

Principles of Software Construction: Objects, Design, and Concurrency

Introduction, Overview, and Syllabus

Bogdan Vasilescu

Jonathan Aldrich



Principles of Software Construction: Objects, Design, and Concurrency

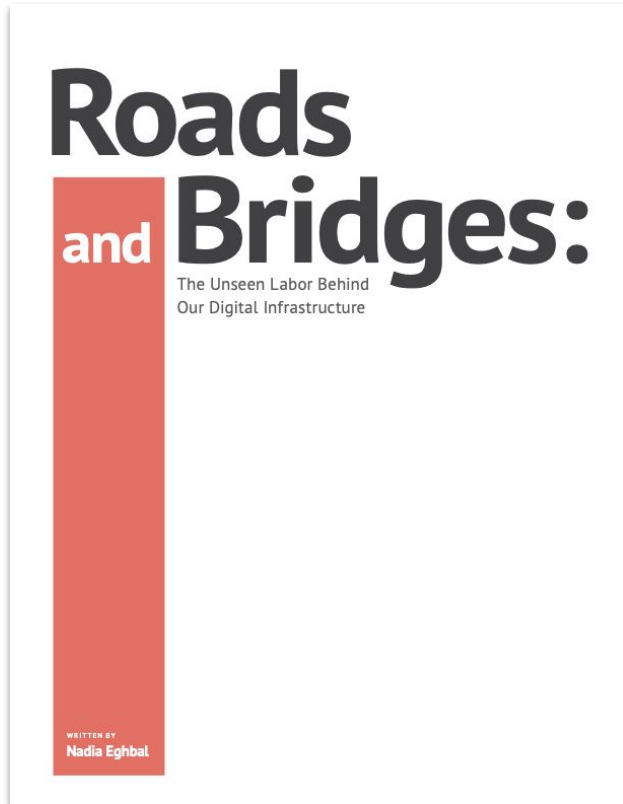
Introduction, Overview, and Syllabus

Bogdan Vasilescu

Jonathan Aldrich



How Modern Software Gets Built



“Building software is like constructing a building. A construction company wouldn’t build its hammers and drills from scratch, or source and chop all of the lumber themselves.”

ABOUT

Company

Press

Jobs

LEGAL

Terms

Privacy

Platform

Libraries

Libraries We Use

The following sets forth attribution notices for third party software that may be contained in portions of the Instagram product. We thank the open source community for all of their contributions.

AFNetworking

The following software may be included in this product: AFNetworking. This software contains the following license and notice below:

Copyright (c) 2011 Gowalla (<http://gowalla.com/>)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

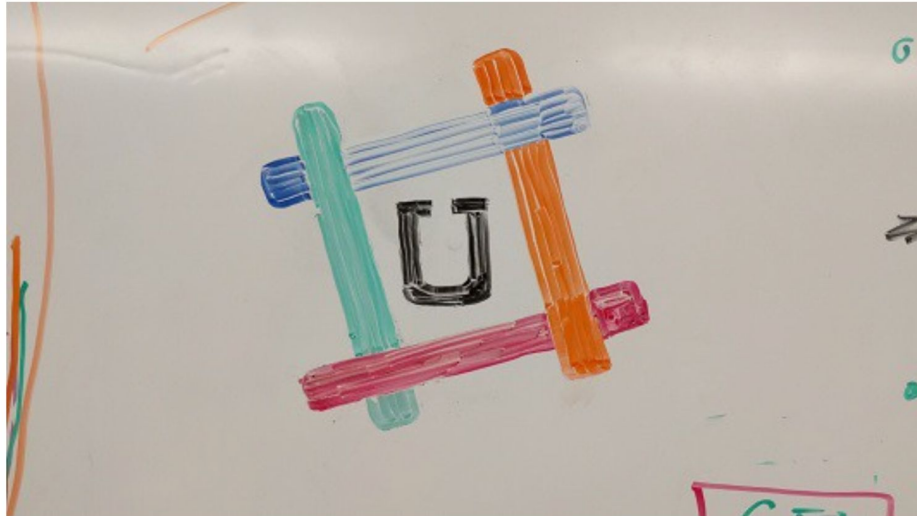
Copyright (c) 2013 AFNetworking (<http://afnetworking.com>)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,

Uber + Slack + Weekend

A story of open APIs

[Uber Developers](#)



What do you get when you combine a group of entrepreneurial hackers, a single weekend, and two open APIs? A brand new slack command to [request an Uber](#) without pulling out your phone — built by some talented young minds in just 48 hours.

