

# Maple and software engineering education

(an activity report on the use of Maple  
in two courses at Western Michigan University)

Prof. Thomas F. Piatkowski (thomas.piatkowski@wmich.edu)

Prof. J. Donald Nelson (nelson@cs.wmich.edu)

Department of Computer Science  
Western Michigan University  
Kalamazoo MI 49008

**keywords :** automata theory, collaborative education, computer science, cs580, cs580Lib, cs661, education, formal languages, Maple, programming in Maple, research, software engineering, testing, validation, verification, Western Michigan University.

**intended audience :** students, teachers, and researchers interested in formal languages and automata theory or software engineering, the sophisticated use of Maple as a programming language, the management of a large collaborative learning project, the use of a large Maple-oriented project to support education in software engineering (especially requirements and specification, and software verification and validation).

**abstract :** This paper outlines how Maple has been employed by the authors over the past several years in graduate-level software engineering education at Western Michigan University.

## Introduction and overview

Two courses are involved:

*cs580 – Theory of Computation* an upper-level course in formal languages and automata which uses T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science* (2e), Addison Wesley Longman, January 1998 as its text.

and *cs661 – Software Engineering II: Verification and Validation* which uses the instructor's lecture notes and an extensive reading list [B.bbt], [B.stt], [K], [KFN], [M], [Pe].

*cs580* is not primarily a software engineering course; however, software engineering topics are introduced secondarily via a major programming project component focused on **cs580Lib**, a comprehensive coherent well-planned collection of help pages, procedures, and other Maple objects, being created at Western Michigan University, that supports research and instruction in formal languages and automata theory. The ongoing **cs580Lib** implementation activities provide students an excellent implicit exposure to the management of a large industrial quality software project, especially the requirements and specification aspects. An overview of **cs580Lib** is presented at the end of this paper in *Appendix 1: Introduction to cs580Lib*, and a more detailed description is provided in [PN].

*cs661* is a course focused on software testing (both verification and validation). Many of the testing exercises associated with this course as we teach it have used Maple-oriented test targets, known as *objects under test*, or OUTs. Two kinds of Maple-oriented OUTs have been used in *cs661*:

### 1. **cs580Lib**-based OUTs

verification: the readme.txt file  
propositional\* procedure help pages  
non-propositional procedure help pages  
propositional procedure code  
non-propositional procedure code

validation: help page browser  
propositional procedure help page links  
non-propositional procedure help page links  
propositional procedures (executables)  
non-propositional procedures (executables)

At present all the **cs580Lib**-based OUTs are non-numeric in functionality.

---

\* a *propositional* procedure is one that returns a boolean value (i.e., false, true).

## 2. **MINIGOLFSCORE**-based OUTs

**MINIGOLFSCORE** (see [Pi.avvt]) is a modification of the **GOLFSCORE** case study and program introduced in Kit [K]. **MINIGOLFSCORE**-based OUTs can be both blackbox and whitebox, and include both numeric and non-numeric functionality components.

The remainder of this paper is organized into the following sections:

- Annotated checklist of software engineering curricular topics
- Historical development
- Creating a blackbox Maple procedure
- Links to some associated works
- Commentary on experiences and future plans
- References
- Appendices

### **Annotated checklist of software engineering curricular topics**

To set the context for discussing the Maple-based content of cs580 and cs661, we present in the following table a checklist of typical curricular topics in software engineering. These topics have been identified from a survey of content in three typical graduate-level textbooks on software engineering [Br], [Pr], and [SLB], and some of the lecture notes for cs580 and cs661 [e.g.: Pi.avvt]. The topics are presented in alphabetic order.

[The reader should be aware that Western Michigan University offers a number of courses dealing with software engineering and this paper considers only the content of cs580 and cs661 as taught by the authors.]

