Introduction: Course Prerequisites & Learning Strategies

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

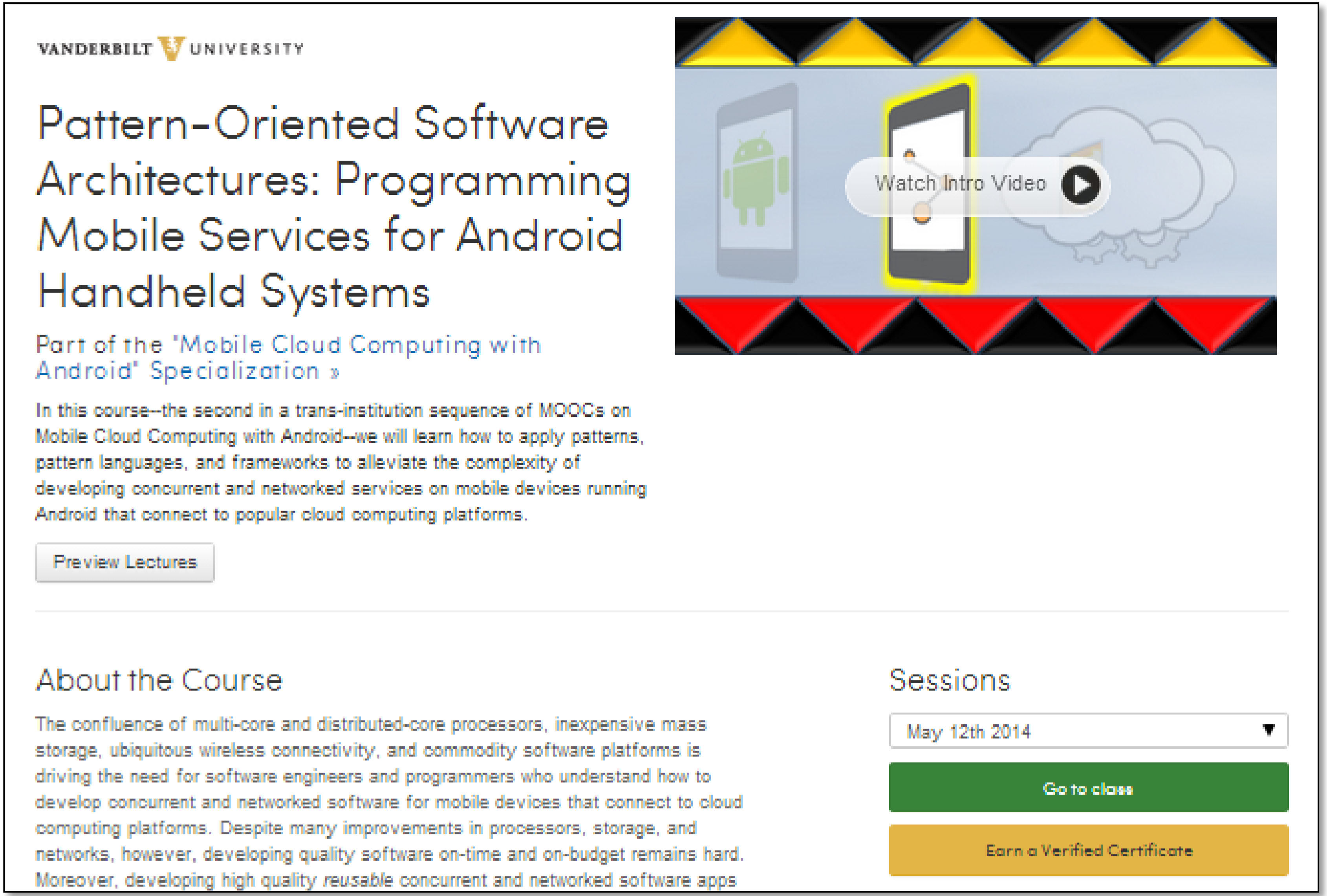
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand the course prerequisites & how to complete it successfully



VANDERBILT UNIVERSITY

Pattern-Oriented Software Architectures: Programming Mobile Services for Android Handheld Systems

Part of the "Mobile Cloud Computing with Android" Specialization »

In this course--the second in a trans-institution sequence of MOOCs on Mobile Cloud Computing with Android--we will learn how to apply patterns, pattern languages, and frameworks to alleviate the complexity of developing concurrent and networked services on mobile devices running Android that connect to popular cloud computing platforms.

About the Course

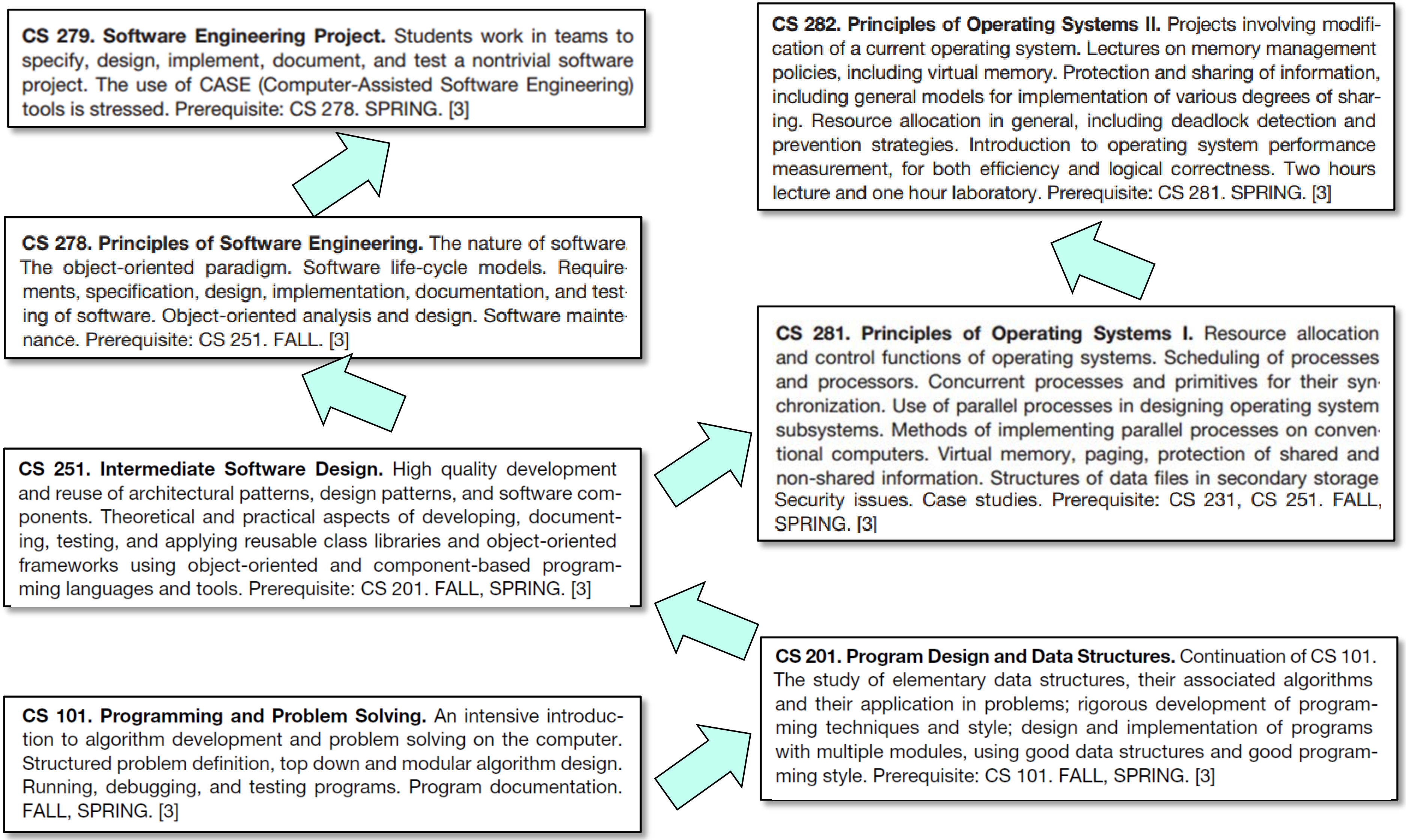
The confluence of multi-core and distributed-core processors, inexpensive mass storage, ubiquitous wireless connectivity, and commodity software platforms is driving the need for software engineers and programmers who understand how to develop concurrent and networked software for mobile devices that connect to cloud computing platforms. Despite many improvements in processors, storage, and networks, however, developing quality software on-time and on-budget remains hard. Moreover, developing high quality *reusable* concurrent and networked software apps