A Few Questions

How many lines of code behind twitter.com?

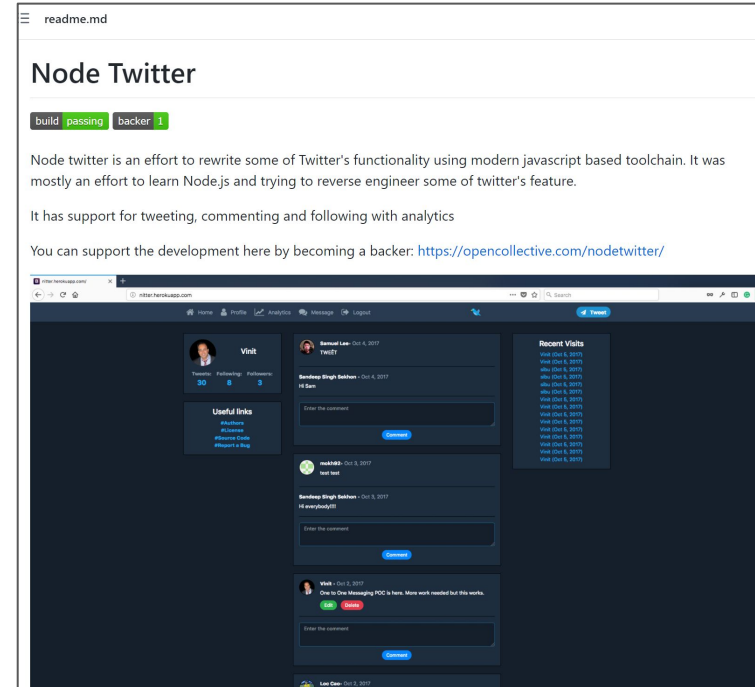
> A few million, maybe more

How many LOC to build an okay Twitter replica?

