor to generate homework assignments that engineering students would do online. Ideally, that method has these features:

1. allows the instructor full flexibility and control to quickly create the desired, focused assignment
2. supports graphics, line drawings, math fonts
3. supports automated scoring and recording
4. is individually tailored to ensure each student understands his or her own assignment
5. gives each student reasonably secure access to his or her assignment
6. encourages the student to think, make a calculation if necessary, and "try" an answer
7. gives instant feedback to the student whether the try was "good" enough

8. allows the student to "finalize" the result (record it), and

9. allows the student to create a "new" version for improved understanding.

In this day and age, one would immediately consider an online solution to this need. Obviously there are some online e-books that utilize Learnsmart^{TM} technology[3], e.g. Mc-Graw Hill, or its equivalent. That technology is an "adaptive learning" concept which does interact with the student. However, the commercial solution limits the instructor in at two ways: (i) it is difficult to know what the student is actually seeing and (ii) it is difficult to get the desired level of focus or control of content on a particular assignment.

A second possibility was to use the quiz options found in Canvas or Sakai. This could be a viable option, if an instructor had only one or two quizzes to do and if the content is multiple choice, without math fonts. Also, individualized quizzes are not options in Canvas or Sakai.

In light of the identified need, the author set out to develop a method using open-source software that was available to use without restriction. The method focused on a software development environment that enables an instructor to easily build an assignment-specific Common Gateway Interface (CGI) program which is accessed by way of a Webserver. Certain parameters of a given assignment would be individually set, so that each student effectively had a dedicated assignment. Remaining parameters are provided by the student and checked for validity by the CGI program. For a given interactive session, the student submits answers using a "Try" button, and the CGI program provides guidance for which answers are correct. The student can then rectify the incorrect answers and use a "Finalize" button to submit for a grade.

Several features are worthy of note. First, the environment supports LaTex math symbols, since providing math symbols is a basic necessity for an engineering assignment. It also facilitates tolerancing on numerical answers based on relative error, e.g. a numerical answer within 0.3% relative error (about 3 significant figures) is "good." The environment can accept pools of multiple choice questions and use a random number generator to create nice subsets of questions to ask a given student. Drop-down boxes are also incorporated when a given parameter needs to be identified from a list of possibles. For actually building a given assignment, the environment currently requires the instructor to supply 4 C-language functions and a few C-language macros.

The remainder of this paper discusses some background to the approach, identifies some open source software that is used, discusses how the software is used to generate a "webgem" assignment, and finally walks through an example of a "webgem" in action.

Background Protocols & Languages

This section briefly discusses some key protocols and languages that were deemed necessary to use:

1. Hypertext Transfer Protocol (HTTP)

2. Common Gateway Interface (CGI)

3. Hypertext Markup Language (HTML)

4. Structured Query Language (SQL)
5. LaTeX Document Markup Language

The HTTP[4] is the backbone of how the World-wide Web works. A client (e.g. student's computer) accesses a webserver (e.g. the instructor's computer) using HTTP. The client issues a `GET` command, for example `GET index.html`. In this case, the webserver responds with the HTML index file in the root directory of the webserver. And the client's web browser kindly displays it. Of course, HTML is the simple language used to markup text in a web browser.

The webserver will have a dedicated directory that allows the client to do more. For example, most webservers identify such a directory with `/cgi-bin`. If a client issues a command such as `GET /cgi-bin/webgem300.cgi`, the webserver will look in the `/cgi-bin` subdirectory for a script or program with the filename `webgem300.cgi`. The webserver will then run that program and collect its HTML output to relay to the client. And the client's web browser kindly displays the results.