Sessions

May 12th 2014 ▼

Course “Prerequisites”

- We know students at our universities have taken the prerequisites



CS 279. Software Engineering Project. Students work in teams to specify, design, implement, document, and test a nontrivial software project. The use of CASE (Computer-Assisted Software Engineering) tools is stressed. Prerequisite: CS 278. SPRING. [3]

CS 278. Principles of Software Engineering. The nature of software. The object-oriented paradigm. Software life-cycle models. Requirements, specification, design, implementation, documentation, and testing of software. Object-oriented analysis and design. Software maintenance. Prerequisite: CS 251. FALL. [3]

CS 251. Intermediate Software Design. High quality development and reuse of architectural patterns, design patterns, and software components. Theoretical and practical aspects of developing, documenting, testing, and applying reusable class libraries and object-oriented frameworks using object-oriented and component-based programming languages and tools. Prerequisite: CS 201. FALL, SPRING. [3]

CS 101. Programming and Problem Solving. An intensive introduction to algorithm development and problem solving on the computer. Structured problem definition, top down and modular algorithm design. Running, debugging, and testing programs. Program documentation. FALL, SPRING. [3]

CS 282. Principles of Operating Systems II. Projects involving modification of a current operating system. Lectures on memory management policies, including virtual memory. Protection and sharing of information, including general models for implementation of various degrees of sharing. Resource allocation in general, including deadlock detection and prevention strategies. Introduction to operating system performance measurement, for both efficiency and logical correctness. Two hours lecture and one hour laboratory. Prerequisite: CS 281. SPRING. [3]

CS 281. Principles of Operating Systems I. Resource allocation and control functions of operating systems. Scheduling of processes and processors. Concurrent processes and primitives for their synchronization. Use of parallel processes in designing operating system subsystems. Methods of implementing parallel processes on conventional computers. Virtual memory, paging, protection of shared and non-shared information. Structures of data files in secondary storage. Security issues. Case studies. Prerequisite: CS 231, CS 251. FALL, SPRING. [3]

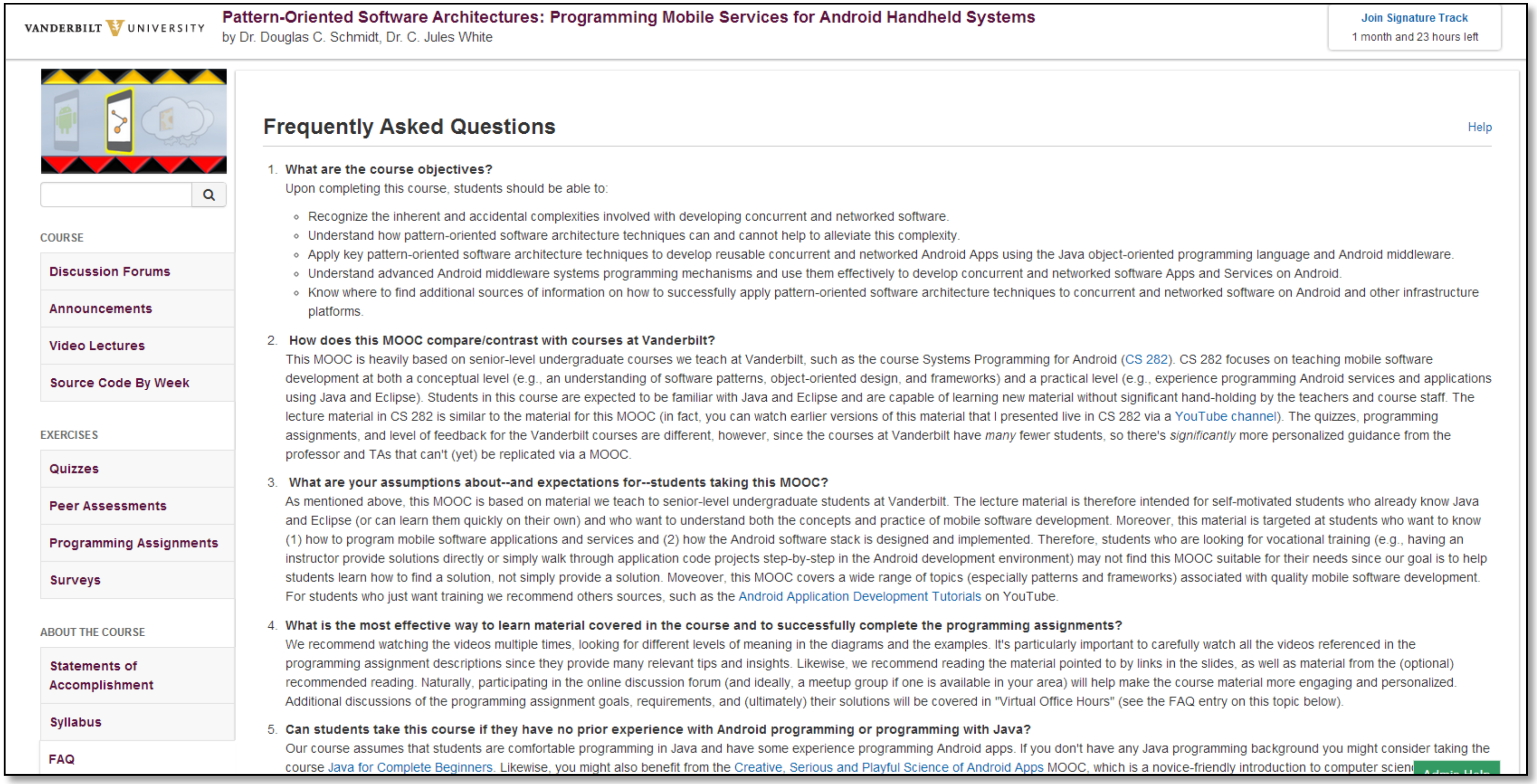
CS 201. Program Design and Data Structures. Continuation of CS 101. The study of elementary data structures, their associated algorithms and their application in problems; rigorous development of programming techniques and style; design and implementation of programs with multiple modules, using good data structures and good programming style. Prerequisite: CS 101. FALL, SPRING. [3]

Course “Prerequisites”

- We know students at our universities have taken the prerequisites
- We don’t know what you know or whether you’re prepared or not!

Course “Prerequisites”

- We know students at our universities have taken the prerequisites
- We don’t know what you know or whether you’re prepared or not!
- This course has assumptions about—& expectations of—the students



Pattern-Oriented Software Architectures: Programming Mobile Services for Android Handheld Systems
by Dr. Douglas C. Schmidt, Dr. C. Jules White