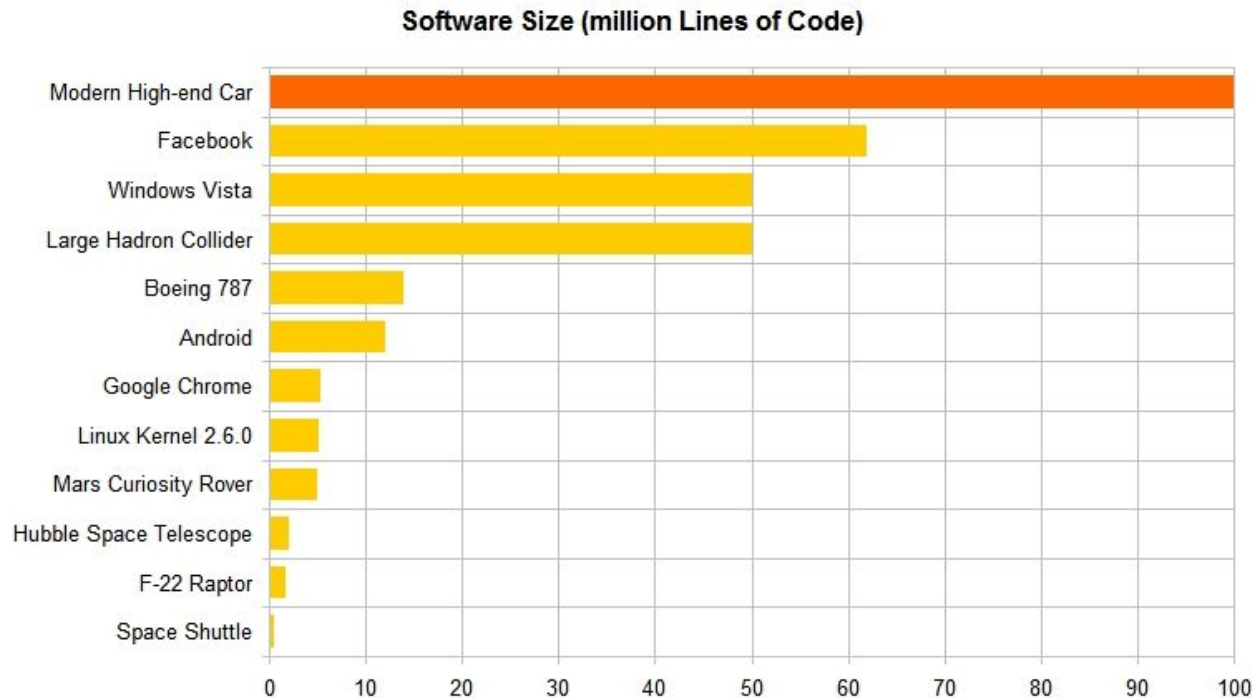
> A few 10K

How many LOC to run a Twitter replica?

> A few



Welcome to the era of “big code”



(informal reports)

Modern Software Engineering

- Nobody wants to write a million lines of code.
 - You don't want to write Twitter.

Modern Software Engineering

- Nobody wants to write a million lines of code.
 - You don't want to write Twitter.
 - (Aside) Sometimes you have to:

Twitter's Shift from Ruby to Java Helps it Survive US Election

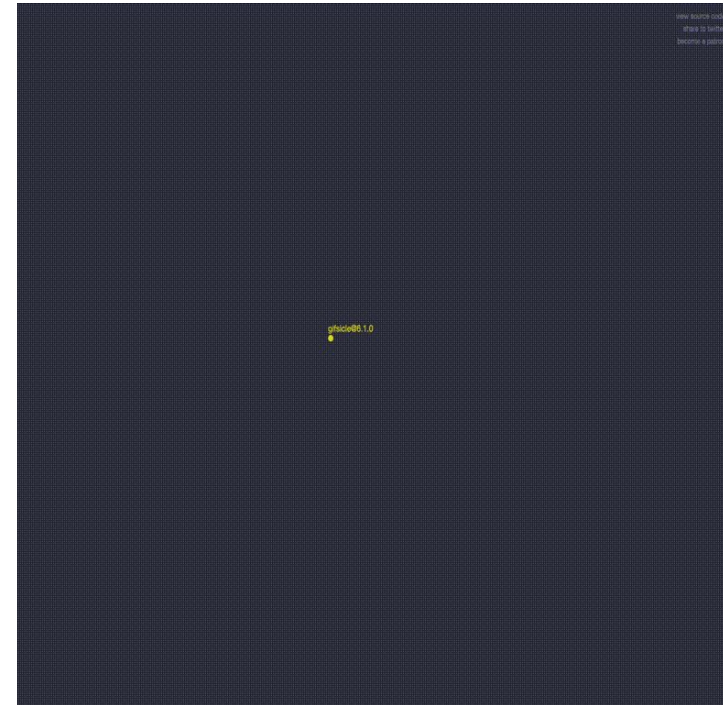
Nov 09, 2012 2 min read

Twitter's infamous [Fail Whale](#) was absent on US presidential election day, even as Twitter's servers were handling a surge of 327,452 tweets per minute, [according to](#) Mazen Rawashdeh, Twitter's VP of Infrastructure Operations Engineering. In total, there were 31 million election-related tweets during the course of the day, and the traffic continued to periodically spike - at one point reaching 15,107 tweets per second. To put this figure in context, on the 2008 election night [Twitter peaked](#) at 229 tweets per second.

Rawashdeh notes that Twitter has been seeing a change in usage pattern over the last year from brief spikes (for example related to the clock striking midnight on [New Year's Eve](#) or a [celebrity pregnancy announcement](#)) to more sustained peaks of traffic lasting several hours. This occurred, for example, during the [Olympics closing ceremony](#), [the NBA finals](#), and now with the election.