A very interesting twist on this interaction is appreciated when considering the CGI[5]. This allows the client to send parameters to the webserver program, thus making the program a CGI program. For example, if the client issues the command:

```
GET /cgi-bin/webgem300.cgi?ID=4610841&VERIFY=Mw5EqLKZvl
```

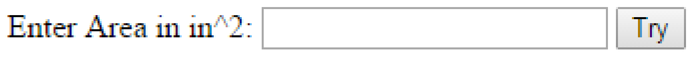
then the webserver runs the CGI program `webgem300.cgi` and provides it with two parameters `ID` with the value `4610841` and `VERIFY` with the value `Mw5EqLKZvl`. To get this to happen, student can actually type the following into the address bar of the web browser:

```
http://whateversite.com/cgi-bin/webgem300.cgi?ID=4610841&VERIFY=Mw5EqLKZvl
```

The CGI program can process these parameters and values to respond to the client. The program can actually add more parameters to the original two parameters and embed the parameters and values into an HTML form that is returned to the client. For example, the following HTML code snippet

```
<form method="POST" action="/cgi-bin/webgem300.cgi">
<input type="hidden" name="ID" value="4610841">
<input type="hidden" name="VERIFY" value="Mw5EqLKZvl">
Enter Area in in^2: <input type="text" name="A" value="">
<input type="submit" name="submit1" value="Try">
</form>
```

replies with an HTML form that includes the original parameters (hidden) and displays a new parameter `A` (text field) and a submit button `TRY`. The CGI program transitions the `GET` command into something else, and the web browser kindly displays the HTML form. The web browser window now only displays `/cgi-bin/webgem300.cgi` in the address bar. What the browser displays might look like this:



Now, after the student enters data into the `A` text field and presses the `Try` button, the client will issue a `POST` command. All parameters and values, whether hidden or not, are resent by the client to the webserver and hence to the CGI program for subsequent processing.

Conceivably, the CGI program could take the parameters `ID` and `VERIFY` and use them to access a local database. (It is the intent of those parameters to establish a session for a particular student, where the session state is stored.) One option for a database language is SQL[6], and that is the one used in this work. Parameters and values can be stored in the database.

The final one, LaTeX Document Markup Language is mainly used to communicate math equations and fonts.

In summary, there are protocols and languages that interrelate here. A client uses the CGI to send parameters & values to the CGI program. That program has its own way of storing these things. After some processing, it responds with an HTML form. And the cycle is repeated as shown below.

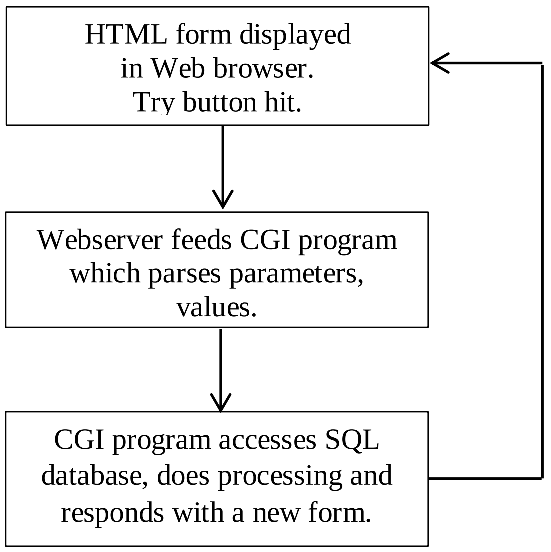

HTML form displayed in Web browser. Try button hit.
Webserver feeds CGI program which parses parameters, values.
CGI program accesses SQL database, does processing and responds with a new form.

Figure 1.

Open-source Software Used

The state of open source software is very good and there are many software packages out there that can be used for free and without restrictions…well, almost with restrictions. The only restriction (typically) is that you cannot place restrictions on derivatives of the source code you develop.

The first open source package to consider is the Apache Hypertext Transfer Protocol (HTTP) Server. This software usually runs on Linux machines but can also run on Windows machines (like the author's). This is the webserver software. It handles the CGI interaction between the CGI program and the client.

The second open source package to consider is the Mingw, which is a Windows port of the GNU compiler collection. The author uses the C-language compiler (also has C++) as it generates a Windows executable without dependencies on external libraries (except for some included with Windows). The CGI programs written by the author are actually C-language console applications.

The third open source package to consider is CGIC, which is an ANSI C-library for Common Gateway Interface (CGI) Programming. This relieved the author of having to worry about parsing the CGI parameters and values. Subroutines were simply called from the author's software.

The next open source package to consider is SQLite, which is an SQL database engine. It stores the database in a single file in the same directory as the CGI program. This package relieved the author of having to worry about accessing the SQL database. Again, subroutines were simply called from the author's software.