Frequently Asked Questions

1. **What are the course objectives?**
   Upon completing this course, students should be able to:
   - Recognize the inherent and accidental complexities involved with developing concurrent and networked software.
   - Understand how pattern-oriented software architecture techniques can and cannot help to alleviate this complexity.
   - Apply key pattern-oriented software architecture techniques to develop reusable concurrent and networked Android Apps using the Java object-oriented programming language and Android middleware.
   - Understand advanced Android middleware systems programming mechanisms and use them effectively to develop concurrent and networked software Apps and Services on Android.
   - Know where to find additional sources of information on how to successfully apply pattern-oriented software architecture techniques to concurrent and networked software on Android and other infrastructure platforms.
2. **How does this MOOC compare/contrast with courses at Vanderbilt?**
   This MOOC is heavily based on senior-level undergraduate courses we teach at Vanderbilt, such as the course Systems Programming for Android (CS 282). CS 282 focuses on teaching mobile software development at both a conceptual level (e.g., an understanding of software patterns, object-oriented design, and frameworks) and a practical level (e.g., experience programming Android services and applications using Java and Eclipse). Students in this course are expected to be familiar with Java and Eclipse and are capable of learning new material without significant hand-holding by the teachers and course staff. The lecture material in CS 282 is similar to the material for this MOOC (in fact, you can watch earlier versions of this material that I presented live in CS 282 via a YouTube channel). The quizzes, programming assignments, and level of feedback for the Vanderbilt courses are different, however, since the courses at Vanderbilt have *many* fewer students, so there's *significantly* more personalized guidance from the professor and TAs that can't (yet) be replicated via a MOOC.
3. **What are your assumptions about--and expectations for--students taking this MOOC?**
   As mentioned above, this MOOC is based on material we teach to senior-level undergraduate students at Vanderbilt. The lecture material is therefore intended for self-motivated students who already know Java and Eclipse (or can learn them quickly on their own) and who want to understand both the concepts and practice of mobile software development. Moreover, this material is targeted at students who want to know (1) how to program mobile software applications and services and (2) how the Android software stack is designed and implemented. Therefore, students who are looking for vocational training (e.g., having an instructor provide solutions directly or simply walk through application code projects step-by-step in the Android development environment) may not find this MOOC suitable for their needs since our goal is to help students learn how to find a solution, not simply provide a solution. Moveover, this MOOC covers a wide range of topics (especially patterns and frameworks) associated with quality mobile software development. For students who just want training we recommend others sources, such as the Android Application Development Tutorials on YouTube.
4. **What is the most effective way to learn material covered in the course and to successfully complete the programming assignments?**
   We recommend watching the videos multiple times, looking for different levels of meaning in the diagrams and the examples. It's particularly important to carefully watch all the videos referenced in the programming assignment descriptions since they provide many relevant tips and insights. Likewise, we recommend reading the material pointed to by links in the slides, as well as material from the (optional) recommended reading. Naturally, participating in the online discussion forum (and ideally, a meetup group if one is available in your area) will help make the course material more engaging and personalized. Additional discussions of the programming assignment goals, requirements, and (ultimately) their solutions will be covered in "Virtual Office Hours" (see the FAQ entry on this topic below).
5. **Can students take this course if they have no prior experience with Android programming or programming with Java?**
   Our course assumes that students are comfortable programming in Java and have some experience programming Android apps. If you don't have any Java programming background you might consider taking the course Java for Complete Beginners. Likewise, you might also benefit from the Creative, Serious and Playful Science of Android Apps MOOC, which is a novice-friendly introduction to computer scien

See item #3 at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

What We'd Like Students to Know

- Ideally, students know certain things

What We’d Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**

```
public class EventHandler
    extends Observer {
  public void update(Observable o,
                     Object arg)
  { /*…*/ }
    …

public class EventSource
    extends Observable,
    implements Runnable {
  public void run()
  { /*…*/ notifyObservers(/*…*/); }
    …

EventSource source =
       new EventSource();
EventHandler handler =
       new EventHandler();
eventSource.addObserver(handler);
Thread thread =
   new Thread(eventSource);
thread.start();
…
```

en.wikipedia.org/wiki/Java_(programming_language) has more info on Java

What We’d Like Students to Know

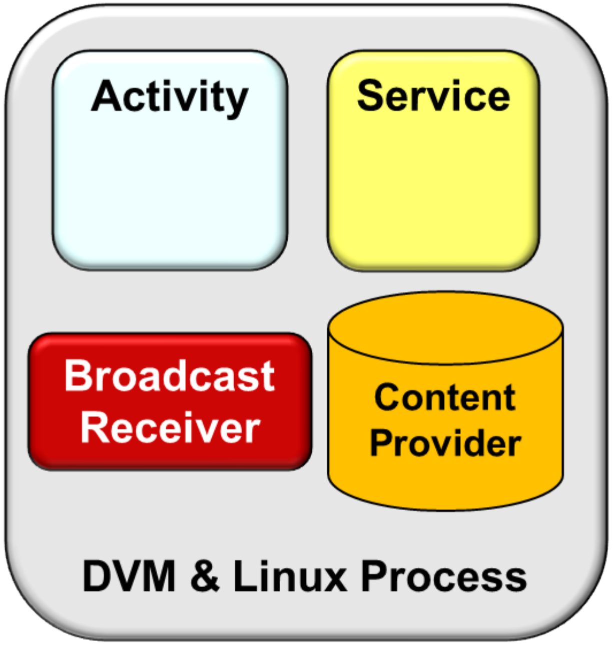
- Ideally, students know certain things
  - **OO programming languages**
    - e.g., Java classes, inheritance, dynamic binding, & generics, etc.

```
public class EventHandler
    extends Observer {
  public void update(Observable o,
                     Object arg)
  { /*…*/ }
    …

public class EventSource
    extends Observable,
    implements Runnable {
  public void run()
  { /*…*/ notifyObservers(/*…*/); }
    …

EventSource source =
       new EventSource();
EventHandler handler =
       new EventHandler();
eventSource.addObserver(handler);
Thread thread =
   new Thread(eventSource);
thread.start();
…
```

en.wikipedia.org/wiki/Java_(programming_language) has more info on Java

What We'd Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**
  - **Android development**
    - e.g., Activities, Intents, UI components, & Eclipse ADT

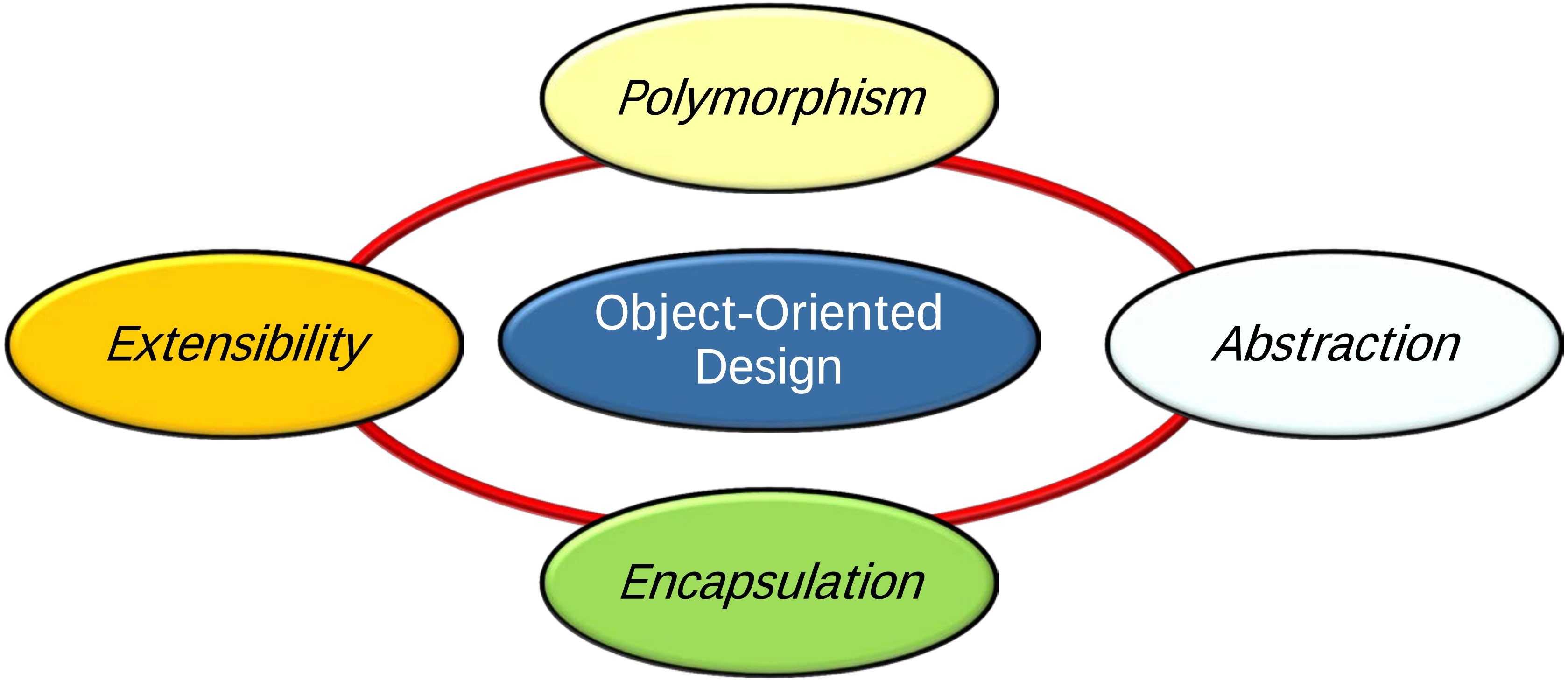


See item #5 at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

Other Useful Things for Students to Know

What We'd Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**
  - **Android development**
  - **OO design concepts & notations**



