SOFTWARE ENGINEERING METHODS (CS 236321)

SPRING 1998 COURSE OUTLINE
LECTURES: Tuesdays 8:30-10:20 Fishbach 507 (Multimedia Room)
RECITATIONS: Wednesdays 10:30-12:20 Fishbach 507 (Multimedia Room)

PROF. DANIEL BERRY
E-mail address for questions: dberry@cs.technion.ac.il
Home page: http://www.cs.technion.ac.il/˜dberry
Room: Fishbach 479
Office hours: Wednesdays 12:30-14:20
Telephone number: NONE (Does not use telephone)
E-mail address for first message and for handing in test cases only: berrycourses@cs.technion.ac.il

Assistant: Ido Nachtom
E-mail address: ido@cs.technion.ac.il
Room: Fishbach 404/2 (call 4528) to gain entry)
Office hours: Sundays at 10:00-11:50
Telephone number: 829-4528

Date	Tue.	Topic	Wed.	Topic	Goodies
March	10	Administrative & Some Software Myths			Goodie 1 assigned
			11	Information Hiding	
	17	Information Hiding			
			18	Faking It & Documentation	
	24	Acting Out ADTs & Windows Decomposition			
			25	Testing	PreGoodie 1 due 14:30
	31	OO Development Cost			
April			1	Walkthrough PreGoodie	Test plan due 14:30
	7	Conference	8	Conference	NO CLASS
	14	Pesach	15	Pesach	NO CLASS
	21	Conference			NO CLASS
			22	SW Quality	Ido Nachtom
	28	Walkthrough Test Plan			
			29	Yom HaZicharon	NO CLASS
May	5	In-class Inspection			
			6	Inspection	Goodie 1 due 14:30 Goodie 2 assigned
	12	Inspection			
			13	SW Project Management & SW Cost Estimation	
	19	Student’s day			NO CLASS
			20	SW Cost Estimation	
	26	SW Legal Issues			
			27	Myths	ReGoodie 1 due 14:30
June	2	Myths			
			3	Myths	

9	Requirements Iceberg	10	Ignorance Hiding	
16	Ignorance Hiding	17	OO Design & Patterns	Goodie 2 due 14:30
23	OO Design & Patterns	24	Decoupling Change from Design	

The recitation (tirgul) sections will be used mainly as additional lectures. I have too much material for just the lectures.

VERY IMPORTANT FOR ENABLING COMMUNICATION

No grades will be given without this information.

From preferably the account at which you prefer to receive e-mail, please by 18 March 1998, send to **berrycourse@cs.technion.ac.il** an e-mail message in which you give as a one-line message,

1. your full nine-digit mispar zehut,
2. your private (first) name, in the Latin letter spelling that you prefer,
3. your family (last) name, in the Latin letter spelling that you prefer,
4. a secret word, and
5. your preferred e-mail address (even though I can see it above in the “From” line),

in that order. (If it is not in that form, it will be returned to you unlogged.)

Please use the same preferred Latin spelling that you give to me in this message in all the homeworks that you give me during the semester. I use this name as the key to my data base and not your student number.

The secret word is a word known only to you and me that you will *not* forget by the end of the semester. I will post the grades paired with these secret words rather than with student identification numbers (which are no longer private, given that anyone can finger t2 for the real-life name associated with any student id-login name). In particular, I will not be saving sent messages in order that I can send all previously sent messages to those who are late in sending me their information. In addition, I will be sending your final grades by e-mail to the account you have given me. For example, for me the one-line message would be:

```
123456789 Daniel Berry qapla dberry@cs.technion.ac.il
```

I will be using this information to build my grading database, and to be able to send you announcements and useful materials, e.g., sample inputs and goodie solutions. If you fail to do this, you will not receive any grades, announcements and materials, and if this messes you up, that’s your problem! In order to help you avoid these problems, I will reply “I got it.” when I receive what you send correctly. If you do not receive this reply within two business days after sending it to me, please try again or come to see me.