The final open source package of note is Mathjax. This is a Javascript mathematics display engine based on LaTeX. All of the worry of conveying math equations and fonts to the student were relieved by using this package. For example, a variable x could be displayed by the LaTeX string "x" and all that was required to achieve this was to add some javascript lines to the HTML form.

Generating a Webgem Assignment

As it stands now, the author has a good foundation of software written in C to handle a large part of the CGI program (call it the C-language program). There are a number of parameters that are pre-established for any given webgem, and the foundation of the software handles interaction of this data. The following table identifies the fundamental parameters that exist in any webgem.

Parameter	Description
state	whether the assignment has been finished
score	what the score is
email	student's email address
id	a version of the ID described earlier
verify	the VERIFY parameter described earlier
scan_bits	a bit-field, where each bit corresponds to whether a corresponding value is correct

In order to generate a particular “webgem” assignment, only a few things need doing:

1. Some mundane things need to be set up, like the name of the C-language program, the filename of the SQL database, the date the assignment is due.
2. The parameters that are assignment-specific need to be established. These are the variable names that are common throughout the C language program, the HTML form, and in the SQL database. From the standpoint of the student, some of these are “givens” and some of these are to be determined and submitted by the student (the “determines”).
3. Three macros need to be written. The first reads all of the parameters from the database into the C-language program. The second writes the givens from the C-language variables to the database. The last updates the database with current C-language variable values that are determines.
4. Four functions need to be written.
    a. `init_quiz`. This function needs to create a coherent set of data to be the givens for a student.
    b. `get_cgi`. This calls the CGI functions to read what the student supplied as determines.
    c. `scan_quiz`. This checks whether (and which) determines are correct. Determines which are floating point values can be checked for error within some relative tolerance (viz. relative error).
    d. `put_cgi`. This generates the HTML form and identifies (with decoration) which determines are correct and which are incorrect.

Webgem Example

As an example, consider we are writing a webgem to quiz factors of safety for ductile or brittle materials together with stress concentration factors. The following figure is the stress concentration factor for a flat plate with a hole, where the plate is under tension (static load).

Figure A–15–1

Bar in tension or simple compression with a transverse hole. $\sigma_0 = F/A$, where $A = (w - d)t$ and t is the thickness.

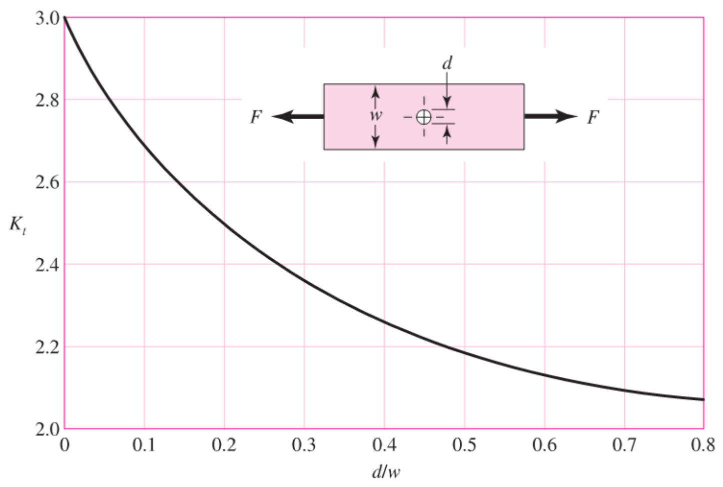


Figure 2

In this case, the stress concentration factor K_t is a function of the dimensionless ratio d/w, where d is the diameter of the hole and where w is the width of the plate. The nominal stress is $\sigma_0 = F/A$, where F is the force, where worst-case area is $A = (w - d)t$, and where t is the thickness of the plate.

The factor of safety depends on whether the material is ductile or brittle. If it is ductile, then the factor of safety is (maximum shear stress theory)

$$n_{MSS} = \frac{S_y}{\sigma_0},$$

where S_y is the yield strength. (In other words, K_t is ignored if ductile, since this just strain hardens a ductile metal.) If it is brittle, then the factor of safety is (brittle Columb-Mohr)

$$n_{BCM} = \frac{S_{ut}}{K_t\ \sigma_0},$$

where S_{ut} is the ultimate (tensile) strength.