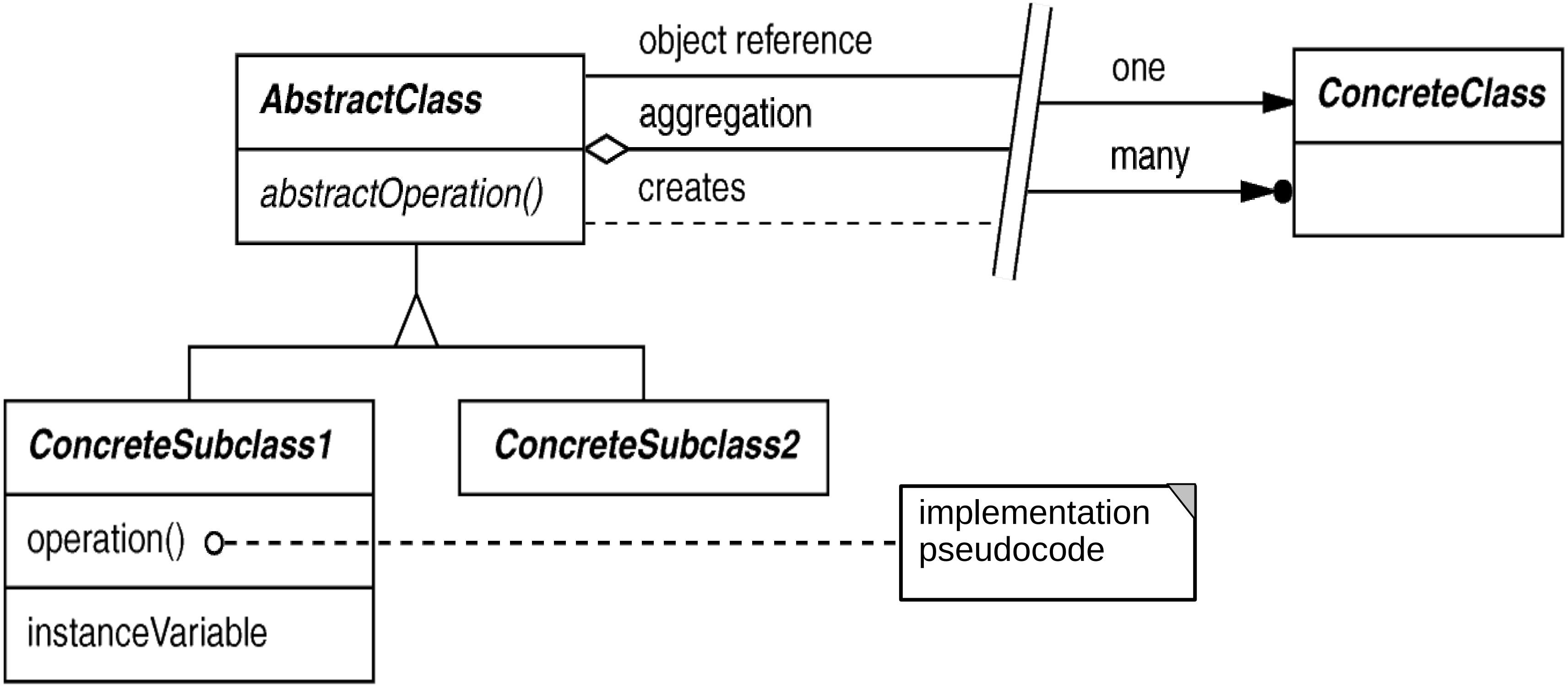
What We’d Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**
  - **Android development**
  - **OO design concepts & notations**

en.wikipedia.org/wiki/Object-oriented_design has more info on OO design

What We’d Like Students to Know

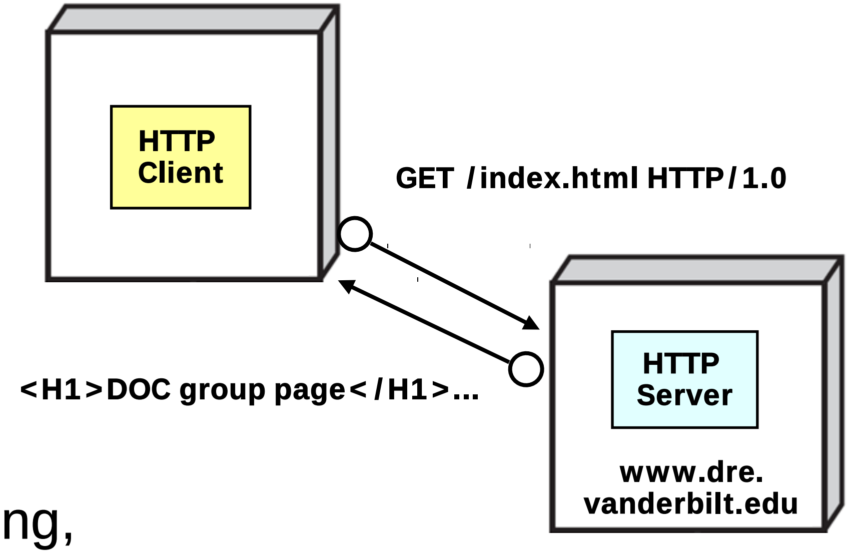
- Ideally, students know certain things
  - OO programming languages
  - Android development
  - **OO design concepts & notations**

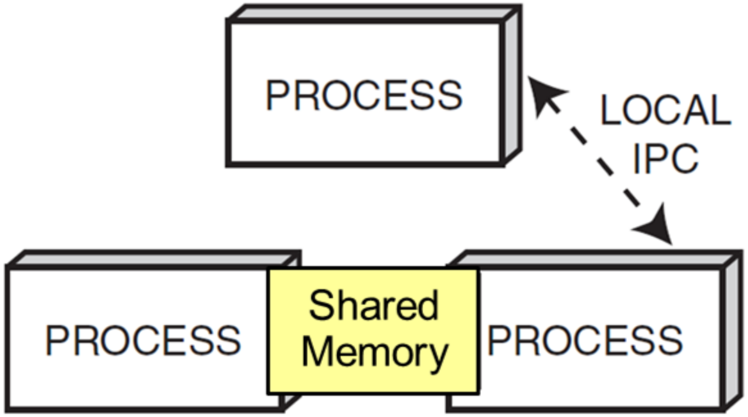


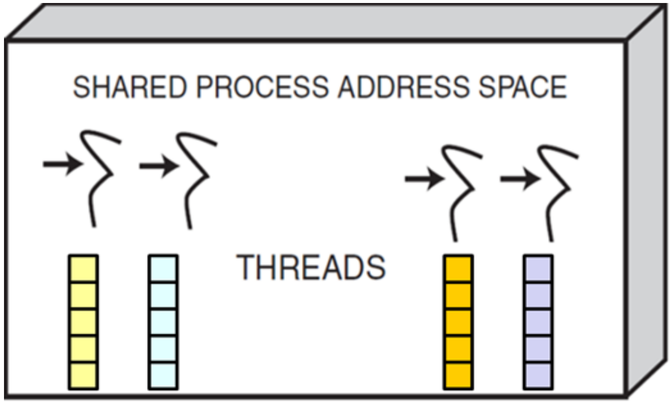
en.wikipedia.org/wiki/Unified_Modeling_Language

What We'd Like Students to Know

- Ideally, students know certain things
  - OO programming languages
  - Android development
  - OO design concepts & notations
  - **Systems & network programming concepts**
    - e.g., event loops, multi-processing & -threading, synchronization, scheduling, & inter-process communication







See Section 1 videos at class.coursera.org/posa-001/lecture for an overview

Strategies for Learning this Material (Part 1)

