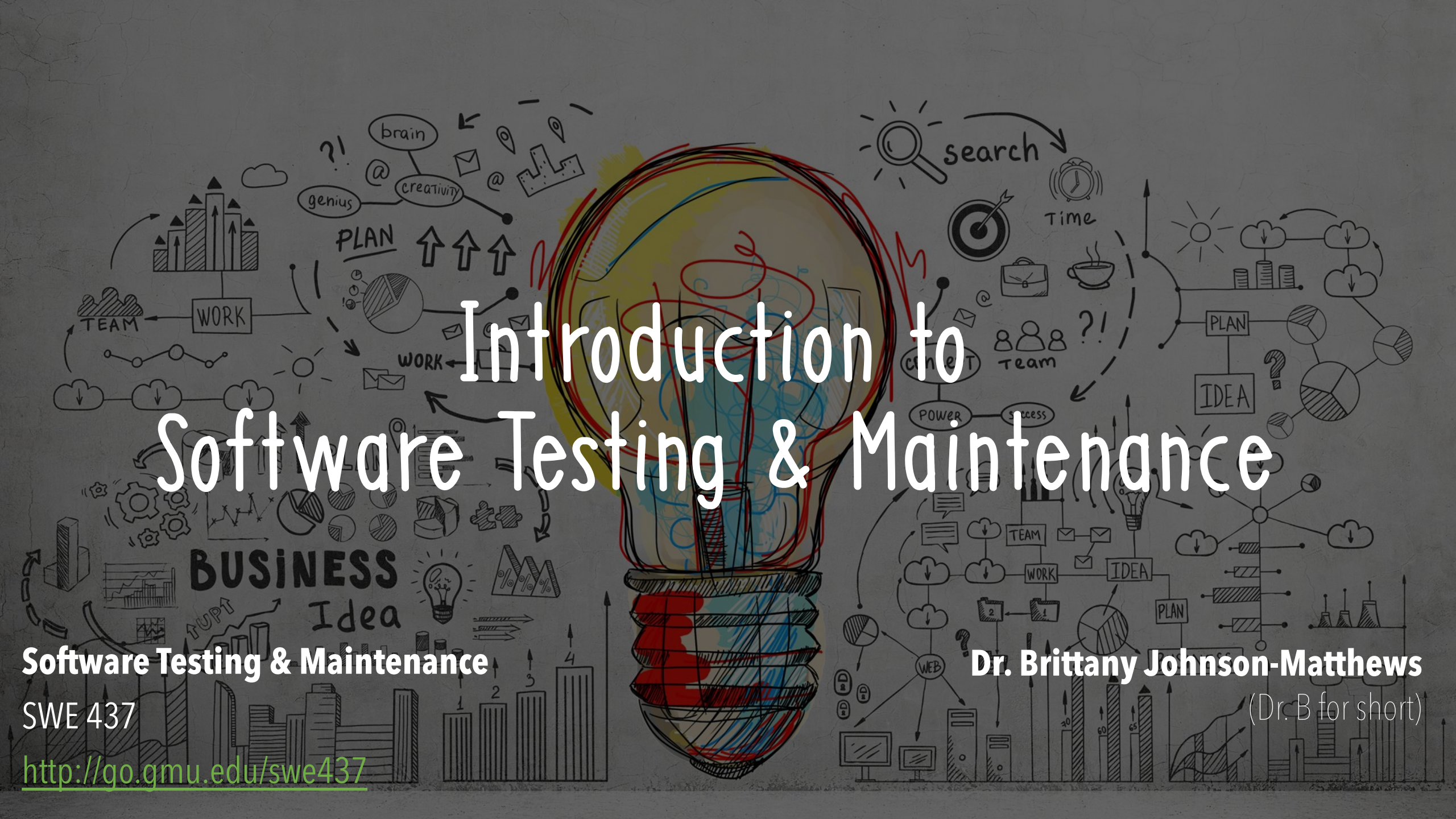


Introduction to Software Testing & Maintenance

Software Testing & Maintenance

SWE 437

http://go.gmu.edu/swe437

Dr. Brittany Johnson-Matthews

(Dr. B for short)

"Traditional" Quality Attributes (1980s)

1. Efficiency of process (time-to-market)
2. Efficiency of execution (performance)

We often teach these as priorities in undergrad computer science classes.