A metal is considered ductile if it has a true strain at fracture greater than or equal to 5%. In other words, ductile if $\varepsilon_f \geq 0.05$. Conversely, a metal is considered brittle if it has a true strain at fracture less than 5%. In other words, brittle if $\varepsilon_f < 0.05$.

With those preliminaries out of the way, it is desirable to create a webgem to quiz the student on these concepts. First identify the givens, shown in the following table.

Givens	Description
F	force (lbf)
W	width (in)
D	diameter (in)
T	thickness (in)
SYA	yield strength of material A (kpsi)
SUA	ultimate strength of material A (kpsi)
EFA	true strain at fracture of material A (-)
SYB	yield strength of material B (kpsi)
SUB	ultimate strength of material B (kpsi)
EFB	true strain at fracture of material B (-)

The objective here was to give the student two materials (one being ductile and the other brittle). The values established by the `init_quiz` function created a coherent set of givens.

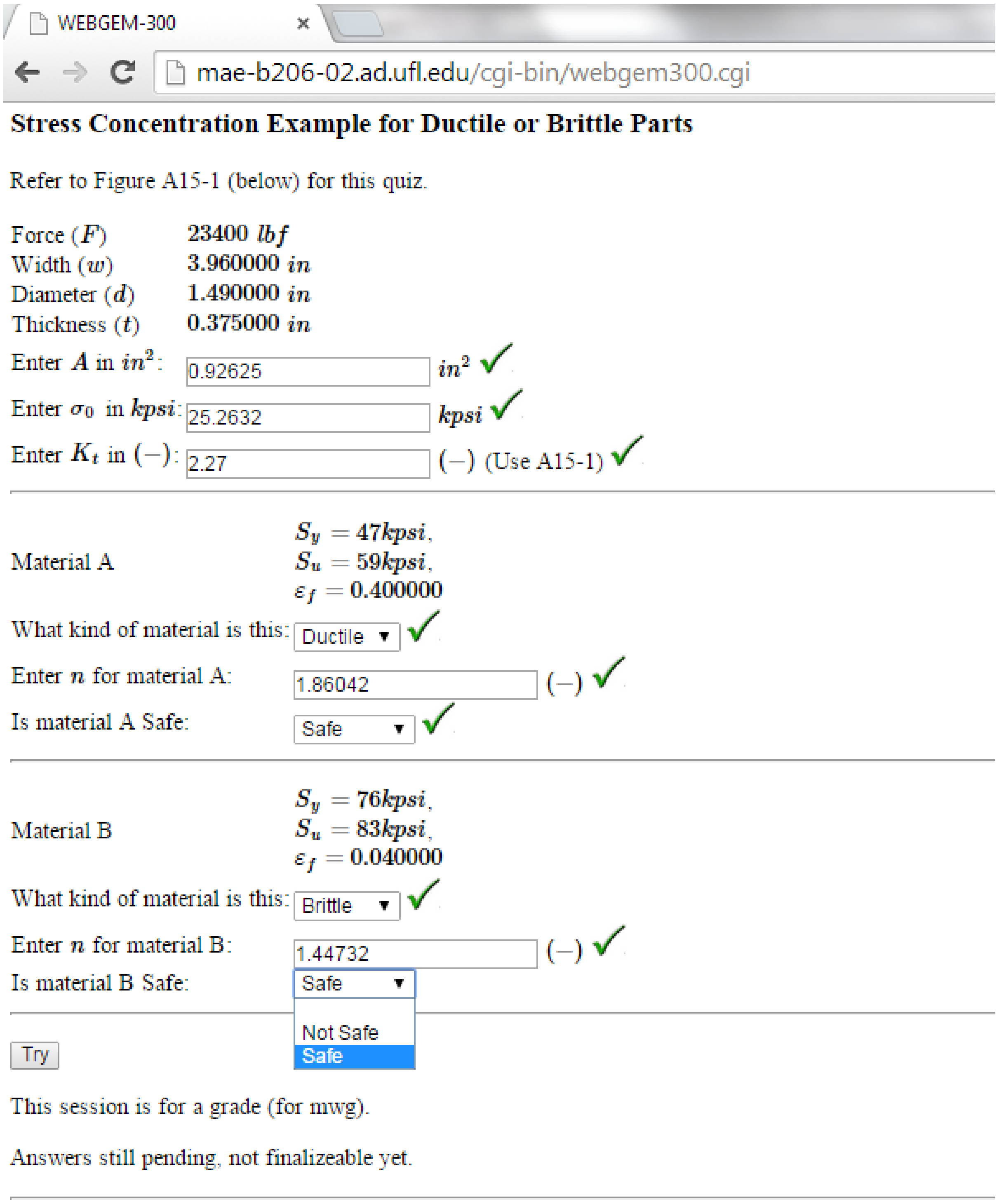
Second, identify the determines, shown in the following table.