Office Hours

My office hours are Wednesdays at 12:30-14:20 in 479 Fishbach, right after the lecture hours. You may send e-mail to *dberry@cs.technion.ac.il* for an appointment if this is not convenient for you. The office hours of the assistant, Ido Nachtom are Sundays at 10:00-11:50 in Fishbach 404/2 (call extensin 4528 to gain entry to 404). You may also send questions to me by e-mail. I will answer usually within a single business day.

I am also available at other times, particularly if you see my door open or if you see me walking around campus. Just ask if I am busy. Usually I will not be, but please respect my answer if it is “No”. However, please do *not* even try to see me from ½ hour before class until class, even if my door is open. I often need that time to think about what I am going to say. Many times, when people come to me during that perion, I find myself getting uptight and snapping at them. Please wait until after class.

About the goodies

1. The goodies are lengthy programming assignments designed to get you to apply the techniques described in the lectures, mainly abstract data typing, information hiding, and object orientation. In fact, you will lose many points if you do not apply these techniques to the problem even if you do well otherwise. Moreover, if you do well on Goodie #1, you will find the subsequent goodies easier. I understand that it is difficult to know ahead of time how I will be grading; it is always a matter of my interpretation of what I require versus yours. Therefore, you will get a chance to redo Goodie #1 using the feedback you get from me in order to improve your grade up to 90% of what the goodie counts *and* to have a better version to use for the subsequent goodies. Indeed, I hope many of you will use this as an opportunity to redo your goodie to have a better version for the subsquent goodies, *even* if you get a satisfactory grade on Goodie #1.

2. Goodie #2 is a revision of Goodie #1. The customer (me) of Goodie #1 will discover corrections and enhancements as a result of using your products and will require them to be incorporated into Goodie #2. These enhancements will be done to the code produced for Goodie #1 or its redo. As you are doing Goodie #1, you are encouraged to suggest improvements on paper to the client for incorporation into the requirements of the subsequent goodie, but you must still implement the requirements for the goodie as given. There may be two different Goodies #2, each to be done by half the teams.

3. The goodies will be done in teams of two. Each member of a team will receive the same grade. That is, each member of the team is reponsible for the whole team. If you find, after all, that you cannot work with your teammate for any reason, especially if you feel that your teammate is not doing his or her share, then come to me and apply for a divorce. While it is better that both come to me and apply for divorce, for good reasons, I will grant a divorce to one in the absence of the other. In case of a divorce, each team member gets a copy of the work accumulated so far and then works independently. The problem is small enough to be done individually. Actually, it is about the same amount of work either way; the time you gain from having two working people is lost in the increased communication required to keep each other up to date. The purpose of the team is to get experience in team dynamics. Because of this divorce mechanism, I will not accept as an excuse for lateness the claim that your teammate did not work. If one teammate has a certified miluim or medical absence, the team can turn in its work late as described below.

4. Pregoodie #1 counts as 5%, The test plan for Goodie #1 counts as 5%, Goodie #1 counts as 40%, and Goodie #2 counts as 50%. Note how the points go up as you get more experienced with the program.

5. Goodies must be turned in on time; the first day late loses 10% of the credit, the second day late loses an additional 7.5%, and the third and any subsequent day each loses an additional 5% of the credit. This means, as in real life, if you are going to be late, you will have to weigh the cost of delaying one more day vs. that of taking your chances with what you have now. For purposes of measuring time, the day ends at 14:30 in the afternoon, when I usually disappear. In fact, on days goodies are due, I *will* disappear at 14:31

and a goodie that is even 1.0000001 minutes late will count as one day late. If you don't like this, you do not have to take the course, but that's the way it is in the real world. I do not accept the excuse that the computer was down or the printer queue was full on the last day, because that is a well-known phenomenon that can be defended against by budgeting enough time to finish before the last day. The shabbat or a yom-tov plus its preparation day do not count in this scheme. Certified miluim and medical absences allow you to be as many days late, without penalty, as the certificate says you were on miluim or sick; the day after that, penalties kick in as if you were 1 day late.

