Software Engineering Seminar

Course Description

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2025-III

Outline

1. You don't know who I am
2. Course Overview
3. Syllabus
4. Grading & Rules
5. Bibliography

Outline

1. You don't know who I am
2. Course Overview
3. Syllabus
4. Grading & Rules
5. Bibliography

Academic Experience

- **Computer Engineer**, M.Sc. in Computer Engineering, and *researcher* for 16 years.
- 8 years as **full-time associate professor** at colleges, in Computer Engineering programs.
- 3 years as **lecturer professor** for both colleges and government STEM programs.
- Speaker at IEEE events and colleges in Colombia, Brazil, and Bolivia.

Academic Experience

- **Computer Engineer**, M.Sc. in Computer Engineering, and *researcher* for 16 years.
- 8 years as **full-time associate professor** at colleges, in Computer Engineering programs.
- 3 years as **lecturer professor** for both colleges and government STEM programs.
- Speaker at IEEE events and colleges in Colombia, Brazil, and Bolivia.

Academic Experience

- **Computer Engineer**, M.Sc. in Computer Engineering, and *researcher* for 16 years.
- 8 years as **full-time associate professor** at colleges, in Computer Engineering programs.
- 3 years as **lecturer professor** for both colleges and government STEM programs.
- Speaker at IEEE events and colleges in Colombia, Brazil, and Bolivia.

Academic Experience

- **Computer Engineer**, M.Sc. in Computer Engineering, and *researcher* for 16 years.
- 8 years as **full-time associate professor** at colleges, in Computer Engineering programs.
- 3 years as **lecturer professor** for both colleges and government STEM programs.
- Speaker at IEEE events and colleges in Colombia, Brazil, and Bolivia.

Non-academic Experience

- PyCon Colombia and Python Bogotá **co-organizer**.
- 3 years as **software engineer** for several tech companies in Colombia.
- 3 years as **Technical Leader** of Machine Learning and Data Science at a USA startup.
- 1.5 years as **MLOps Engineer** for a Fintech company in LATAM.
- Currently, **Technical Leader** of Data Engineering and Machine Learning at `Blend 360`.

Non-academic Experience

- PyCon Colombia and Python Bogotá **co-organizer**.
- 3 years as **software engineer** for several tech companies in Colombia.
- 3 years as **Technical Leader** of Machine Learning and Data Science at a USA startup.
- 1.5 years as **MLOps Engineer** for a Fintech company in LATAM.
- Currently, **Technical Leader** of Data Engineering and Machine Learning at Blend 360.

Non-academic Experience

- PyCon Colombia and Python Bogotá **co-organizer**.
- 3 years as **software engineer** for several tech companies in Colombia.
- 3 years as **Technical Leader** of Machine Learning and Data Science at a USA startup.
- 1.5 years as **MLOps Engineer** for a Fintech company in LATAM.
- Currently, **Technical Leader** of Data Engineering and Machine Learning at `Blend 360`.

Non-academic Experience

- PyCon Colombia and Python Bogotá **co-organizer**.
- 3 years as **software engineer** for several tech companies in Colombia.
- 3 years as **Technical Leader** of Machine Learning and Data Science at a USA startup.
- 1.5 years as **MLOps Engineer** for a Fintech company in LATAM.
- Currently, **Technical Leader** of Data Engineering and Machine Learning at `Blend 360`.

Non-academic Experience

- PyCon Colombia and Python Bogotá **co-organizer**.
- 3 years as **software engineer** for several tech companies in Colombia.
- 3 years as **Technical Leader** of Machine Learning and Data Science at a USA startup.
- 1.5 years as **MLOps Engineer** for a Fintech company in LATAM.
- Currently, **Technical Leader** of Data Engineering and Machine Learning at `Blend 360`.

Outline

1. You don't know who I am
2. Course Overview
3. Syllabus
4. Grading & Rules
5. Bibliography

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Overview

This course is designed to introduce undergraduate students to the fundamental concepts of **software engineering**, including *requirements engineering*, **agile methodologies**, and **collaborative development practices**.

The main focus of the course is on software testing engineering. Students will learn about **testing principles**, **test design techniques**, and **automation tools** to ensure software quality. The course covers **unit testing**, **integration testing**, **system testing**, and **test-driven development (TDD)** within agile frameworks.

Classes will include lectures, practical exercises, and **team project**. By the end of the course, students will be able to **define requirements**, apply agile practices, and implement effective **testing strategies** in real-world software projects.

Overview

This course is designed to introduce undergraduate students to the fundamental concepts of **software engineering**, including *requirements engineering*, **agile methodologies**, and **collaborative development practices**.

The main focus of the course is on software testing engineering. Students will learn about **testing principles**, **test design techniques**, and **automation tools** to ensure software quality. The course covers **unit testing**, **integration testing**, **system testing**, and **test-driven development (TDD)** within agile frameworks.

Classes will include lectures, practical exercises, and **team project**. By the end of the course, students will be able to **define requirements**, apply agile practices, and implement effective **testing strategies** in real-world software projects.