Determines	Description
A	area
SIG0	σ_0
OKA	whether material A is safe
OKB	whether material B is safe
MATA	whether material A is ductile or brittle
MATB	whether material B is ductile or brittle

The preceding variable names are the ones actually used in the C-language file, in the SQL database, and in the HTML form fields.

The `get_cgi` function is responsible for retrieving the above determines from the CGI. The `scan_quiz` function is responsible for accessing which determines are correct. Finally, the `put_cgi` function is responsible for formatting the HTML form. The following sequence of figures show how this worked for this example.

With the setup as described, the webgem appears to the student as shown in the following figure. Here the student has progressed nearly to the end. The checkmarks show successful progress.



WEBGEM-300

mae-b206-02.ad.ufl.edu/cgi-bin/webgem300.cgi

Stress Concentration Example for Ductile or Brittle Parts

Refer to Figure A15-1 (below) for this quiz.

Force (F)	23400 lbf
Width (w)	3.960000 in
Diameter (d)	1.490000 in
Thickness (t)	0.375000 in
Enter A in in^2:	0.92625 in^2 ✓
Enter σ_0 in $kpsi$:	25.2632 $kpsi$ ✓
Enter K_t in $(-)$:	2.27 $(-)$ (Use A15-1) ✓

Material A $S_y = 47 kpsi$, $S_u = 59 kpsi$, $\varepsilon_f = 0.400000$

What kind of material is this:	Ductile ✓
Enter n for material A:	1.86042 $(-)$ ✓
Is material A Safe:	Safe ✓