6. You may use any account you may legally have access to. You may use any of the compiled algorithmic languages available on the system you are using, e.g., C, Fortran, Ada, C++, or even Pascal. If the language you want to use is not mentioned in the previous line, then see me. I want to verify that it is algorithmic. Therefore, I can tell you now that LISP is not acceptable. It is your responsibility to learn how to use the editing and job submission facilities on the machine you use. You are expected to pick up the necessary knowledge to use whatever system you pick on your own.

7. When you turn in a program you will turn in to me

   a. in human-readable form:

      1. The source program listing, with a table of contents and page numbers or with tabs labelling sections attached to the edge of the pages.

      2. **Two** copies of a module diagram, showing in the style of my drawings for the "Information Hiding" lecture, the modules of your program and their use of other modules. These are not inheritance diagrams; you may show me inheritance diagrams in *addition* to these required diagrams, but not in place of them.

      3. Module descriptions, showing the name of the module and the header of each identifier, usually those of procedures, invokable from outside the module. In C++-terminology, this will be a collection of a variation of `.h` files, showing *only* the exported identifiers and not anything that should be hidden, such as data structure definitions. In order not to confuse these with the standard `.h` files, which must show private and protected data in C++, these files should be called `.mod` files. The module descriptions must be *separate* from the source program listing, which consists of its own `.h` and `.c` files, even though all the information required for the module descriptions is in the source listing. I do not like having to hunt for individual `.h` files buried among other files; I want the `.mod` files all together. Failure to provide the module descriptions as separate from the source listing will cost points.

      4. Either each exported procedure is named so that its function is obvious or you will have comments explaining its function.

   For Goodie #2, the parts of the *module description* that are modified from the previous goodie must be clearly marked, by highlighting, change bars, underlining, boldfacing, etc. Failure to do so will cost points.

   b. in electronic form, either electronic mail, **sent to berrycourses@cs.technion.ac.il**, or a diskette, as described in detail below, containing the input and output from test runs (as many as *you* deem appropriate given the nature of the problem; indeed part of the correctness grade depends on how well you selected your test data). Failure to do as described will cost points.

      1. Please prepare a single directory for all and only the test cases, their documentation,

which must explain what each test case is testing, and test case drivers.

2. If you are sending me the tests by e-mail, then send me as a single message the result of a uuencoding of the result of tar'ing the directory containing all the test files.

   If you are giving me a diskette, then give me a diskette with all and only the test files at the root directory of the diskette.

   Do not include the source program either in the e-mail or in the diskette.

3. For each test case *xxx*, (where *xxx* is some descriptive name), have an *xxx*`.in` that is the input and *xxx*`.out` that is the resulting output.

4. The documentation about test case *xxx* describes in words what is tested and is in a file called *xxx*`.doc`.

5. The test case driver for test case *xxx* is in a file called *xxx*`.run`. This is so I can see the full command lines that runs each case.

6. If the goodie involves doing something with PostScript, hand in with the hard copy module diagram, decomposition, and source code, the result of printing all non-error *xxx*`.out` files on a PostScript printer.

7. If you hand in a diskette, write your name(s) on it!

8. If several test cases share the same *xxx*`.in` file, document that in the corresponding *xxx*`.doc` files and use the same *xxx*`.in` in each of the different *xxx*`.run` files. That is, it's possible that there are fewer *xxx*`.in` files than *xxx*`.out` files, but *xxx*`.out` files, *xxx*`.doc` files, and *xxx*`.run` have to be one-for-one-for-one.

The reasons for this electronic submission are to save paper and to make it easier to grade the goodies. By following the requirements exactly, I will be able to follow a simple uniform procedure to grade the test cases. Failure to follow these requirements exactly will cost points.

It is a real shame when you lose points, totally unnecessarily, on a goodie because of failure to follow these simple, explicit directions, but it is *your* problem.

8. DO *NOT* HAND IN

   a. A FLOWCHART,

   b. YOUR PROGRAM FLOODED WITH COMMENTS, OR

   c. A NATURAL LANGUAGE DESCRIPTION OF INNER WORKINGS OF PROGRAM.

   If you hand in one of these, you will *lose* points even if you have handed in all of the required documents.