## Checklist of typical curricular topics in software engineering

Legend: **blue** software engineering topics implicitly taught in cs580  
**red** software engineering topics explicitly taught in cs661  
**green** software engineering topics with a Maple orientation

cs580	cs661	Maple	topic
	✓		activity journals
	✓		brainstorming
	✓		bug reports
	✓		checklists
			chief programmer team
✓	✓		collaborative learning
✓	✓		component testing
✓	✓		configuration management
	✓		control flow testing
	✓		critical path graphs
	✓	✓	data flow testing
✓	✓	✓	data structures
✓	✓	✓	data-centered paradigm
✓	✓	✓	documentation
	✓	✓	domain testing
✓	✓		editing
✓	✓		executive summary
	✓	✓	finite-state testing
	✓		flowcharts
	✓		formal modeling
	✓	✓	formal notation
	✓		formal system theory
	✓		formal testing
✓	✓	✓	functional decomposition
	✓	✓	implementation assumptions
✓	✓	✓	integration testing
✓	✓	✓	legacy code
	✓	✓	loop testing
✓	✓	✓	maintenance
		✓	metrics
	✓	✓	object-oriented paradigm
✓	✓	✓	oral presentation
✓	✓	✓	procedures
✓	✓	✓	programming
✓	✓	✓	project management
	✓		proof of correctness
	✓	✓	pseudo-code
✓		✓	rapid prototyping

cs580	cs661	Maple	topic
✓		✓	re-engineering
	✓		requirement decomposition
	✓		requirement partitioning
✓	✓		requirements
✓	✓	✓	research
✓			reverse engineering
✓	✓	✓	robust/non-robust programming
	✓		role playing
✓	✓	✓	specifications
			spiral/incremental model
	✓		stakeholders
✓		✓	structured programming
✓	✓	✓	style guides (documentation)
✓	✓	✓	style guides (programming)
✓	✓		system decomposition
	✓		teamwork
✓	✓	✓	templates
	✓		test analysis
	✓		test execution
	✓		test reporting
	✓		test strategy and design
✓	✓		time management
			uml
✓	✓	✓	unit testing
✓	✓	✓	upgrades
✓	✓	✓	user interfaces
✓	✓	✓	validation
✓	✓	✓	verification
	✓	✓	walk-throughs
✓	✓		waterfall model
✓	✓	✓	white box
	✓		xml

## Historical development

cs580 *Theory of Computation* is a required course in the Bachelor's and Master's curricula at Western Michigan University. The course assumes a student's familiarity with discrete mathematics (e.g., set theory, relations, functions, equivalence, countability, recursion, induction, graphs, and trees) and develops the mathematical foundations of formal languages as generated by grammars and recognized by automata and as categorized by the chomsky hierarchy. In recent years, the instruction has been based on T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science (2e)*, Addison Wesley Longman, January 1998, augmented by technical material on additional topics such as:

- extended regular expressions — adding the operators of *intersection*, *difference*, and *complement*;
- cartesian-style finite-directed graphs;
- a technique of directly constructing a deterministic finite automaton (DFA) equivalent to any extended regular expression without use of non-deterministic finite automata (NFA);
- mnemonic naming conventions for certain entities, such as variables in grammars and states in automata.

For several years prior to 2001, students had the opportunity to use Maple in a major class project. Typically, students would develop a Maple worksheet as a tutorial *living document* in which they document the design and demonstration of a Maple procedure to:

- test if an arbitrary object is a particular standard data type defined for cs580, such as *finite directed graph* or *finite tree*, or
- perform an algorithm associated with cs580, such as *minimizing the state-set of a DFA*, or
- perform a useful utility related to cs580, such as drawing in elegant format a tree graph or DFA transition graph.

These early projects used Maple as a programming language (rather than as an interactive mathematical assistant), and were each fairly independent — each student having to design and implement Maple code that "did it all" (i.e., the student did not take immediate direct advantage of any other student's complementary work).

In the summer of 2001 the authors decided to embark on a major and much more organized long-term pedagogical development project in cs580, one using Maple and attempting to:

- develop a complete *industrial strength* library of Maple data structures, procedures, and help pages to support education in formal languages and automata;
- engage successive classes of cs580 students in a major effort of coordinated collaborative undergraduate and graduate research and projects.