Modern Software Engineering

- Nobody wants to write a million lines of code.
 - You don't want to write Twitter.
- Instead, you use libraries
 - E.g., `import Android => +12M LOC`
 - You don't write most of the code you use
 - And why would you want to?
- And your libraries use libraries
 - Et cetera
 - <https://npm.anvaka.com/#/view/2d/gifsicle>



But “a few lines of code” does not mean easy!

- An engineer understands the pieces and how to put them together.
- But:
 - There are many (and always new) pieces.
 - They involve different and continuously changing programming languages and technologies.
 - There are many ways to compose applications, with different tradeoffs.
 - The implications can be very subtle.

NPM & left-pad: Have We Forgotten How To Program?

[David Haney](#)

Intro

Okay developers, time to have a serious talk. As you are probably already aware, this week React, Babel, and a bunch of other high-profile packages on NPM broke. The reason they broke is rather astounding:

A simple NPM package called [left-pad](#) that was a dependency of their code.

left-pad, at the time of writing this, [has 11 stars on GitHub](#). The entire package is [11 simple lines that implement a basic left-pad string function](#). In case those links ever die, here is the entire code of the left-pad package:

```
module.exports = leftpad;
function leftpad (str, len, ch) {
  str = String(str);
  var i = -1;
  if (!ch && ch !== 0) ch = ' ';
  len = len - str.length;
  while (++i < len) {
    str = ch + str;
  }
  return str;
}
```

What concerns me here is that *so many packages and projects* took on a **dependency** for a simple left padding string function, rather than their

What is Log4j? A cybersecurity expert explains the latest internet vulnerability, how bad it is and what's at stake

[Santiago Torres-Arias](#) December 22, 2021 8:12am EST

Log4Shell, an internet vulnerability that affects millions of computers, involves an obscure but nearly ubiquitous piece of software, Log4j. The software is used to record all manner of activities that go on under the hood in a wide range of computer systems.

Jen Easterly, director of the U.S. Cybersecurity & Infrastructure Security Agency, called Log4Shell the [most serious vulnerability](#) she's seen in her career. There have already been hundreds of thousands, perhaps millions, of [attempts to exploit the vulnerability](#).

So what is this humble piece of internet infrastructure, how can hackers exploit it and what kind of mayhem could ensue?

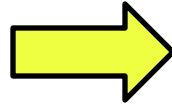
But “a few lines of code” does not mean easy!

- An engineer understands the pieces and how to put them together.
- But:
 - There are many (and always new) pieces.
 - They involve different and continuously changing programming languages and technologies.
 - There are many ways to compose applications, with different tradeoffs.
 - The implications can be very subtle.
- You need to become fluent in using and composing new systems.
And you’ll need to do it over and over again throughout your careers.

This class teaches principles of software construction

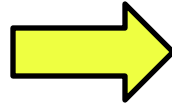
17-214/514: From Programs to Applications & Systems

Writing algorithms, data structures from scratch



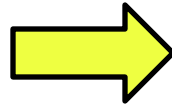
Reuse of libraries, frameworks

Functions with inputs and outputs



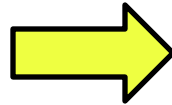
Asynchronous and reactive designs

Sequential and local computation



Parallel and distributed computation

Full functional specifications



Partial, composable, targeted models

Our goal: understanding both the **building blocks** and also the **design principles** for construction of software systems **at scale**

Equipment of a Modern Programmer

Less emphasis on:

(though not unimportant!)

- Clever algorithmics
- Low-level code (kernels, drivers)
- Writing common components
(command-line parsers, HTML)

More emphasis on:

- Using APIs, libraries (hw1)
- Quality assurance (hw2)
- Design for reuse, extension (hw3+)
- Flexibility in ecosystems (all)