9. On the day you are to hand in a programming assignment, there *may* be a short quiz in class. The quiz is designed to be trivial if you did your own work on the problem and impossible if someone else did your work. Your grade on the goodie will be weighted by your performance on this quiz. If you are absent from class on that day, get in touch with me.

Course Materials

Course materials are available by anonymous ftp from *ftp.cs.technion.ac.il*. Login with user "anonymous" and send your *login@site* as the password. The material for this class is in the directory:

/pub/courses/software.engr

Under the directory *lectures.ps*, you will find there, shortly before any lecture, PostScript files (all file names ending with *.ps*) for the lecture slides. The file names should be obvious, and in any case, there is a *README* file. At the suggestion of a student in the Fall of 1994, I am making the notes available this way. You can ftp them and print them at your own expense. Evidently, this is cheaper than paying Michlol for a hard copy. Under the subdirectory *lectures.pdf*, you will also find the same material in PDF form (all file names ending with *.pdf*), readable by the Adobe Acrobat Reader, which is available for free by anonymous ftp from *ftp.adobe.com* or from Abobe's web site, *www.adobe.com*. If you get the PDF form material, I highly recommend that you use it *only* for viewing with an Acrobat reader and *NOT* for printing. The pages of the PDF form material have a blue background and this will use a lot of ink, even on a black & white printer, which will approximate the blue with a shade of gray. Moreover, it prints one pages per sheet. If you want to print out a hard copy, please also take the corresponding PostScript file. It prints strictly black text on white and 4 pages per sheet. Also a single hard copy of this 4-page-per-sheet output is available for copying in the library.

There are *adm* and *goodie* subdirectories, which contain PostScript, PDF, and plain ASCII files for administration and goodie-related documents, such as this outline, which is under *adm* as *outline.ps*.

To get the files, after connecting, *cd* to the directory mentioned above. If you getting PDF files, say "*binary*". PostScript files must be gotten in ASCII mode unless you happen also to be at a UNIX machine. To get any particular file *f* say "*get f*". Don't forget to disconnect by saying "*quit*". If *f* is of the form *g.ps*, then the result is a PostScript file. This can be printed by use of "*lpr g.ps*" or it can be previewed on your screen with ghostscript, which is described below. If *f* is of the form *g.pdf*, then the result is a PDF file, viewable and printable with an Acrobat Reader.

GhostScript is called *gs* on most UNIX systems and it assumes that you are running an X-windows. There is a PC version of Ghostscript that comes in two different flavors, one for DOS and one for WINDOWS. The advantage of the WINDOWS version is that you can type commands to it in one window, a DOS shell window, and you will see the results in another, ghostscript graphics window. In the DOS version, both the commands and the results are written to the same, and only screen. You can find the PC version of Ghostscript in `x:\software\ghost` in the file server for the PC farm. Since the program is public domain, you are welcome to copy the stuff there in order to install it on any other PC you have access to.

These course materials are also available through the course Web page, whose URL is:

http://www.cs.technion.ac.il/~cs236321

Note though, that this page is under construction this semester, so it may not be as up to snuff as the ftp site, which has been around for some time now.

Additional Reading

G.M. Weinberg, *The Psychology of Computer Programming*, van Nostrand Reinhold: New York, NY, 1971

D.L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", *Communications of the ACM*, 15: 2, pp. 1053-1058, December, 1972

F.P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering*, Second Edition, Addison-Wesley: Reading, MA, 1975

B.W. Boehm, *Software Engineering Economics*, Prentice-Hall: Englewood Cliffs, NJ, 1981

F.P. Brooks, Jr., “No Silver Bullet”, *IEEE Computer*, 20: 4, pp. 10-19, April, 1987

S.R. Schach, *Software Engineering*, Second Edition, Aksen Associates & Irwin: Boston, MA, 1992

D.L. Parnas and P.C. Clements, “A Rational Design Process: How and Why to Fake It”, *IEEE Transactions on Software Engineering*, SE-12: 2, pp. 196-257, February, 1986

E. Gamma, R. Helm, R. Johnson, and J. Vlissides *Design Patterns* Addison-Wesley: Reading, MA, 1995