The name currently used to denote both our cs580 project effort and the collection of materials contained in it is **cs580Lib**.

In the winter of 2003, an experiment began in introducing Maple *objects under test* into cs661. Two types of Maple objects were used:

1. **cs580Lib**-based OUTs

verification: the readme.txt file  
propositional procedure help pages  
non-propositional procedure help pages  
propositional procedure code  
non-propositional procedure code

validation: help page browser  
propositional procedure help page links  
non-propositional procedure help page links  
propositional procedures (executables)  
non-propositional procedures (executables)

At present all the **cs580Lib**-based OUTs are non-numeric in functionality.

2. **MINIGOLFSCORE**-based OUTs

**MINIGOLFSCORE** (see [Pi.avvt]) is a modification of the **GOLFSCORE** case study and program introduced in Kit [K]. **MINIGOLFSCORE**-based OUTs can be both blackbox and whitebox, and include both numeric and non-numeric functionality components.

Figure 1 illustrates some of the chronological milestones associated with the topics addressed in this paper.

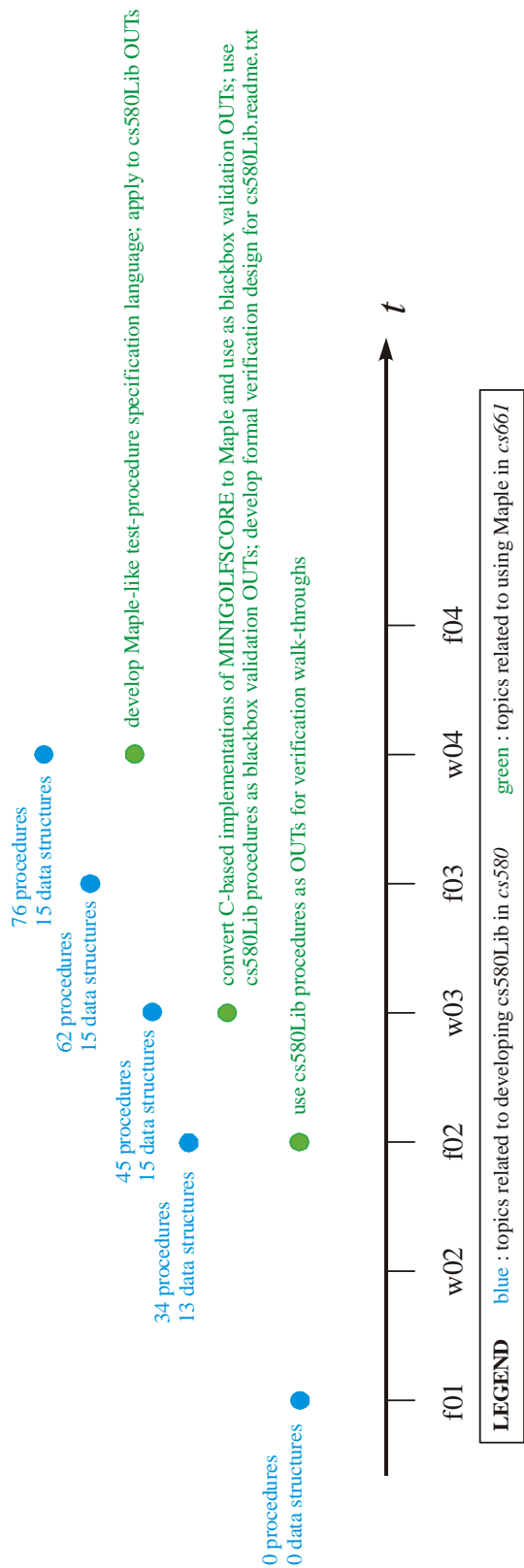


Figure 1 Chronological milestones

## Creating a blackbox Maple procedure

One of the desirable techniques in teaching software testing is that of creating blackbox software implementations to be used as OUTs. When working in a compiled software environment (such as the UNIX/C environment), this is easily accomplished by distributing only the executable object file and not the source code file for any function or program ... relying on the difficulty most users would have in reverse-compiling the object file into the original source code.