Material B $S_y = 76 kpsi$, $S_u = 83 kpsi$, $\varepsilon_f = 0.040000$

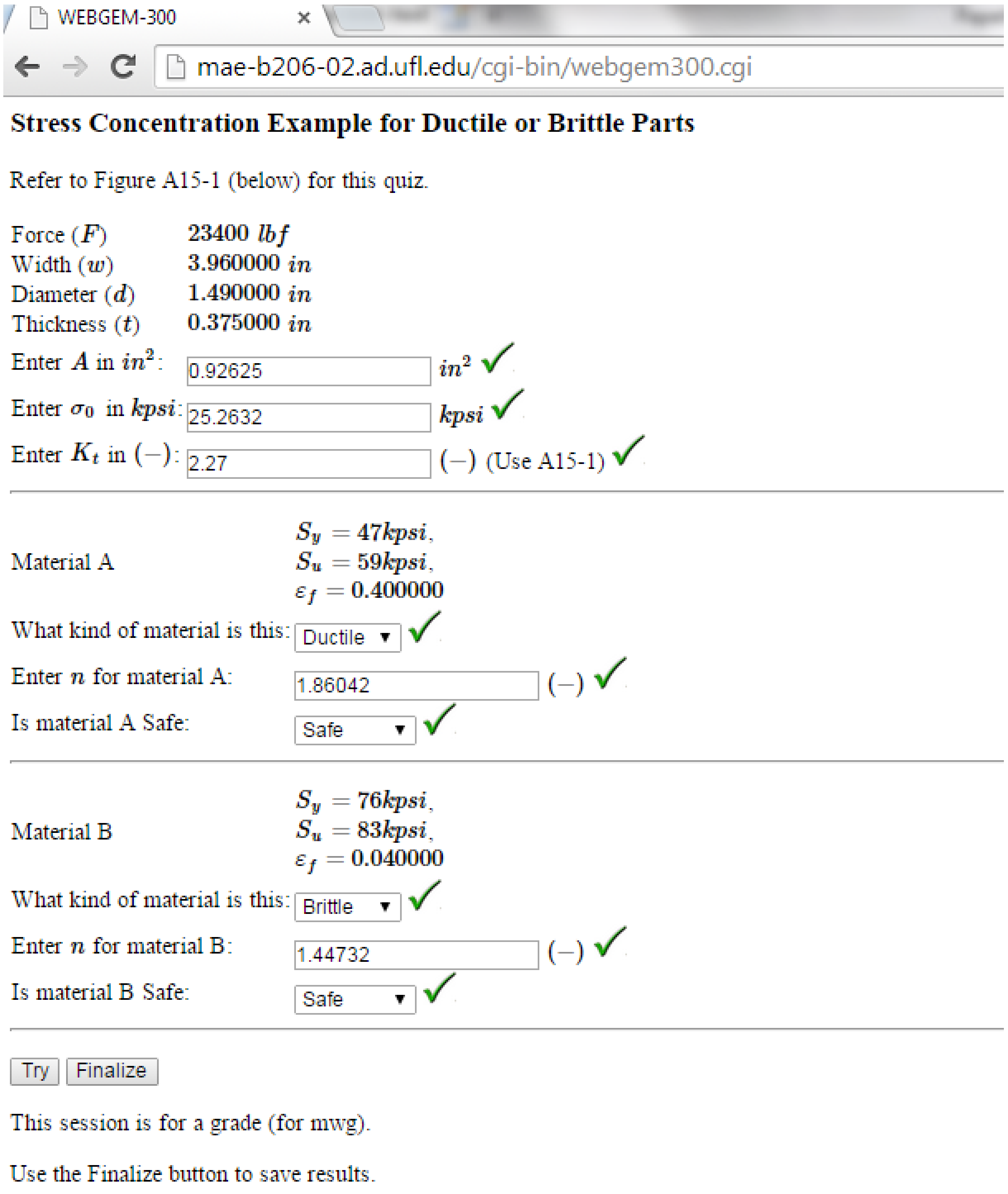
What kind of material is this:	Brittle ✓
Enter n for material B:	1.44732 $(-)$ ✓
Is material B Safe:	Safe

This session is for a grade (for mwg).

Answers still pending, not finalizeable yet.

Figure 3.

Once the student declares answers for all determines, the webgem transitions such that it can be finalized:



Stress Concentration Example for Ductile or Brittle Parts

Refer to Figure A15-1 (below) for this quiz.

Force (F)	23400 lbf
Width (w)	3.960000 in
Diameter (d)	1.490000 in
Thickness (t)	0.375000 in
Enter A in in^2:	0.92625 in^2 ✓
Enter σ_0 in $kpsi$:	25.2632 $kpsi$ ✓
Enter K_t in $(-)$:	2.27 $(-)$ (Use A15-1) ✓

Material A $S_y = 47 kpsi$, $S_u = 59 kpsi$, $\varepsilon_f = 0.400000$

What kind of material is this:	Ductile ✓
Enter n for material A:	1.86042 $(-)$ ✓
Is material A Safe:	Safe ✓