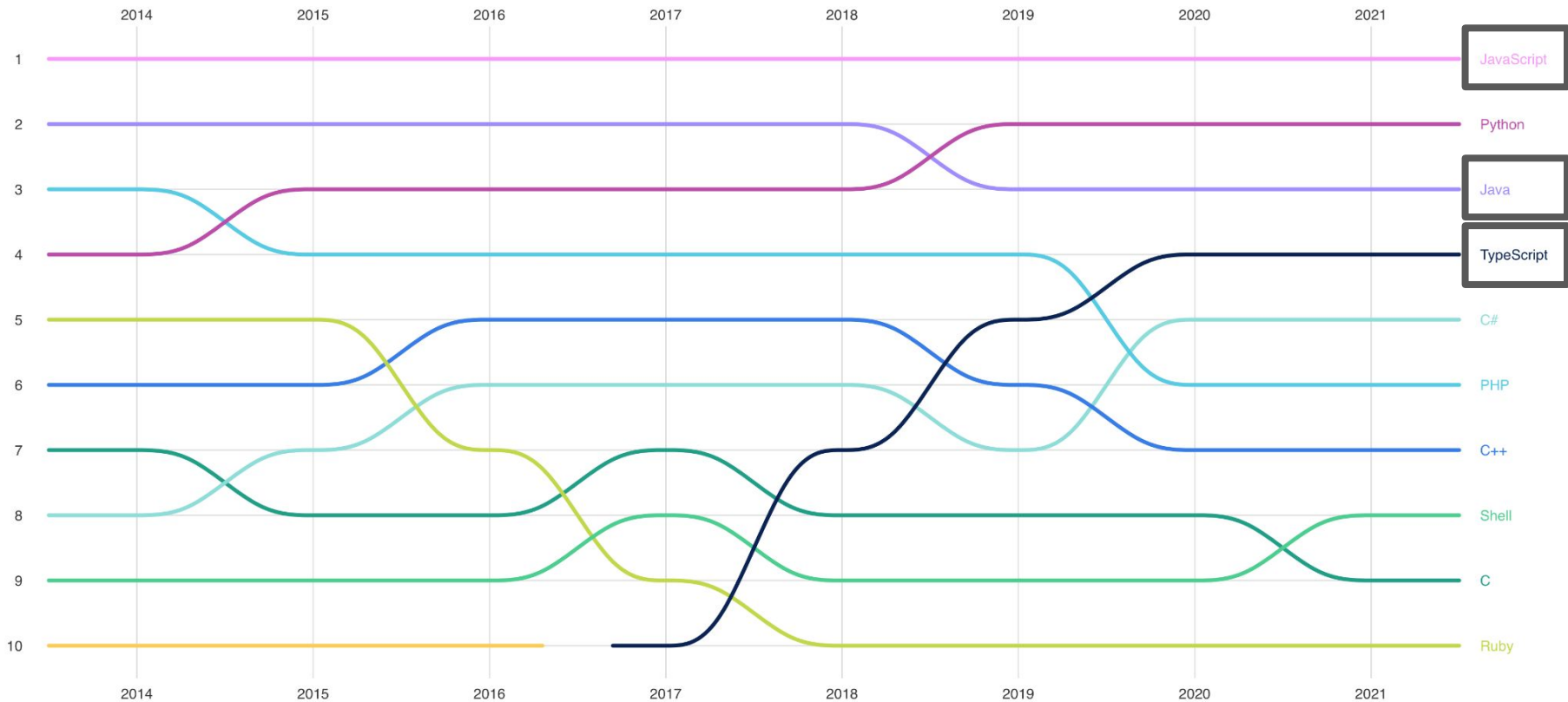
Flexibility & Ecosystems

Flexibility is a key skill, besides good design.

In this course:

- Learn to choose & use libraries
- Adopting new tools, troubleshooting
- Also, Java vs. JavaScript/TypeScript

Top languages over the years



Why are multiple programming languages used in the development of one product or piece of software?

Asked 3 years, 8 months ago · Active 1 year, 10 months ago · Viewed 42k times

▲ 120 ▼
I am a recent grad student aiming to start my Master's in Computer Science. I have come across multiple open source projects that really intrigue me and encourage me to contribute to them (CloudStack, OpenStack, moby, and Kubernetes to name a few). One thing I've found that the majority of them have in common is the use of multiple programming languages (like Java + Python + Go or Python + C++ + Ruby). I have already looked at this other question, which deals with how multiple programming languages are made to communicate with each other: [How to have two different programmings with two different languages interact?](#)

I want to understand the requirement that prompts enterprises to use multiple programming languages. What requirement or type of requirement makes the software architect or project lead say, "I'm proposing we use language X for task 1 and language Y for task 2"? I can't seem to understand the reason why multiple programming languages are used in the same product or software.

programming-practices

programming-languages

methodology



Why are multiple programming languages used in development?

