Lesson 1

Course Introduction:
Knowledge Has Organizing Power

QUIZ IN PROGRESS
DOOR LOCKED FROM
10:00 to 10:05

Topics: Course Introduction

- Introduction to Software Engineering
- Problem statement for our class project

Introduction to the Course

- Course Approach:
  - Each team project will cover all the steps of software engineering up to system testing.
  - Four people per team.
- See Sakai CS425-2018-01A-01D for lectures, readings, useful links, tests, assignments, etc.
- Grading:
  - 35% Test 1
  - 45% Final Test
  - 15% project deliveries and presentations
  - 5% quizzes, class participation, professional etiquette

Compro Policy for Morning Group Meditations

- Students need to attend 60% of the Dalby Hall morning meditations over the aggregate of their regular computer science courses in order to graduate. Attendance during SCI and Forest Academy courses must be 80% or more.
- Extra Credit For a block (22 days per block)
  - 70% and above: .5% EC (16 days in a standard block)
  - 80% and above: 1% EC (18 days in a standard block)
  - 90% and above: 1.5% EC (20 days in a standard block)

Introduction to the Course

Lab Assignments

- The purpose of all labs and homework is to provide an opportunity for learning with a minimum pressure for graded outcomes.
- For the purpose of learning, all students should attempt their part of the lab individually, but are also encouraged to work in pairs or small groups.
- Discussions among fellow students are very beneficial to the learning process in order to clarify ones own understanding and to remove any road blocks that may slow down progress.

Introduction to the Course

Professional Etiquette

Consider MUM to be your employer, and your professor to be your technical manager. Just like a regular job, the inability to show respect can get you thrown out; regardless of work performance. Failure to attend class, to be punctual, or to dress professionally can result in reduced grade, dismissal from class, or outright failure.

Courtesy during group meditation is a part of professional etiquette. No typing, texting, reading, or talking during meditation. We take 2 minutes of silence after meditation before starting activity. Everyone is expected to allow their neighbors this 2 minutes of silence.

Project Development and Demos

- Each group will share their design and implementation of their project with several demos.
- We can all learn from the demos.
- Most of us learn:
  - 10% of what we read
  - 20% of what we hear
  - 30% of what we see
  - 50% of what we see and hear
  - 70% of what we talk over with others
  - 80% of what we use and do in a job
  - 90% of what we teach someone else
- Placement reports that presentations are great interview preparation.

Final Presentation

Final Presentation - 15 minutes per group member

You will demo your code and show how it follows your class design. I will ask you to walk me through your source code, and I will evaluate that your source matches your design. Evaluation includes:

Does your code match your final design?

Can you show your code flow clearly?

Can you answer questions on how you implemented your use cases?

Can you show at least one unit test?

If there are parts of your use case not implemented can you describe the implementation?

Think of it as a job interview and be prepared to answer questions like the following:

your development experience – what you learned

design choices that you made

parts that were easy

parts that were complex

development process used – iterations, design/architecture/coding approach

f team experience – what worked well, what could improve

Brief history of software engineering

- Early days: small toy programs
- As programming languages and processing speed improved, more useful applications were developed
- New problems arose in creating still larger applications:
  - Need for effective way to communicate requirements
  - Need for effective way to formulate and communicate design ideas; to record what has been done so far and why

Evolution of Solutions

- Early solutions
  - More emphasis on documentation and good documentation styles
  - Tools were developed to help visualize software design (eventually, UML)
  - A new design paradigm emerged – the “Object-Oriented” paradigm, which greatly facilitated the process of building larger scale applications with less cost and fewer failures

Software Development Processes and RUP

- Emergence of Software Development Processes
- As developers learned good OO techniques and documentation practices, the problem remained: How to put all of the techniques together to produce good software?
  - How long do we spend on initial requirements?
  - To what extent do requirements need to be fully understood before beginning design? Coding?
  - How do we formulate a plan for completing a project when most of the details about the project are unknown?
  - How do we make sure that the stakeholders for the project are satisfied with the result?

Software Development Processes and RUP

- Software development processes address these concerns
- RUP = Rational Unified Process has become the industry standard for providing methods of development that provide reliable answers to these questions
- Agile Modeling, and agile development methods including SCRUM and eXtreme programming are widely used.
- SCRUM and eXtreme are compatible with RUP.
- Goal –> optimize and choose the best software development method based on project size, staff expertise, project goals, and other project specific characteristics.

Fundamental Goals of Software Engineering

Examining the aims of development methodologies, we see several fundamental goals:

- A clear vision about a project that can be communicated to others ("see the tree in the seed").
- Ability to respond positively to change
- Ability to minimize mistakes
- Ability to fulfill a large-scale intention (get the right result on time and within budget)

These aims have as much to do with the quality of the developer's awareness as they do about development processes themselves

The Role of TM in Software Engineering

- Hundreds of studies and the experience of thousands of practitioners – including software developers – verify that the qualities needed for successful software development are developed and enhanced by TM
  - Clear thinking
  - Ability to adapt to change
  - Ability to achieve goals
  - Ability to make fewer mistakes

The Role of TM in Software Engineering

- *SCI Principles:*
  - *Subtler Levels of the Mind Are More Powerful.*
  - *Water the root to enjoy the fruit.*