Material B $S_y = 76 kpsi$, $S_u = 83 kpsi$, $\varepsilon_f = 0.040000$

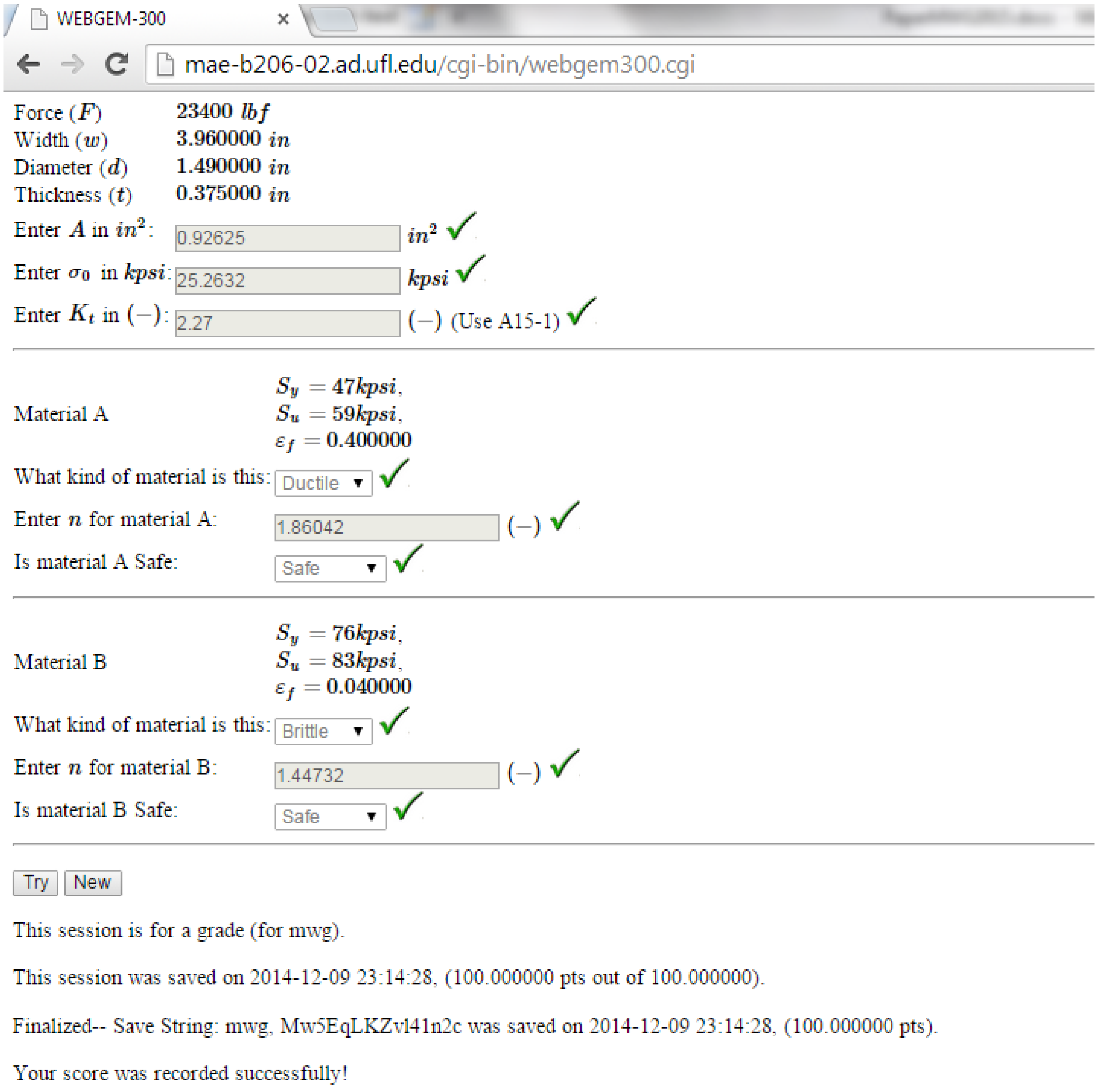
What kind of material is this:	Brittle ✓
Enter n for material B:	1.44732 $(-)$ ✓
Is material B Safe:	Safe ✓

This session is for a grade (for mwg).

Use the Finalize button to save results.

Figure 4.

After it is finalized, the system shows the student the results and allows the student to create a new quiz.



Force (F)	23400 lbf
Width (w)	3.960000 in
Diameter (d)	1.490000 in
Thickness (t)	0.375000 in
Enter A in in^2:	0.92625 in^2 ✓
Enter σ_0 in $kpsi$:	25.2632 $kpsi$ ✓
Enter K_t in $(-)$:	2.27 $(-)$ (Use A15-1) ✓

Material A	$S_y = 47kpsi$, $S_u = 59kpsi$, $\varepsilon_f = 0.400000$
What kind of material is this:	Ductile ✓
Enter n for material A:	1.86042 $(-)$ ✓
Is material A Safe:	Safe ✓

Material B	$S_y = 76kpsi$, $S_u = 83kpsi$, $\varepsilon_f = 0.040000$
What kind of material is this:	Brittle ✓
Enter n for material B:	1.44732 $(-)$ ✓
Is material B Safe:	Safe ✓

This session is for a grade (for mwg).