Overview

This course is designed to introduce undergraduate students to the fundamental concepts of **software engineering**, including *requirements engineering*, **agile methodologies**, and **collaborative development practices**.

The main focus of the course is on software testing engineering. Students will learn about **testing principles**, **test design techniques**, and **automation tools** to ensure software quality. The course covers **unit testing**, **integration testing**, **system testing**, and **test-driven development (TDD)** within agile frameworks.

Classes will include lectures, practical exercises, and **team project**. By the end of the course, students will be able to **define requirements**, apply agile practices, and implement effective **testing strategies** in real-world software projects.

Course Goals

The main goal of this course is to provide students with a solid understanding of **software engineering fundamentals**, with a strong emphasis on software testing.

By the end of the course, you should be able to:

- **Elicit and document requirements** for software projects.
- Apply agile methodologies and collaborative practices in development teams.
- Understand and implement **testing principles** and **test design techniques**.
- Develop and execute **unit**, **integration**, and **system tests**.
- Use **automation tools** and apply **test-driven development (TDD)**.
- Evaluate and improve software quality through effective testing strategies.

Course Goals

The main goal of this course is to provide students with a solid understanding of **software engineering fundamentals**, with a strong emphasis on software testing.

By the end of the course, you should be able to:

- **Elicit and document requirements** for software projects.
- Apply agile methodologies and collaborative practices in development teams.
- Understand and implement **testing principles** and **test design techniques**.
- Develop and execute **unit**, **integration**, and **system tests**.
- Use **automation tools** and apply **test-driven development (TDD)**.
- Evaluate and improve software quality through effective testing strategies.

Prerequisites

This is a basic course, so you must have some knowledge of:

- *Programming* in Java, Python, or C++.
- *Foundations* of Object-Oriented Programming.
- Basic concepts of **UML** and Class Diagrams.
- Basic usage of Git and `GitHub`. Pull Request
- Basic concepts of `data systems` and the relational model.
- Use of *IDEs* such as VS Code, Eclipse, or PyCharm.

Outline

1 You don't know who I am

2 Course Overview

3 Syllabus

4 Grading & Rules

5 Bibliography

Syllabus I

Topic	Time
Software Engineering Introduction	3 sessions
Agile Methodologies ✓	2 sessions
Project Management ✓	2 sessions
Requirements Engineering ✓	3 sessions
System Analysis & Design ✓	2 sessions
Software Architectures Fundamentals ✓	2 sessions
Testing Engineering Fundamentals ✓	2 sessions
Catch Up — Course Project	2 sessions

Table: Course Schedule — Period I

Syllabus II

Period	Topic	Time
Period II	Unit Testing	4 sessions
	Integration Testing	2 sessions
	Acceptance Testing	3 sessions
	System Performance Testing	2 sessions
	Final Test	1 session
Period III	Project Dissertation	2 sessions

Table: Course Schedule — Period II & III

→ Deploy

Outline

1 You don't know who I am

2 Course Overview

3 Syllabus

4 Grading & Rules

5 Bibliography

Grades Percentages

Period	Item	Percentage
Period I	Assignments	5%
	Workshops	20%
	Project Catch-Up	10%
Period II	Assignments	5%
	Workshops	20%
	Test	10%
Period III	Paper + Poster	5%
	Report + Implementation	15%
	Presentation	10%

Table: Software Engineering Seminar — Grades Distribution

2

30%.

80%.

30%.

Don't hate the player, hate the game

- All assignments must be submitted handwritten on **time** and in **English**. Grammar and spelling will **not** be evaluated.
- Copying and pasting from the internet are **forbidden**. Please **develop** your own ideas and solutions.
- Class attendance is **not mandatory**. If you **miss** classes, you must *study independently*.
- No cell phones, no smartwatches, no WhatsApp, no Tinder, no smart-anything. **Just you and your brain**. Pay attention in class.
- Communication with me must be via **email** or **Slack**. I will **not** answer any questions via *WhatsApp*.

Don't hate the player, hate the game

- All assignments must be submitted handwritten on **time** and in **English**. Grammar and spelling will **not** be evaluated.
- Copying and pasting from the internet are **forbidden**. Please **develop** your own ideas and solutions.
- Class attendance is **not mandatory**. If you **miss** classes, you must *study independently*.
- No cell phones, no smartwatches, no WhatsApp, no Tinder, no smart-anything. **Just you and your brain**. Pay attention in class.
- Communication with me must be via **email** or **Slack**. I will **not** answer any questions via *WhatsApp*.

Don’t hate the player, hate the game

- All assignments must be submitted handwritten on **time** and in **English**. Grammar and spelling will **not** be evaluated.
- Copying and pasting from the internet are **forbidden**. Please **develop** your own ideas and solutions.
- Class attendance is **not mandatory**. If you **miss** classes, you must *study independently*.
- No cell phones, no smartwatches, no WhatsApp, no Tinder, no smart-anything. **Just you and your brain**. Pay attention in class.
- Communication with me must be via **email** or **Slack**. I will **not** answer any questions via *WhatsApp*.