This was true…**in 1985**

Modern Quality Attributes

1. Reliability
2. Usability
3. Security
4. Availability
5. Scalability
6. Maintainability
7. Performance & time to market
8. …

Software projects in the 1960s

In the **1960s** we built **log cabins**…

Built by **single programmer**

Very **little complexity**

No process needed

Simple design (could be kept in short term memory)

Software projects in the 1970s

In the **1970s** we built **bigger houses**…

Still built by **single programmer**
- focus on algorithms & programming

A little **more complex**

Lack of process = **disasters**
Quality didn't affect **bottom line**
But **costs** were starting to **increase**…

Software projects in the 1980s

In the **1980s** we built **office buildings**…

We needed **teamwork** and **communication**

A lot more **complex + data abstraction**

Needed written **requirements** and **design**

Poor process → spectacular **failures**

Missing skills and **knowledge** for successful engineering

Software projects in the 1990s

In the **1990s** we built **skyscrapers**…

Teamwork & communication **not enough**

Needed **new technologies** – languages, modeling, techniques, and processes

Big changes to software development
New languages (Java, UML, etc.) led to **revolutionary procedures**…

But (sadly) education fell **behind**…

Software projects in the 2000s

In the **2000s** we build integrated collections of continuously **evolving cities**…

Primary focus shift from algorithm design and programming

CS education fell so far behind it became **obsolete**

Developers get more practical knowledge from **training courses** than college

Not much **new development**

Pace of change is alarming

In a matter of decades, we've gone from
log cabins → houses → office buildings → skyscrapers → ecosystems

Civil engineers took thousands of years for this kind of change

Electrical engineers took a couple of centuries

So it's not surprising researchers, educators, and engineers can't keep up!

Theory, Practice, & Education

What have you learned in college?

How to build houses

General software engineering courses (SWE/CS 321) introduce a few concepts about buildings

The way we build software has changed dramatically over the years

- **CS curriculum stabilized in 1980s**!

What about…

Maintenance…evolution…re-engineering…maintainability…being "agile"?

What Can You Do?

As a **developer** or **software engineer**…

- Write **clean code**
- **Design** for change
- Follow processes that **make change easy**

As a **professional**…

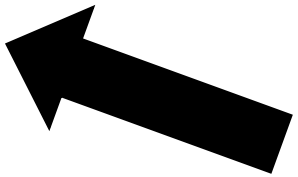
- **Listen** when colleagues teach you new things
- Take **training classes** eagerly
- Further your **education** (MS degree)

Goal of this class

1. Reliability & Testing
2. Usability
3. Security
4. Availability
5. Scalability
6. Maintainability
7. Performance & time to market

Goal of this class

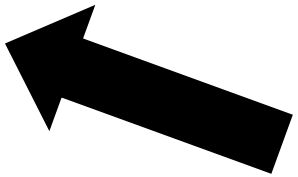
1. **Reliability & Testing**
2. Usability
3. Security
4. Availability
5. Scalability
6. Maintainability
7. Performance & time to market



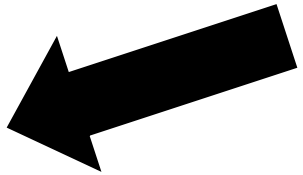
First third of SWE 437

Goal of this class

1. **Reliability & Testing**
2. Usability
3. Security
4. Availability
5. Scalability
6. **Maintainability**
7. Performance & time to market



First third of SWE 437



Last two thirds of SWE 437