This session was saved on 2014-12-09 23:14:28, (100.000000 pts out of 100.000000).

Finalized-- Save String: mwg, Mw5EqLKZvl41n2c was saved on 2014-12-09 23:14:28, (100.000000 pts).

Your score was recorded successfully!

Figure 5.

Conclusions

Obviously, there are many commercial interests attempting to teach by using interactive online guidance. It was noted in the **Introduction** there are commercial packages like Mc-Graw Hill's Learnsmart^{TM}. Two typical key limitations (from the perspective of the instructor) were identified:

i. it is difficult to know what the student is actually seeing and

ii. it is difficult to get the desired level of focus or control of content on a particular assignment.

With the help of a reviewer to this paper, the author was made aware of ALEKS^{R}. While this does look promising, the author had a terrible initial experience. The author established a "guest account" and attempted to appreciate the *Math Prep for College Physics* course as an example, viewing as a potential student. That particular online software gave the author a sequence of 20 math questions. The author answered the first two and received no feedback whether the answers were correct. Perhaps being impatient, the author then answered "I don't know" for the remaining 18 questions. The software responded that the author had a "Course Mastery" of 56 out of 190 topics! The software never did say whether the two answered questions were correct. (One might guess yes, since mastery was scored as 56.)

From the author's perspective, the prevailing conclusion here is the instructor needs more control (and knowledge) over what is being shown to the student. (Hence, the author, as an instructor, developed the software being presented.) This is true particularly if the online program is a core part of the course. As it stands currently, the author views his software as complementary. It is no where near a stand alone course.

The software as described herein has seemed to improve the students' performance with respect to being able to make engineering calculations and understanding them. The author is now attempting to quantify the improvement by looking at semester-by-semester exam grades (before the online webgems with respect to now).

While the software described herein has many positive aspects, it must currently be supplemented with in-class guidance. For example, when there is a common mistake on a particular part of a webgem, the author will discuss the challenge in class in order to provide needed guidance. The author feels that further development of the software could incorporate some additional feedback with respect to what the student is doing wrong (e.g. some audio guides). That way, the student is getting more feedback on this "Try" in addition to the fact the student has provided the wrong answer.

In conclusion, thus far, the method described herein works extremely well and achieves the goals of

1. allowing the instructor full flexibility and control to quickly create the desired, focused assignment

2. supporting some graphics and math fonts

3. supporting automated scoring and recording
4. being individually tailored to ensure each student understands his or her own assignment
5. giving each student reasonably secure access to his or her assignment
6. encouraging the student to think, make a calculation if necessary, and "try" an answer
7. giving instant feedback to the student whether the try was "good" enough
8. allowing the student to "finalize" the result (record it), and
9. allowing the student to create a "new" version for improved understanding.

The next area of improvement regards incorporating realtime graphics and line drawing interactively. That way the author can support interactive free-body diagrams.

References

1 http://www.instructure.com/
2 https://sakaiproject.org/
3 http://learnsmartadvantage.com/
4 Wong, Clinton, HTTP Pocket Reference, O'Reilly and Associates, Sebastopol, CA, 2000.
5 Stein, Lincoln, Official Guide to Programming With Cgi.Pm, The Standard for Building Web Scripts, John Wiley and Sons Ltd, New York, 1998
6 Reese, George, MySQL Pocket Reference, O'Reilly and Associates, Sebastopol, CA, 2007.

Michael Griffis

Dr. Griffis has 29+ years experience in the robotics and machine design industry and is a recognized expert in motion and force control. Professional accomplishments include a novel kinestatic control strategy for force control, design and development of a sub-micron machine tool for the semiconductor industry, design and development of a fully automatic laser lace cutter, design and development of a fully automatic battery welder, and design of a control system for a robotic refueling system. Recently as a Lecturer, Dr. Griffis has been teaching Mechanical Design at the University of Florida in the Mechanical and Aerospace Dept.