In an interpreted system, like Maple, there is no source code to object code translation associated with Maple worksheets; thus, making it difficult to hide the source code. However, in Maple, source code is translated into the equivalent of object code when a Maple *module* is converted into a set of library files. We can take advantage of this feature of the library-creating-process in Maple (plus a few other techniques) to create blackbox procedures in Maple.

The general approach is illustrated with the following example module. This material is based on a 26 April 2002 suggestion from Dr. Jason Schattman and the R&D staff of MapleSoft.

```
blackboxLib:=module()  
  
  export Print;  
  local  PrintPrivate, x;  
  
  PrintPrivate:=proc(x)  
    printf("Tom is %s.\n",x)  
  end proc;  
  
  Print:=proc()  
    x:=args;  
    PrintPrivate(x)  
  end proc;  
  
end module;
```

- "blackboxLib" is a Maple module.
- Two procedures are defined in the module – "Print" which is *exported* and "PrintPrivate" which is *local* (hidden).
- The functionality of "Print" is totally provided by "PrintPrivate" – i.e., "Print" passes its actual argument sequence in a call to "PrintPrivate" which executes with side effects and a return. The return value of "PrintPrivate" is in turn returned by "Print".
- To make "Print" a blackbox procedure, disseminate the module as a library object – not as a worksheet. This will make the source code representation of the "PrintPrivate" *compiled*, if you will, and difficult to reconstruct by any library user.

A Maple worksheet with a more detailed presentation of the above example is provided in *blackboxCreationAndDemonstration.mws* in <http://www.cs.wmich.edu/~piat/cs661/> ...

## Links to some associated works

1. A directory containing the latest version(s) of cs580Lib.  
Located at <http://www.cs.wmich.edu/~piat/cs580/...>  
Each cs580Lib is a self-extracting zipped file.

2. A directory of files supporting cs661.  
Located at <http://www.cs.wmich.edu/~piat/cs661/...>

Includes:

- a. *The Architecture of Verification and Validation Testing(pdf)*, current version of the major lecture notes supporting cs661, presents a formal theory of testing and includes details on MINIGOLFSCORE.

- b. cs580LibOut.exe -- A library containing a collection of ten (10) blackbox OUTs implemented in Maple, each a candidate implementation of the help browser portion of cs580Lib. A self-extracting zipped file.

This set of OUTs is used in conjunction with *popquiz4* which is copied in Appendix 2.

- c. MINIGOLFSCOREOutLib -- A library containing a collection of ten (10) blackbox OUTs, each a candidate implementation of MINIGOLFSCORE.

This set of OUTs is used in conjunction with *cs661 assignment8* which is copied in Appendix 3.

- d. *blackboxCreationAndDemonstration.mws* -- the Maple worksheet discussed in the previous section (creating a blackbox Maple procedure).

In <http://www.cs.wmich.edu/~piat/cs661/...>

3. A directory of several chronologically different versions of cs580Lib. Each is a self-extracting zipped file. In <http://www.cs.wmich.edu/~piat/cs580/...>

4. A directory of files supporting cs661. In <http://www.cs.wmich.edu/~piat/cs661/...>  
Includes:

- a. *The Architecture of Verification and Validation Testing*, one of the lecture notes supporting cs661, presents a formal theory of testing and includes details on MINIGOLFSCORE.

- b. A collection of ten (10) blackbox OUTs, each a candidate implementation of the help browser portion of cs580Lib. This set of OUTs is used in conjunction with *popquiz4* which is copied in Appendix 2.