Strategies for Learning All this Material

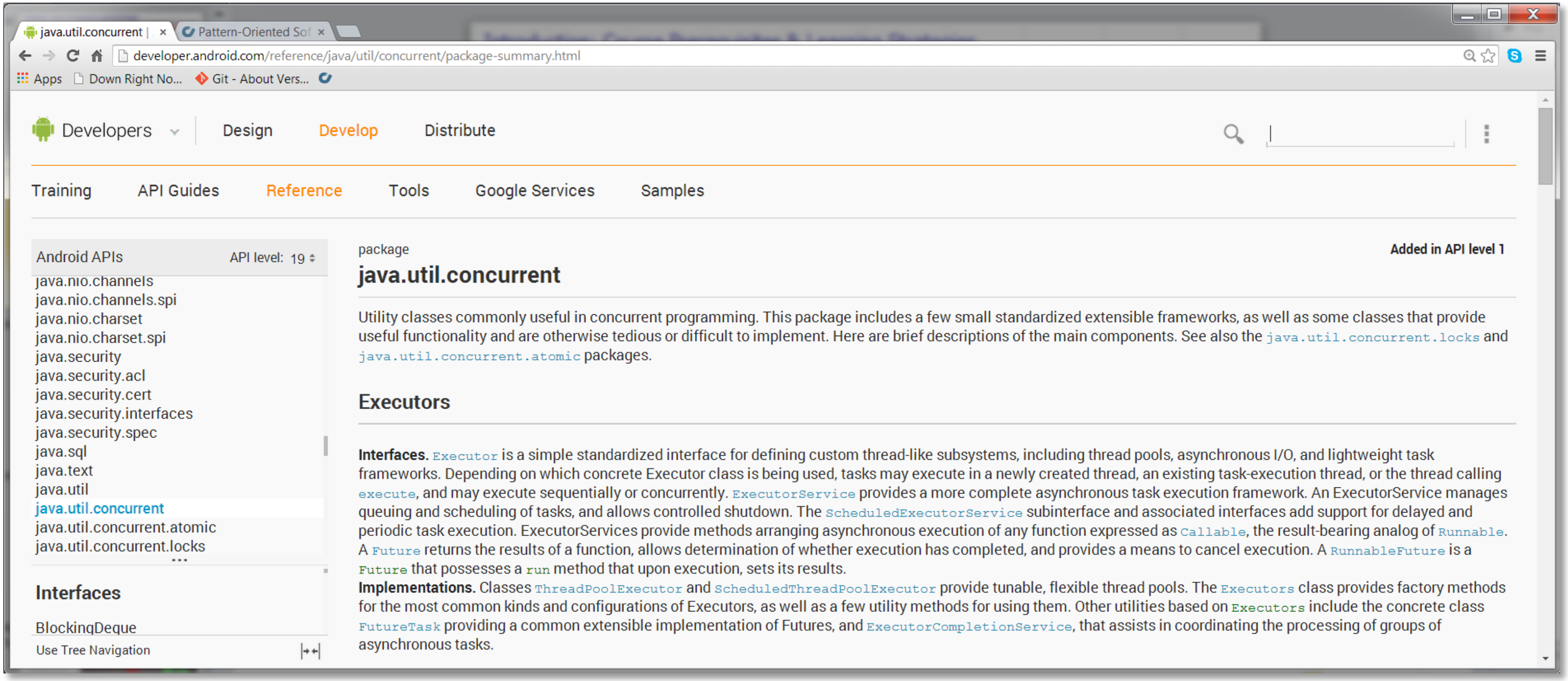
- This MOOC covers a lot of material, so we encourage you to do the following

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times (at multiple speeds!)

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times (at multiple speeds!)
  - Read online resources



package

Added in API level 1

java.util.concurrent

Utility classes commonly useful in concurrent programming. This package includes a few small standardized extensible frameworks, as well as some classes that provide useful functionality and are otherwise tedious or difficult to implement. Here are brief descriptions of the main components. See also the `java.util.concurrent.locks` and `java.util.concurrent.atomic` packages.

Executors

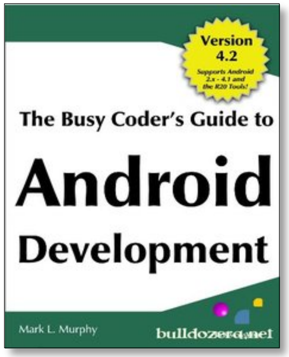
Interfaces. `Executor` is a simple standardized interface for defining custom thread-like subsystems, including thread pools, asynchronous I/O, and lightweight task frameworks. Depending on which concrete Executor class is being used, tasks may execute in a newly created thread, an existing task-execution thread, or the thread calling `execute`, and may execute sequentially or concurrently. `ExecutorService` provides a more complete asynchronous task execution framework. An ExecutorService manages queuing and scheduling of tasks, and allows controlled shutdown. The `ScheduledExecutorService` subinterface and associated interfaces add support for delayed and periodic task execution. ExecutorServices provide methods arranging asynchronous execution of any function expressed as `Callable`, the result-bearing analog of `Runnable`. A `Future` returns the results of a function, allows determination of whether execution has completed, and provides a means to cancel execution. A `RunnableFuture` is a `Future` that possesses a `run` method that upon execution, sets its results.
Implementations. Classes `ThreadPoolExecutor` and `ScheduledThreadPoolExecutor` provide tunable, flexible thread pools. The `Executors` class provides factory methods for the most common kinds and configurations of Executors, as well as a few utility methods for using them. Other utilities based on `Executors` include the concrete class `FutureTask` providing a common extensible implementation of Futures, and `ExecutorCompletionService`, that assists in coordinating the processing of groups of asynchronous tasks.

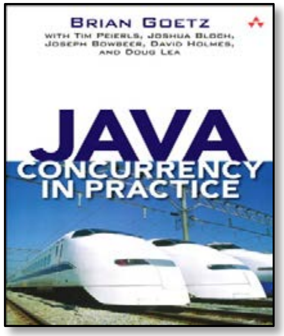
See item #4 at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

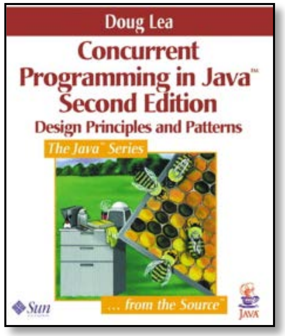
Strategies for Learning this Material (Part 2)

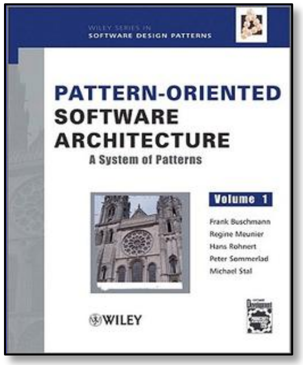
Strategies for Learning All this Material

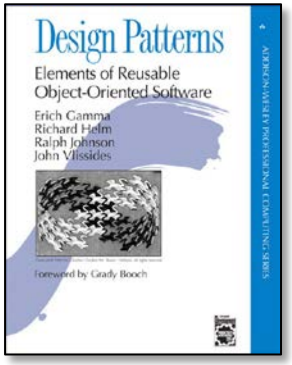
- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times (at multiple speeds!)
  - Read online resources
  - Read suggested books

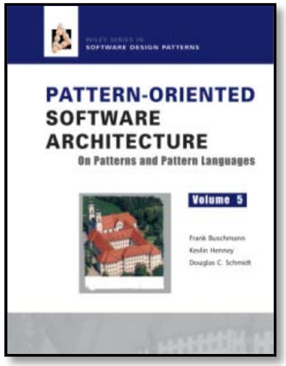

Version 4.2
Supports Android 2.x - 4.1 and the R20 Tools!
The Busy Coder's Guide to
Android
Development
Mark L. Murphy
bulldozer.net

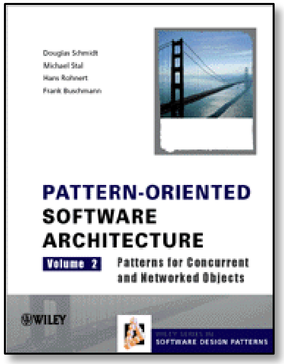

Brian Goetz
with Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea
JAVA
CONCURRENCY
IN PRACTICE