Don't hate the player, hate the game

- All assignments must be submitted handwritten on **time** and in **English**. Grammar and spelling will **not** be evaluated.
- Copying and pasting from the internet are **forbidden**. Please **develop** your own ideas and solutions.
- Class attendance is **not mandatory**. If you **miss** classes, you must *study independently*.
- No cell phones, no smartwatches, no WhatsApp, no Tinder, no smart-anything. **Just you and your brain**. Pay attention in class.
- Communication with me must be via **email** or **Slack**. I will **not** answer any questions via *WhatsApp*.

Don't hate the player, hate the game

- All assignments must be submitted handwritten on **time** and in **English**. Grammar and spelling will **not** be evaluated.
- Copying and pasting from the internet are **forbidden**. Please **develop** your own ideas and solutions.
- Class attendance is **not mandatory**. If you **miss** classes, you must *study independently*.
- No cell phones, no smartwatches, no WhatsApp, no Tinder, no smart-anything. **Just you and your brain**. Pay attention in class.
- Communication with me must be via **email** or **Slack**. I will **not** answer any questions via *WhatsApp*.

Code of Conduct

- Always be **respectful** to your classmates and to me. You must be **kind** to everyone inside (*and outside*) the classroom.
- There is no best programming language, tool, or technology. There are only **better** or **worse** solutions.
- You must be **honest** with your work. If you don't know something, just **ask** me. I will be glad to help you.
- You must be **responsible** with your work. If you don't submit **on time**, please don't complain.
- You must not be **disruptive** or **negatively affect** the **classroom environment**. If you do, I will ask you to **leave** the classroom.

Code of Conduct

- Always be **respectful** to your classmates and to me. You must be **kind** to everyone inside (*and outside*) the classroom.
- There is no best programming language, tool, or technology. There are only **better** or **worse** solutions.
- You must be **honest** with your work. If you don't know something, just **ask** me. I will be glad to help you.
- You must be **responsible** with your work. If you don't submit **on time**, please don't complain.
- You must not be **disruptive** or **negatively affect** the **classroom environment**. If you do, I will ask you to **leave** the classroom.

Code of Conduct

- Always be **respectful** to your classmates and to me. You must be **kind** to everyone inside (*and outside*) the classroom.
- There is no best programming language, tool, or technology. There are only **better** or **worse** solutions.
- You must be **honest** with your work. If you don't know something, just **ask** me. I will be glad to help you.
- You must be **responsible** with your work. If you don't submit **on time**, please don't complain.
- You must not be **disruptive** or **negatively affect** the **classroom environment**. If you do, I will ask you to **leave** the classroom.

Code of Conduct

- Always be **respectful** to your classmates and to me. You must be **kind** to everyone inside (*and outside*) the classroom.
- There is no best programming language, tool, or technology. There are only **better** or **worse** solutions.
- You must be **honest** with your work. If you don't know something, just **ask** me. I will be glad to help you.
- You must be **responsible** with your work. If you don't submit **on time**, please don't complain.
- You must not be **disruptive** or **negatively affect** the **classroom environment**. If you do, I will ask you to **leave** the classroom.

6:00 → 5:59.30

Code of Conduct

- Always be **respectful** to your classmates and to me. You must be **kind** to everyone inside (*and outside*) the classroom.
- There is no best programming language, tool, or technology. There are only **better** or **worse** solutions.
- You must be **honest** with your work. If you don't know something, just **ask** me. I will be glad to help you.
- You must be **responsible** with your work. If you don't submit **on time**, please don't complain.
- You must not be **disruptive** or **negatively affect** the **classroom environment**. If you do, I will ask you to **leave** the classroom.

Outline

1. You don't know who I am
2. Course Overview
3. Syllabus
4. Grading & Rules
5. Bibliography

Bibliography

Recommended bibliography:

- ✓ **Software Engineering**, by Ian Sommerville.
- ✗ **Software Engineering at Google**, by Titus Winters, Tom Manshreck, and Hyrum Wright.
- ✗ **The Pragmatic Programmer**, by Andrew Hunt and David Thomas.
- ✗ **Clean Code: A Handbook of Agile Software Craftsmanship**, by Robert C. Martin.
- ✗ **Refactoring: Improving the Design of Existing Code**, by Martin Fowler.
- ✓ **Test-Driven Development: By Example**, by Kent Beck.
- ✗ **Agile Estimating and Planning**, by Mike Cohn.
- ✗ **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment**, by Jez Humble and David Farley.

Bibliography

Recommended bibliography:

- ✓ **Agile Testing: A Practical Guide for Testers and Agile Teams**, by Lisa Crispin and Janet Gregory.
- ✓ **Specification by Example: How Successful Teams Deliver the Right Software**, by Gojko Adzic.
- ✓ **Domain-Driven Design: Tackling Complexity in the Heart of Software**, by Eric Evans.
- ✓ **Patterns of Enterprise Application Architecture**, by Martin Fowler.
- ✓ **Design Patterns: Elements of Reusable Object-Oriented Software**, by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.

Outline

1. You don’t know who I am
2. Course Overview
3. Syllabus
4. Grading & Rules
5. Bibliography

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Thanks!

Questions?

Profile: *www.linkedin.com/in/casierrav*