- c. A collection of ten (10) blackbox OUTs, each a candidate implementation of MINIGOLFSCORE. This set of OUTs is used in conjunction with *assignment8* which is copied in Appendix 3.
- d. *blackboxCreationAndDemonstration.mws*, the Maple7 worksheet discussed in the previous section (creating a blackbox Maple procedure).

## Commentary on experiences and future plans

Some observations deriving from the effort to date include:

- The process of creating an industrial strength library in support of *cs580* has been a very satisfying one. The initial hopes for this project as reported in [PN] are being realized. The resulting library has integrity and usefulness.
- Maple continues to demonstrate that it is an excellent programming language and environment for the **cs580Lib** application.
- **cs580Lib** as a case study in our graduate software testing course has worked out well. It is satisfying to be able to use what is now a really large home-grown student-produced project in such a way.
- The students of *cs580* and *cs661* have been exposed to a project that is of long duration, large, and managed with some care for good software engineering practice, including:
  - implementing in conformance with rigorous semi-formal specifications,
  - testing with respect to rigorous semi-formal specifications,
  - dealing with configuration management concerns,
  - producing high quality implementation documentation.
- The testing of **cs580Lib** has been only partial (a few help pages, a few procedures) but has, nevertheless, resulted in the finding of a few errors in the help pages and procedures of the library.

If the future provides opportunity to continue the multi-dimensional experiments reported on in this paper, then some of the directions we would hope to explore include:

- The topics in the **cs580Lib** project can be used as targets in our *cs660* graduate course, *Software Engineering I: Requirements and Specification*.
- The current versions of **cs580Lib** are limited to constructs based on finite sets. In the future we would like to extend the functionality of our Maple objects to include infinite sets and constructs built on them.
- At some point we would like to begin illustrating the power and elegance of **cs580Lib** by applying it to several illustrative problems that are larger and more realistic than those treated in the text.

## References

- [B] E.J. Braude, *Software Engineering: An Object-Oriented Perspective*, John Wiley & Sons, 2001.
- [B.bbt] B. Beizer, *Black-Box Testing: Techniques for Functional Testing of Software and Systems*, John Wiley & Sons, 1995.
- [B.stt] B. Beizer, *Software Testing Techniques* (2e), International Thomson Computer Press, 1990.
- [K] E. Kit, *Software Testing in the Real World: Improving the Process*, Addison-Wesley, 1995.
- [KFN] C. Kaner, J. Falk, H.Q. Nguyen, *Testing Computer Software*, John Wiley & Sons, 1999.
- [M] B. Marick, *The Craft of Software Testing: Subsystem Testing Including Object-Based and Object-Oriented Testing*, Prentice Hall, 1995.
- [Pe] W.F. Perry, *Effective Methods for Software Testing* (2e), John Wiley & Sons, 1999.
- [Pi.avvt] T.F. Piatkowski, *The Architecture of Verification and Validation Testing*. Available online at: <http://www.cs.wmich.edu/~piat/cs661/...>
- [PN] “cs580Lib – a comprehensive library of Maple objects supporting instruction and research in formal languages and automata theory,” *Proceedings of the Maple Summer Workshop 2002*, Waterloo, Ontario, July 28-30, 2002 (with J.D. Nelson). Available online at: <http://www.cs.wmich.edu/~piat/professions.html>
- [Pr] R.S. Pressman, *Software Engineering: A Practitioner's Approach* (5e), McGraw-Hill, 2001.
- [S] T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science* (2e), Addison Wesley Longman, January 1998.
- [SLB] E. Stiller and C. LeBlanc, *Project-Based Software Engineering: An Object-Oriented Approach*, Addison-Wesley, 2002.
- [WM.m9apg] *Maple 9 Advanced Programming Guide*, Waterloo Maple, Waterloo ON Canada, 2003.
- [WM.m9gsg] *Maple 9 Getting Started Guide*, Waterloo Maple, Waterloo ON Canada, 2003.

[WM.m9ipg] *Maple 9 Introductory Programming Guide*, Waterloo Maple, Waterloo ON Canada, 2003.