Asked 3 years, 8 months ago

▲ 120 ▼

🔖 54

🕒

I am a recent graduate across multiple programming languages. I've found that looking at this question has made me think about different languages used in development. I want to understand why multiple languages are used in a project. I can't seem to understand the same product.

programming-

▲ 162 ▼



I can't seem to understand the reason as to why multiple programming languages are used in the same project.

It is quite simple: the

Read Michael L. Scott

Some programming languages also high-level productivity development is important runtime performance

Other programming languages compiled implementation (...); often their special matters, it is preferred

Some external libraries mind. You may need perhaps by writing

In practice, it is unproductive of the runtime. In practice

▲ 20 ▼



[This answer](#) has superb coverage and links on why different languages can provide distinct benefits to a project. However, there is quite a bit more than just language suitability involved in why projects end up using multiple languages.

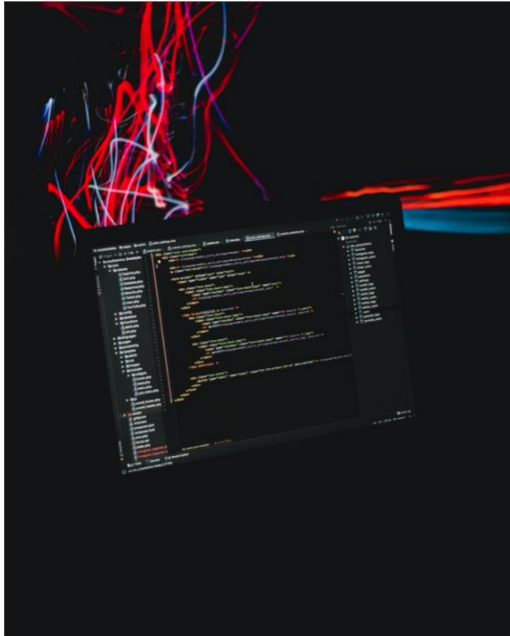
Projects end up using multiple languages for six main reasons:

1. Cost benefits of reusing code written in other languages;
2. The need to include and accommodate legacy code;
3. Availability of coders for specific languages;
4. The need for special languages for specialty needs;
5. Legacy language biases; and
6. Poor project management (unplanned multi-language use).

Reasons 1-4 are positive reasons in the sense that addressing them directly can help a project conclude faster, more efficiently, with a higher-quality product, and with easier long-term support. Reasons 5 and 6 are negative, symptoms of resistance to needed change, poor planning, ineffective management, or some combination of all of these factors. These negative factors unfortunately are common causes of "accidental" multi-language use.

What is a Polyglot Programmer — And Why You Should Become One

Paul Azorin



Whenever a person takes an interest in programming, they set out to choose where to start in a landscape that's vast and widely dynamic. Usually, their journey begins with choosing one of the many, many programming languages out there. Then, they learn everything they can about that language and, hopefully, continue their journey coding software using what they've learned.

That's what the path of most developers looks like: they learn the ins and outs of a particular language, extend their reach through frameworks, and they start to gain experience out in the fields. That's a great way to get into the programming world but, if you stop there, you aren't doing yourself any favor.

Though you might think it's best to stick with a particular programming language and become an expert in it (like many software engineers before you), limiting your toolkit to just one language can be, well, limiting your

How to Become a Polyglot Programmer

Pen Magnet



Photo by [Juan Gomez](#) on [Unsplash](#)

Wiktionary defines [Polyglot](#) as: (notice #4?)

- 1: *One who has mastered, notably speaks, several languages.*
- 2: *A publication containing several versions of the same text, or the same subject matter in several languages; especially, the Bible in several languages.*
- 3: *A mixture of languages or nomenclatures.*
- 4: *(programming) A program written in multiple programming languages.*

Here We Go Again: Why Is It Difficult for Developers to Learn Another Programming Language?