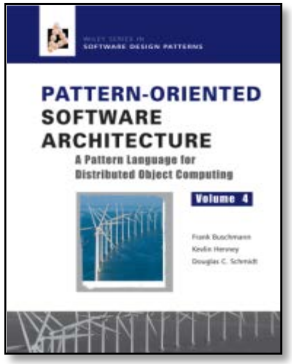








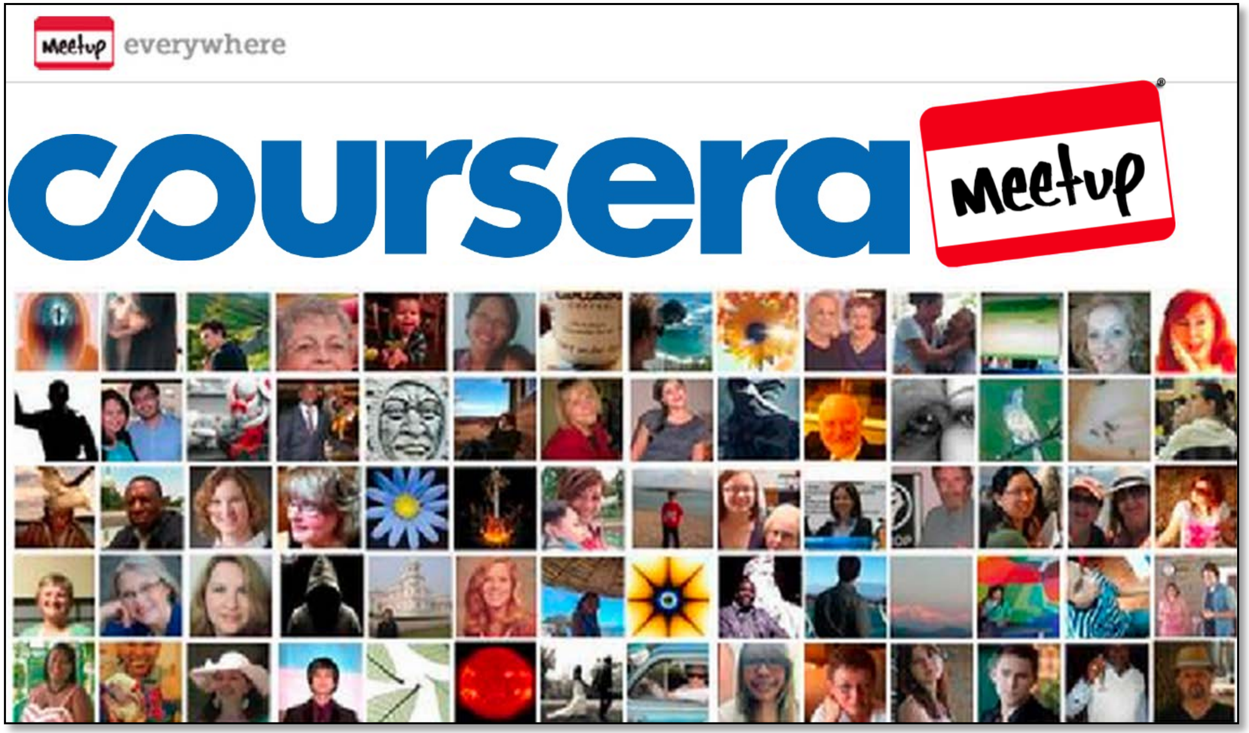



WILEY SERIES IN SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE
A Pattern Language for Distributed Object Computing
Volume 4
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

See "Suggested Reading" at www.coursera.org/course/posa

Strategies for Learning All this Material

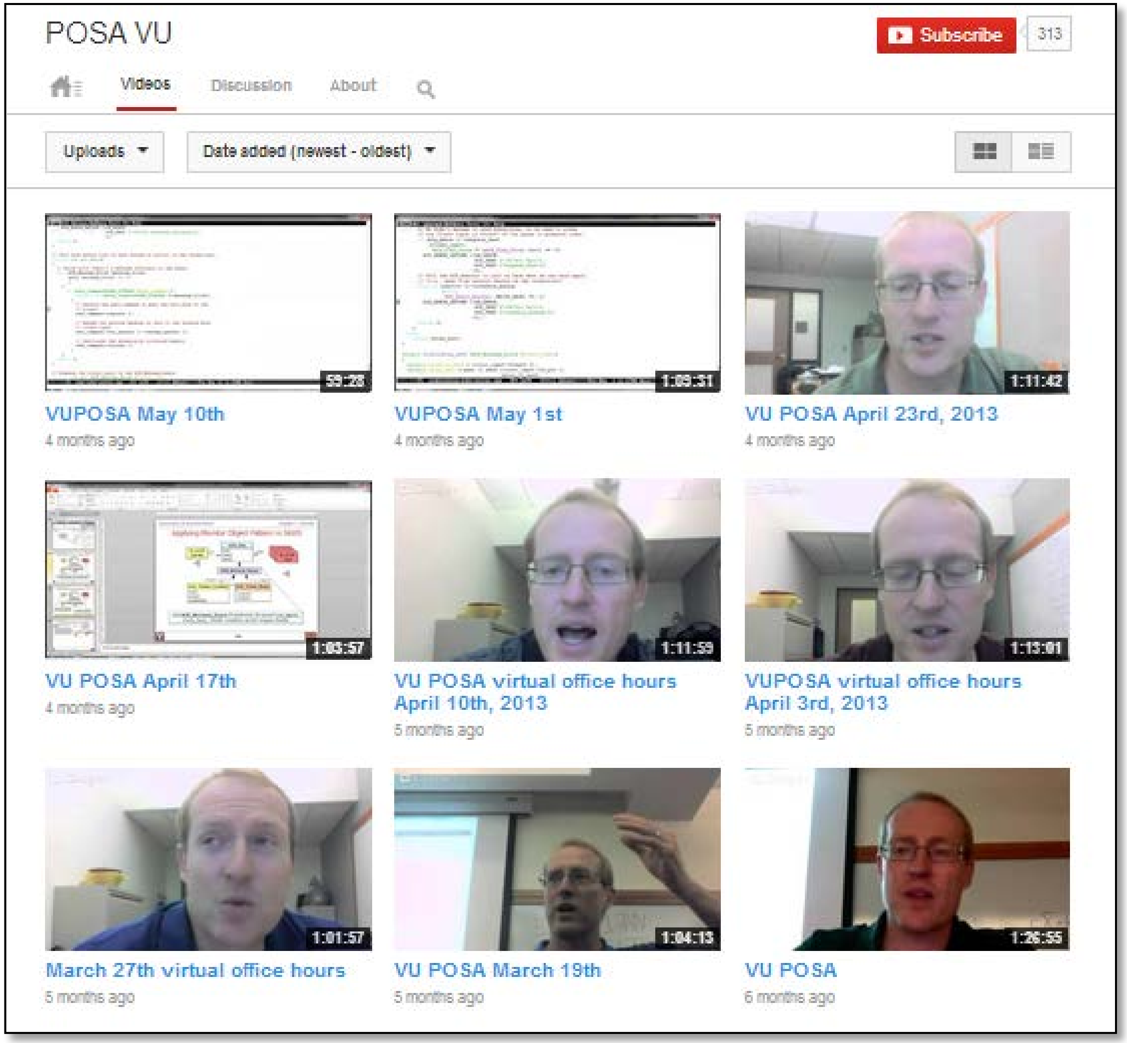
- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times (at multiple speeds!)
  - Read online resources
  - Read suggested books
  - Join a "meetup group"