[WM.m9lg] *Maple 9 Learning Guide*, Waterloo Maple, Waterloo ON Canada, 2003.

## **Appendix 1: Introduction to cs580Lib**

A copy of the help page from **cs580Lib** that explains the purpose and scope of the library, and how to install and link to it.

## Introduction to cs580Lib

**cs580Lib** is a collection of procedures and other Maple objects created at Western Michigan University to support research and instruction in formal languages and automata theory, in particular our upper-level course: cs580 -- Theory of Computation.

The material in **cs580Lib** is supported by a comprehensive set of help files that are accessible through the Maple online browser.

**cs580Lib** has been undertaken for several reasons, including:

- The materials included in **cs580Lib** are useful in their own right.
- The activities involved in creating the materials in **cs580Lib** provide an excellent learning opportunity for students of formal languages and automata theory.
- The total project is a good educational exercise in using Maple in an elegant large-scale application.

**cs580Lib** is presented to our students, colleagues, and all readers, in the hope that they will find here a useful toolkit for study and research in the theory of computation, and a paradigm for using the computer (and especially Maple) in that endeavor.

## Current content of cs580Lib

cs580Lib contains

- an introductory help page (i.e., this help page)
- a help page on each of the following data structures

CFGDataStructure  
CFGDerivationDataStructure  
DFAComputationDataStructure  
DFADDataStructure  
OFTDataStructure  
UFTDataStructure  
arcPathDataStructure  
finiteDirectedGraphDataStructure  
functionDataStructure  
nodePathDataStructure  
partitionDataStructure  
propositionDataStructure

reDataStructure  
relationDataStructure  
stringDataStructure

- a help page on each of the following procedures (in addition, those procedures marked with asterisks are implemented)

\*Ancestors  
\*Children  
\*Descendents  
\*Frontier  
\*N  
\*NodeValues  
\*Range  
\*ReachableOne  
\*ReachableZero  
\*Siblings  
\*cartesianProduct  
\*cfgDerivable  
\*cfgDerivations  
  cfgDrawDerivation  
\*cfgPrintDerivation  
\*cfgSentences  
\*cfgSententialForms  
\*convertCs580StringToString  
  convertDFAToRE  
\*convertERToPartition  
\*convertFunctionToTable  
\*convertPartitionToER  
  convertREToDFA  
  convertREToRestrictedRE  
  convertRGToDFA  
\*convertTableToFunction  
\*depth  
  dfaConcatenation  
  dfaDifference  
  dfaIntersect  
  dfaInverse

dfaReachable  
dfaReachingInputs  
dfaStar  
dfaUnion  
\*equivalenceClass  
\*implicitSet  
\*inDegree  
\*isAcyclicFDG  
\*isArcPath  
\*isCFG  
\*isCFGDerivable  
\*isCFGDerivation  
\*isCFGLeftmostDerivation  
\*isCFGRightmostDerivation  
\*isCompleteBinaryFT  
\*isCycle  
\*isDFA  
  isDFAAccepted  
  isDFAComputation  
\*isER  
\*isF  
\*isFDG  
\*isFL  
\*isLeaf  
\*isNodePath  
\*isNullArcPath  
\*isNullNodePath  
\*isOFT  
\*isOneToOneF  
\*isPalindrome  
\*isPartition  
\*isProperSubset  
\*isProposition  
\*isR  
\*isRE  
\*isRG  
\*isRLG

```

*isRR
*isSR
*isStrictlyBinaryFT
*isString
*isSubset
*isTR
*isUFT
*minCommonAncestor
*minDFA
*outDegree
*parent
*partitionBlock
*projList
*projListSet
*reComplement
*reConcatenate
*reIntersect
  reMember
*reMinus
*reStar
*reUnion
*stringConcatenate
*stringReverse

```

## How to get started

To install **cs580Lib** read the instructions included in the *readme.txt* worksheet that accompanies the installation files, and follow the instructions.