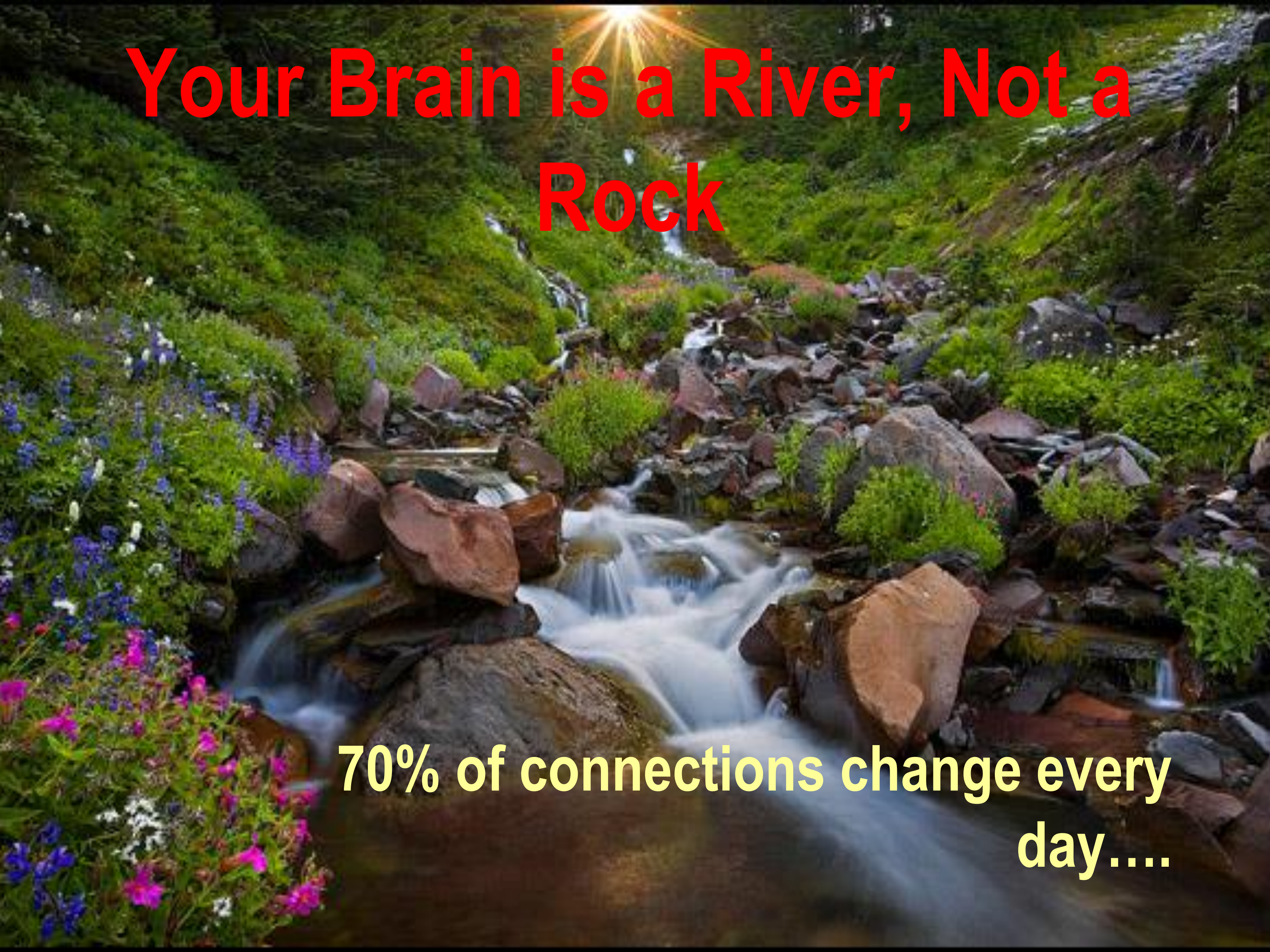
1. Frequently during TM® or immediately after TM® solutions to problems will spontaneously arise.
2. The mind has had a chance to relax and transcend the level of the problem.

The Role of TM in Software Engineering

- *SCI Principle:*

- *Release of stress and contact with Pure Consciousness supports spontaneous right action.*

1. Learn to recognize when you and your team members are being stressed.
2. Team dynamics will start to deteriorate under stress. Stress causes spontaneous **wrong** action.
3. Use TM® as a way to avoid and release stress.
4. One person with a balanced and calm perspective can improve a whole team’s productivity.



Your Brain is a River, Not a Rock

70% of connections change every day….

Size and activity of our brain

- From - _Your Brain is a River, Not a Rock_ – Dr. Fred Travis
- We start with 100 Billion neurons
- Each neuron can connect to over 1000 other neurons
- Total connections – 100 Trillion
- For first 3 years we add 24 Million new connections ***every minute!***
- Through training/studying we can optimize speed and reliability of our connections via myelination
- We have 100,000 miles of myelinated neurons/axons

Transcending thought is infinitely more valuable than thinking.
--Maharishi

Course Project

- MUMSched – our own tool that will build a Compro schedule and allow students to register for classes.
- MUMSched will take input on the classes we want to offer, the number of students for each entry, and the number of faculty and their expertise. It will then generate a block-by-block schedule for a semester or academic year.

Project

- We will use various technologies to help in developing the project . These technologies are examples of the type of tools used in the industry. Teams may choose to use other comparable tools.
  - StarUML, Visual Paradigm, Visual Studio UML, Violet
  - **RUP**
  - **Scrum**
  - MySql, Java DB
  - Java EE 7 and NetBeans
  - Spring/Hibernate and Eclipse
  - **GitHub** for team development and managing code
  - JUnit for testing

Class Conventions

- The first two weeks are devoted to learning the RUP process, completing RUP steps, and doing proof of concept coding for our project in your chosen enterprise architecture.
- Groups divide up the work to complete the class project
- Once coding starts, use daily Scrums to assign and track work
- Modifying design and requirements after coding starts is ok
- Group work for all labs and assignments is ok
- Each group does at least one early demo of their project
- Be sure you are ready to do the tests, and your final project presentation on your own.