CS 322-Software Engineering Practicum

Syllabus-Fall 2008

Instructor: Lewis Barnett  Jepson Hall 212A
            (804) 289-8091
            lbarnett@richmond.edu
            http://www.mathcs.richmond.edu/~lbarnett/
Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that the work you will be doing is organized primarily around one large project which you will be working on collectively. We will use lecture meetings to cover some of the standard techniques of Software Engineering and for various software design exercises. The consulting laboratory component will consist of planning and status meetings for the work your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most likely encounter after graduation. You will be dealing with a number of issues that you have not faced before, such as how large software systems should be structured, how to design with future modifications in mind, how to design software with testing in mind, and how to manage the interaction among team members. My philosophy in this course is to run it with a light hand: you are responsible, personally and collectively, for the success of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and having one or two other 300 level courses under your belt is a good idea, though not strictly necessary.

Readings

Textbook:


Other recommended readings:

Sinan Si Alhir, *UML in a Nutshell*, O’Reilly, 1998. (The library has one copy of this. I have put one chapter, which is a UML tutorial, on e-reserve.)

You may want to buy a reference book for Python. There is a very good tutorial by the designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual assignments</td>
<td>55</td>
</tr>
<tr>
<td>Overall project grade</td>
<td>25</td>
</tr>
<tr>
<td>Individual contribution</td>
<td>-25 to +30</td>
</tr>
</tbody>
</table>

Maximum possible points 110

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grade is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies
CS 322-Software Engineering Practicum

Syllabus-Fall 2008

Instructor: Lewis Barnett  
Jepson Hall 212A  
(804) 289-8091  
lbarnett@richmond.edu  
http://www.mathcs.richmond.edu/~lbarnett/  
Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that the work you will be doing is organized primarily around one large project which you will be working on collectively. We will use lecture meetings to cover some of the standard techniques of Software Engineering and for various software design exercises. The consulting laboratory component will consist of planning and status meetings for the work your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most likely encounter after graduation. You will be dealing with a number of issues that you have not faced before, such as how large software systems should be structured, how to design with future modifications in mind, how to design software with testing in mind, and how to manage the interaction among team members. My philosophy in this course is to run it with a light hand: you are responsible, personally and collectively, for the success of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and having one or two other 300 level courses under your belt is a good idea, though not strictly necessary.

Readings

Textbook:


Other recommended readings:

Sinan Si Alhir, *UML in a Nutshell*, O'Reilly, 1998. (The library has one copy of this. I have put one chapter, which is a UML tutorial, on e-reserve.)

You may want to buy a reference book for Python. There is a very good tutorial by the designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

- Individual assignments: 55 points
- Overall project grade: 25 points
- Individual contribution: -25 to +30 points

Maximum possible points: 110

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grad is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies
Instructor: Lewis Barnett  
Jepson Hall 212A  
(804) 289-8091  
lbarnett@richmond.edu  
http://www.mathcs.richmond.edu/~lbarnett/  
Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that the work you will be doing is organized primarily around one large project which you will be working on collectively. We will use lecture meetings to cover some of the standard techniques of Software Engineering and for various software design exercises. The consulting laboratory component will consist of planning and status meetings for the work your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most likely encounter after graduation. You will be dealing with a number of issues that you have not faced before, such as how large software systems should be structured, how to design with future modifications in mind, how to design software with testing in mind, and how to manage the interaction among team members. My philosophy in this course is to run it with a light hand: you are responsible, personally and collectively, for the success of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and having one or two other 300 level courses under your belt is a good idea, though not strictly necessary.

Readings

Textbook:

Other recommended readings:
Sinan Si Alhir, *UML in a Nutshell*, O’Reilly, 1998. (The library has one copy of this. I have put one chapter, which is a UML tutorial, on e-reserve.)
You may want to buy a reference book for Python. There is a very good tutorial by the designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual assignments</td>
<td>55</td>
</tr>
<tr>
<td>Overall project grade</td>
<td>25</td>
</tr>
<tr>
<td>Individual contribution</td>
<td>-25 to +30</td>
</tr>
<tr>
<td>Maximum possible points</td>
<td>110</td>
</tr>
</tbody>
</table>

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grad is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies
CS 322-Software Engineering Practicum

Syllabus-Fall 2008

Instructor: Lewis Barnett  Jepson Hall 212A
              (804) 289-8091
              lbarnett@richmond.edu
              http://www.mathcs.richmond.edu/~lbarnett/

Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that
the work you will be doing is organized primarily around one large project which you will
be working on collectively. We will use lecture meetings to cover some of the standard
techniques of Software Engineering and for various software design exercises. The con-
sulting laboratory component will consist of planning and status meetings for the work
your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most
likely encounter after graduation. You will be dealing with a number of issues that you
have not faced before, such as how large software systems should be structured, how to
design with future modifications in mind, how to design software with testing in mind,
and how to manage the interaction among team members. My philosophy in this course is
to run it with a light hand: you are responsible, personally and collectively, for the success
of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and
having one or two other 300 level courses under your belt is a good idea, though not
strictly necessary.

Readings

Textbook:

Stephen R. Schach, Classical and Object-oriented Software Engineering with UML and

Other recommended readings:

Sinan Si Alhir, UML in a Nutshell, O’Reilly, 1998. (The library has one copy of this. I
have put one chapter, which is a UML tutorial, on e-reserve.)