Assuming that **cs580Lib** is already installed (here we assume the installation is on the local drive C):

- To access the **cs580Lib** help files through the Maple browser and the **cs580Lib** procedures:
  - add `c:\cs580Lib` to *libname*, and
  - *with* the `cs580Lib` module to your worksheet.
- Both of the above steps are typically accomplished by the 2-command-sequence:

```

libname:="c:\\cs580Lib",libname;
with(cs580Lib);

```

## References

The material presented in cs580Lib is based on definitions and other subject matter taken from a number of sources, including:

- C. Berge (translated by A. Doig), *The Theory of Graphs*, Methuen, London, 1958.
- J.E. Hopcroft, R. Motwani, and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation (2e)*, Addison Wesley Longman, 2001.
- T.A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science (2e)*, Addison Wesley Longman, January 1998.

## Acknowledgements

A debt of appreciation is owed to all who have contributed to this collection. Authors of both the help files and the utility procedures are identified individually in the associated help files accessed through the browser.

## Whom to contact

We would like to hear from you if you have any suggestions, contributions, or criticisms that would help improve **cs580Lib**.

[Thomas F. Piatkowski](#)

Department of Computer Science  
Western Michigan University  
Kalamazoo MI 49008  
(269) 276-3115  
thomas.piatkowski@wmich.edu

## Help file author:

- Thomas F. Piatkowski

## Version:

2004 January 29

## **Appendix 2: cs661 popquiz4**

An in-class online exercise to perform ad hoc validation on OUTs comprising candidate cs580Lib help browsers.

# CS 661 (51867) Software Engineering II: Verification and Validation of Software Systems

Spring 2004

Piatkowski

popquiz4

cs580Lib ad hoc blackbox validation exercise

open book

you can collaborate in any groupings you wish

---

## Preliminaries

The firm is evaluating the programming skills of ten (10) job applicants.

As part of the assessment process we have asked each applicant to create a Maple library duplicating the navigator only portion of our existing cs580Lib; procedure implementation code was not requested and is NOT included.

Management requests that the team perform a crash-effort ad hoc validation of the work of these applicants and that a set of concise evaluation reports be provided.

The ten applicant library files are available to you in the directory *cs580LibOut* which will be made available to you online; the library files are grouped in the subdirectories: *cs580Lib1*, *cs580Lib2*, ..., *cs580Lib10*.

---

## Assignment

Validate each of these ten implementations and report back to management.

One report per student.

You may organize and collaborate to cover as many aspects of the validation task as time and other resources permit. BUT, one report per student is required.

Each student's report may be submitted hardcopy or electronically by the end of class.

Consider using the report philosophy and templates proposed in *The Architecture of Verification and Validation Testing*. It would be very helpful if, for each (?primitive) validation you explicitly stated the associated (?primitive) requirement.

Time is limited; do your best accordingly.

### **Appendix 3: cs661 assignment8**

An exercise to perform ad hoc validation on OUTs comprising candidate MINIGOLFSCORE implementations.

-----  
assignment8 -- ad hoc MINIGOLFSCORE() blackbox validation exercise

made : 2003 March 12 Wednesday

due : 2003 March 17 Monday  
-----

\*\*\* each student is to work alone on this assignment \*\*\*  
-----

Preliminaries  
-----

The class subdirectory

/home/cs/piat/661/MINIGOLFSCORE/blackboxLib/

contains two library files

maple.ind  
maple.lib

Download these two files to a local directory of your choice, denoted by <yourLocalDir>.

When in a Maple worksheet you can then link to the library via:

```
libname:="<yourLocalDir>",libname;
```

-----

Assignment  
-----

The library contains ten different implementations of MINIGOLFSCORE, named:

MINIGOLFSCORE1, MINIGOLFSCORE2, ... , MINIGOLFSCORE10

Your assignment is to validate each of these ten implementations and report back to management. One report per student.

Turn in two (2) copies of your report.  
-----

endAssignment8  
-----