www.meetup.com/Coursera has more information on meetup groups

Strategies for Learning All this Material

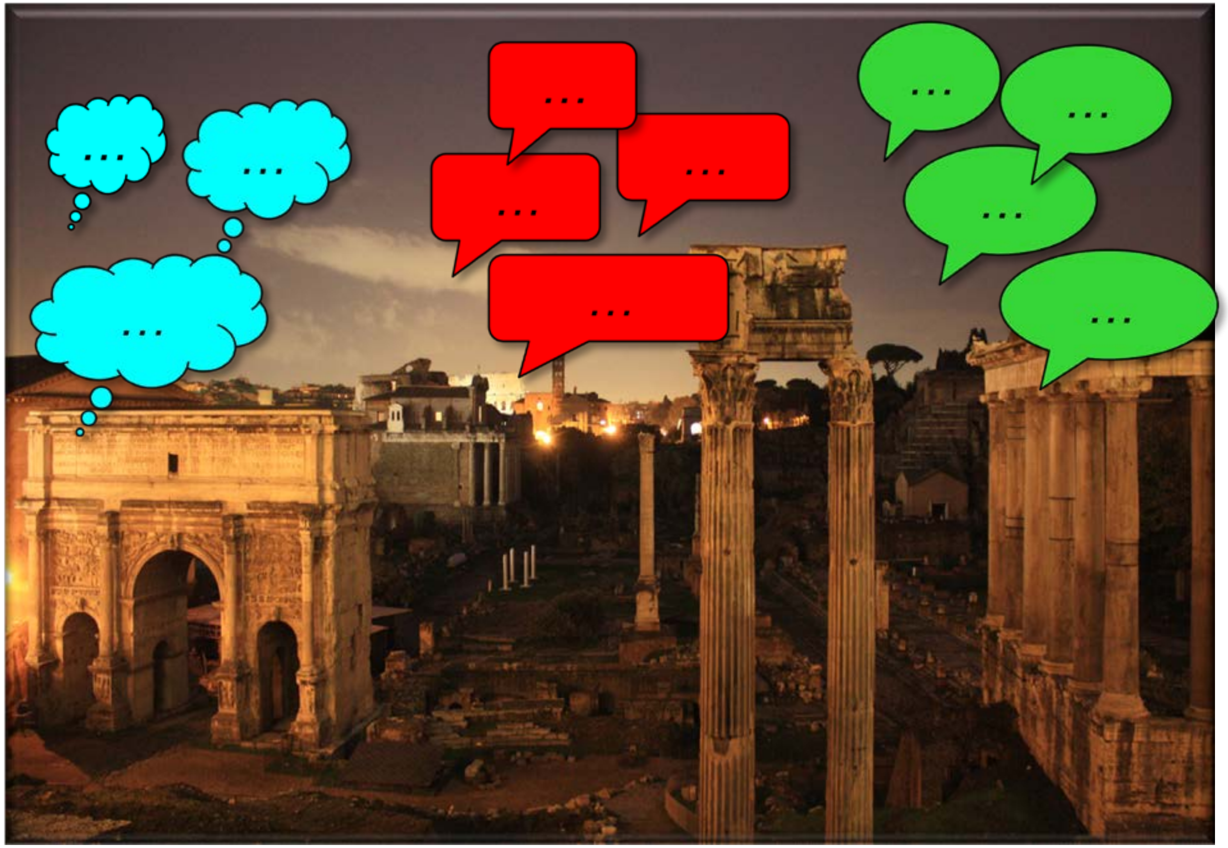
- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times (at multiple speeds!)
  - Read online resources
  - Read suggested books
  - Join a "meetup group"
  - Attend "virtual office hours"



See item #38 at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times (at multiple speeds!)
  - Read online resources
  - Read suggested books
  - Join a “meetup group”
  - Attend “virtual office hours”
  - Participate in the course online discussion forum

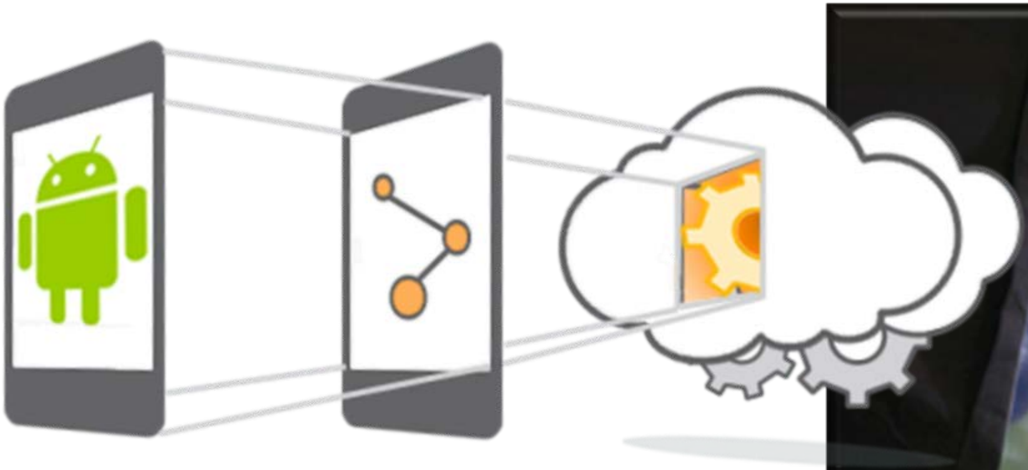


See item #7 at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

Summary

Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android



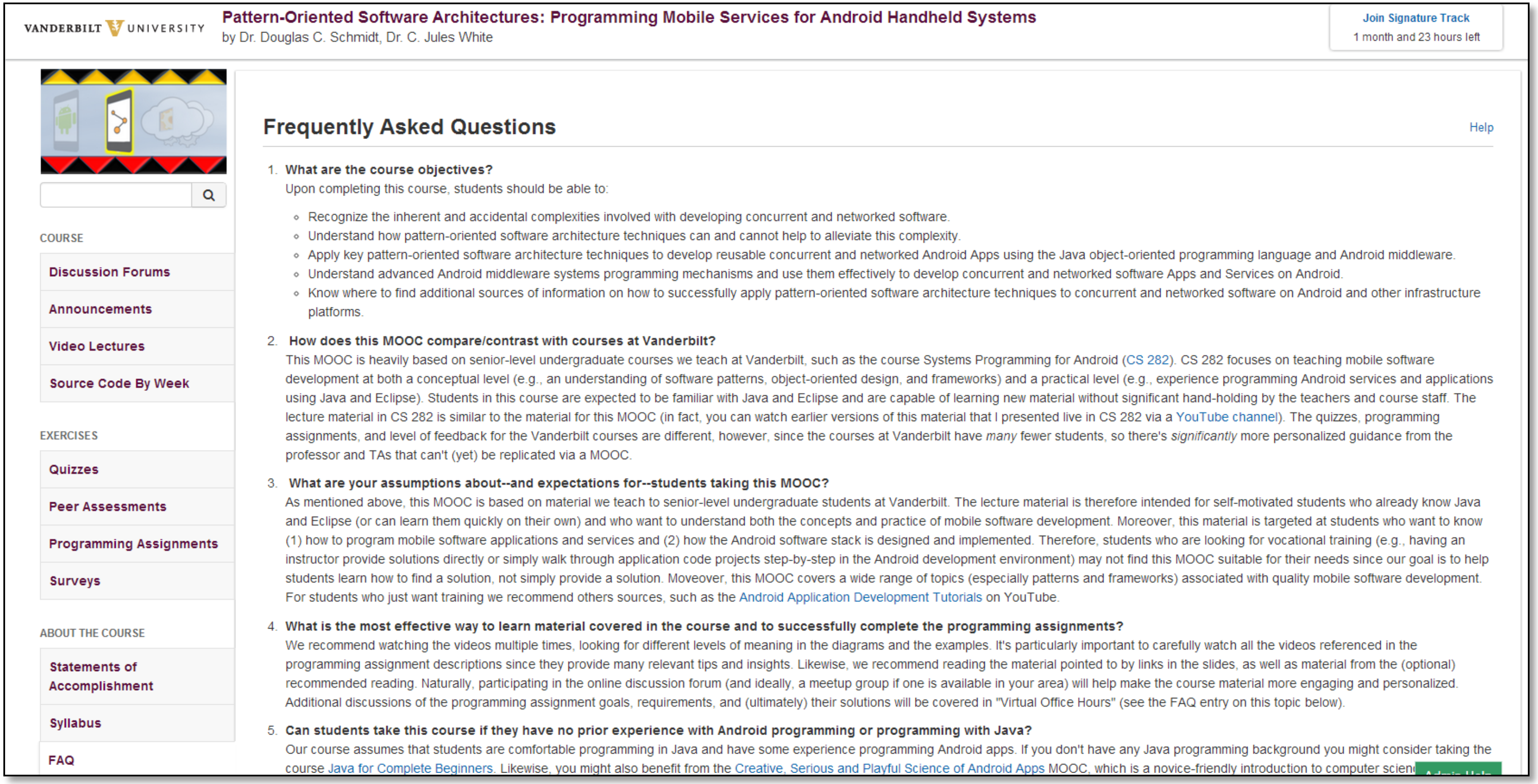
Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android

See the FAQ at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android



Pattern-Oriented Software Architectures: Programming Mobile Services for Android Handheld Systems
by Dr. Douglas C. Schmidt, Dr. C. Jules White