You may want to buy a reference book for Python. There is a very good tutorial by the
designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

- Individual assignments: 55 points
- Overall project grade: 25 points
- Individual contribution: -25 to +30 points

**Maximum possible points: 110**

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grade is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies
CS 322-Software Engineering Practicum

Syllabus-Fall 2008

Instructor: Lewis Barnett  Jepson Hall 212A
(804) 289-8091
lbarnett@richmond.edu
http://www.mathcs.richmond.edu/~lbarnett/

Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that the work you will be doing is organized primarily around one large project which you will be working on collectively. We will use lecture meetings to cover some of the standard techniques of Software Engineering and for various software design exercises. The consulting laboratory component will consist of planning and status meetings for the work your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most likely encounter after graduation. You will be dealing with a number of issues that you have not faced before, such as how large software systems should be structured, how to design with future modifications in mind, how to design software with testing in mind, and how to manage the interaction among team members. My philosophy in this course is to run it with a light hand: you are responsible, personally and collectively, for the success of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and having one or two other 300 level courses under your belt is a good idea, though not strictly necessary.

Readings

Textbook:

Other recommended readings:
Sinan Si Alhir, UML in a Nutshell, O'Reilly, 1998. (The library has one copy of this. I have put one chapter, which is a UML tutorial, on e-reserve.)
You may want to buy a reference book for Python. There is a very good tutorial by the designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

- Individual assignments: 55 points
- Overall project grade: 25 points
- Individual contribution: -25 to +30 points

Maximum possible points: 110

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grade is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies
CS 322-Software Engineering Practicum

Syllabus-Fall 2008

Instructor: Lewis Barnett  
Jepson Hall 212A  
(804) 289-8091  
lbarnett@richmond.edu  
http://www.mathcs.richmond.edu/~lbarnett/

Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that the work you will be doing is organized primarily around one large project which you will be working on collectively. We will use lecture meetings to cover some of the standard techniques of Software Engineering and for various software design exercises. The consulting laboratory component will consist of planning and status meetings for the work your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most likely encounter after graduation. You will be dealing with a number of issues that you have not faced before, such as how large software systems should be structured, how to design with future modifications in mind, how to design software with testing in mind, and how to manage the interaction among team members. My philosophy in this course is to run it with a light hand: you are responsible, personally and collectively, for the success of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and having one or two other 300 level courses under your belt is a good idea, though not strictly necessary.

Readings

Textbook:

Other recommended readings:
Sinan Si Alhir, UML in a Nutshell, O’Reilly, 1998. (The library has one copy of this. I have put one chapter, which is a UML tutorial, on e-reserve.)
You may want to buy a reference book for Python. There is a very good tutorial by the designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

<table>
<thead>
<tr>
<th>Component</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual assignments</td>
<td>55</td>
</tr>
<tr>
<td>Overall project grade</td>
<td>25</td>
</tr>
<tr>
<td>Individual contribution</td>
<td>-25 to +30</td>
</tr>
<tr>
<td>Maximum possible points</td>
<td>110</td>
</tr>
</tbody>
</table>

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grade is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies
CS 322-Software Engineering Practicum

Syllabus-Fall 2008

Instructor: Lewis Barnett  
Jepson Hall 212A  
(804) 289-8091  
lbarnett@richmond.edu  
http://www.mathcs.richmond.edu/~lbarnett/

Office hours: MWF 2:00 PM - 3:00 PM, or by appointment

Course Description

The Software Engineering Practicum is unique in the Computer Science curriculum in that the work you will be doing is organized primarily around one large project which you will be working on collectively. We will use lecture meetings to cover some of the standard techniques of Software Engineering and for various software design exercises. The consulting laboratory component will consist of planning and status meetings for the work your group is doing on the term project.

The purpose of this course is to simulate the type of work environment you will most likely encounter after graduation. You will be dealing with a number of issues that you have not faced before, such as how large software systems should be structured, how to design with future modifications in mind, how to design software with testing in mind, and how to manage the interaction among team members. My philosophy in this course is to run it with a light hand: you are responsible, personally and collectively, for the success of the project.

Prerequisites

The prerequisites for this course are somewhat flexible. CS 315 is good background, and having one or two other 300 level courses under your belt is a good idea, though not strictly necessary.

Readings

Textbook:


Other recommended readings:

Sinan Si Alhir, *UML in a Nutshell*, O’Reilly, 1998. (The library has one copy of this. I have put one chapter, which is a UML tutorial, on e-reserve.)

You may want to buy a reference book for Python. There is a very good tutorial by the designer of the language, Guido van Rossum, at http://docs.python.org/tut/tut.html.
Frederick P. Brooks, Jr., *The Mythical Man-month*, Addison-Wesley, 1995. (The library has two copies of this.) This is a collection of essays about the development of a historically important IBM mainframe operating system which remain relevant today. A classic that belongs on any software developer’s bookshelf.

Additional readings from the technical literature and other sources are on reserve at Boatwright library.

**Grading Policy**

Your grade will be based on a number of individual assignments and the instructor’s evaluation of the quality of the finished group project plus an evaluation of your individual contribution to the development effort. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgement. Part of the overall grade for the project will be how well the team met the deadlines for various parts of the project.

The grade will be divided as follows:

- Individual assignments: 55 points
- Overall project grade: 25 points
- Individual contribution: -25 to +30 points

Maximum possible points: 110

Scale: A: 90 - 110; B: 80 - 89; C: 70 - 79; D: 60 - 69; F: 0 - 59

If you do the absolute minimum amount of work possible on the group project, your individual score will be 0. Note that even if the overall project grade is an A, the best individual grade you could receive under those circumstances is a C. If you fail to perform your duties, your individual score will be negative.

There will be no final exam.

**Topics**

The following is an approximate ordering of topics we will be covering in lecture.

- Software development teams and roles
- Software life cycle models
- Requirements engineering
- The Object Model/Object technology
- Object-oriented design
- Unified Modeling Language
- Software architectures
- Object decomposition
- Use-case modeling
Design exercises
Testing
Configuration management
Project-specific technologies