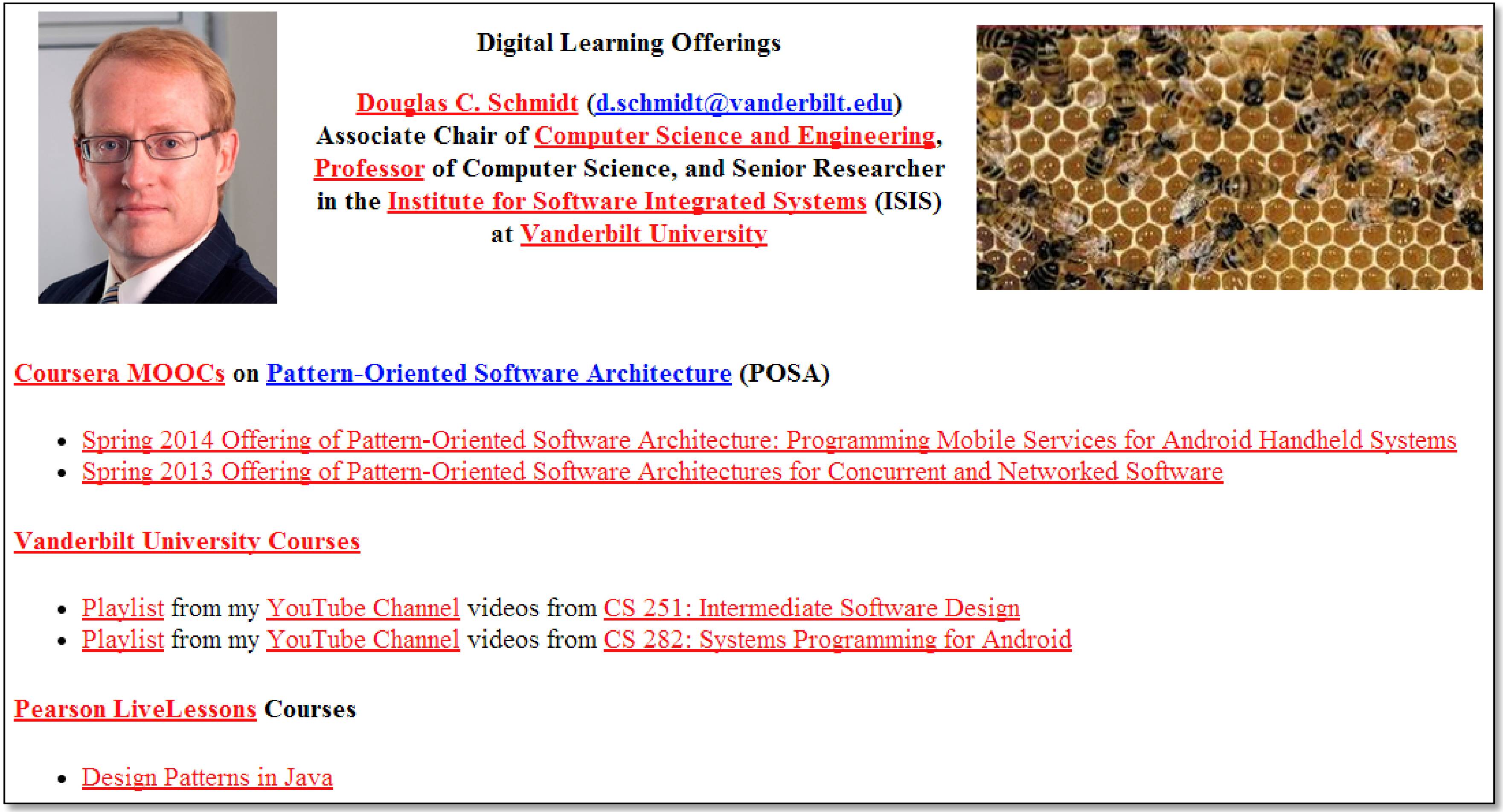
Frequently Asked Questions

1. **What are the course objectives?**
   Upon completing this course, students should be able to:
   - Recognize the inherent and accidental complexities involved with developing concurrent and networked software.
   - Understand how pattern-oriented software architecture techniques can and cannot help to alleviate this complexity.
   - Apply key pattern-oriented software architecture techniques to develop reusable concurrent and networked Android Apps using the Java object-oriented programming language and Android middleware.
   - Understand advanced Android middleware systems programming mechanisms and use them effectively to develop concurrent and networked software Apps and Services on Android.
   - Know where to find additional sources of information on how to successfully apply pattern-oriented software architecture techniques to concurrent and networked software on Android and other infrastructure platforms.
2. **How does this MOOC compare/contrast with courses at Vanderbilt?**
   This MOOC is heavily based on senior-level undergraduate courses we teach at Vanderbilt, such as the course Systems Programming for Android (CS 282). CS 282 focuses on teaching mobile software development at both a conceptual level (e.g., an understanding of software patterns, object-oriented design, and frameworks) and a practical level (e.g., experience programming Android services and applications using Java and Eclipse). Students in this course are expected to be familiar with Java and Eclipse and are capable of learning new material without significant hand-holding by the teachers and course staff. The lecture material in CS 282 is similar to the material for this MOOC (in fact, you can watch earlier versions of this material that I presented live in CS 282 via a YouTube channel). The quizzes, programming assignments, and level of feedback for the Vanderbilt courses are different, however, since the courses at Vanderbilt have *many* fewer students, so there's *significantly* more personalized guidance from the professor and TAs that can't (yet) be replicated via a MOOC.
3. **What are your assumptions about--and expectations for--students taking this MOOC?**
   As mentioned above, this MOOC is based on material we teach to senior-level undergraduate students at Vanderbilt. The lecture material is therefore intended for self-motivated students who already know Java and Eclipse (or can learn them quickly on their own) and who want to understand both the concepts and practice of mobile software development. Moreover, this material is targeted at students who want to know (1) how to program mobile software applications and services and (2) how the Android software stack is designed and implemented. Therefore, students who are looking for vocational training (e.g., having an instructor provide solutions directly or simply walk through application code projects step-by-step in the Android development environment) may not find this MOOC suitable for their needs since our goal is to help students learn how to find a solution, not simply provide a solution. Moveover, this MOOC covers a wide range of topics (especially patterns and frameworks) associated with quality mobile software development. For students who just want training we recommend others sources, such as the Android Application Development Tutorials on YouTube.
4. **What is the most effective way to learn material covered in the course and to successfully complete the programming assignments?**
   We recommend watching the videos multiple times, looking for different levels of meaning in the diagrams and the examples. It's particularly important to carefully watch all the videos referenced in the programming assignment descriptions since they provide many relevant tips and insights. Likewise, we recommend reading the material pointed to by links in the slides, as well as material from the (optional) recommended reading. Naturally, participating in the online discussion forum (and ideally, a meetup group if one is available in your area) will help make the course material more engaging and personalized. Additional discussions of the programming assignment goals, requirements, and (ultimately) their solutions will be covered in "Virtual Office Hours" (see the FAQ entry on this topic below).
5. **Can students take this course if they have no prior experience with Android programming or programming with Java?**
   Our course assumes that students are comfortable programming in Java and have some experience programming Android apps. If you don't have any Java programming background you might consider taking the course Java for Complete Beginners. Likewise, you might also benefit from the Creative, Serious and Playful Science of Android Apps MOOC, which is a novice-friendly introduction to computer scien

See the FAQ at class.coursera.org/posa-002/wiki/FrequentlyAskedQuestions

Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android



Digital Learning Offerings

Douglas C. Schmidt (d.schmidt@vanderbilt.edu)
Associate Chair of Computer Science and Engineering,
Professor of Computer Science, and Senior Researcher
in the Institute for Software Integrated Systems (ISIS)
at Vanderbilt University

Coursera MOOCs on Pattern-Oriented Software Architecture (POSA)

- Spring 2014 Offering of Pattern-Oriented Software Architecture: Programming Mobile Services for Android Handheld Systems
- Spring 2013 Offering of Pattern-Oriented Software Architectures for Concurrent and Networked Software

Vanderbilt University Courses

- Playlist from my YouTube Channel videos from CS 251: Intermediate Software Design
- Playlist from my YouTube Channel videos from CS 282: Systems Programming for Android

Pearson LiveLessons Courses

- Design Patterns in Java

www.dre.vanderbilt.edu/~schmidt/DigitalLearning has more info