Nischal Shrestha
NC State University
Raleigh, North Carolina
nshrest@ncsu.edu

Colton Botta
NC State University
Raleigh, North Carolina
cgbotta@ncsu.edu

Titus Barik
Microsoft
Redmond, Washington
titus.barik@microsoft.com

Chris Parnin
NC State University
Raleigh, North Carolina
cjparnin@ncsu.edu

ABSTRACT

Once a programmer knows one language, they can leverage concepts and knowledge already learned, and easily pick up another programming language. But is that always the case? To understand if programmers have difficulty learning additional programming languages, we conducted an empirical study of Stack Overflow questions across 18 different programming languages. We hypothesized that previous knowledge could potentially interfere with learning a new programming language. From our inspection of 450 Stack Overflow questions, we found 276 instances of interference that occurred due to faulty assumptions originating from knowledge about a different language. To understand why these difficulties occurred, we conducted semi-structured interviews with 16 professional programmers. The interviews revealed that programmers make failed attempts to relate a new programming language with what they already know. Our findings inform design implications for technical authors, toolsmiths, and language designers, such as designing documentation and automated tools that reduce interference, anticipating uncommon language transitions during language design, and welcoming programmers not just into a language, but its entire ecosystem.

PRELUDE

Peter Norvig wrote a guide, “Python for Lisp Programmers” [48], to teach Python from the perspective of Lisp. We interviewed Peter regarding this transition and he described a few challenging aspects of switching to Python, such as how lists are not treated as a linked list and solutions where he previously used macros required re-thinking. When asked about the general problem of switching programming languages, he said:

Most research is on beginners learning languages. For experts, it's quite different and we don't know that process. We just sort of assume if you're an expert you don't need any help. But I think that's not true! I've only had a couple times when I had to deal with C++ and I always felt like I was lost. It's got all these weird conventions going on. There's no easy way to be an expert at it and I've never found a good answer to that and never felt confident in my C++.

Peter believes that learning new languages is difficult—even for experts—despite their previous experience working with languages. Is Peter right?

Outcomes, hopefully

You'll learn to be:

- An architect, approaching programming as design
 - This is the only way to scale up to larger systems
 - You'll learn a rich vocabulary, of both components and their combinations
- A polyglot, able to pick up new languages and libraries
 - Because you know the underlying concepts
 - And you've had plenty of practice reading documentation, debugging setups
- An engineer, safeguarding the quality of your programs
 - You'll get dextrous at testing, be explicit about specification
 - You'll know the tools that improve your work

Principles of Software Construction: **Objects**, Design, and Concurrency

Introduction, Overview, and Syllabus

Bogdan Vasilescu

Jonathan Aldrich



Objects in the real world



Object-oriented programming

Programming based on structures that contain both data and methods



```
public class Bicycle {  
    private int speed;  
    private final Wheel frontWheel, rearWheel;  
    private final Seat seat;  
    ...  
    public Bicycle(...) { ... }  
  
    public void accelerate() {  
        speed++;  
    }  
  
    public int speed() { return speed; }  
}
```

Principles of Software Construction: Objects, **Design**, and Concurrency

Introduction, Overview, and Syllabus

Bogdan Vasilescu

Jonathan Aldrich



User needs
(Requirements)

Miracle?

Code

User needs
(Requirements)

Miracle?

Code

Maintainable?
Testable?
Extensible?
Scalable?
Robust? ...

A typical Intro CS design process

1. Discuss software that needs to be written
2. Write some code
3. Test the code to identify the defects
4. Debug to find causes of defects
5. Fix the defects
6. If not done, return to step 1

Better software design

- Think before coding: broadly consider quality attributes
 - Maintainability, extensibility, performance, ...
- Propose, consider design alternatives
 - Make explicit design decisions

Sorting with a configurable order, version A

```
static void sort(int[] list, boolean ascending) {  
    ...  
    boolean mustSwap;  
    if (ascending) {  
        mustSwap = list[i] > list[j];  
    } else {  
        mustSwap = list[i] < list[j];  
    }  
    ...  
}
```

Sorting with a configurable order, version B

```
interface Order {
    boolean lessThan(int i, int j);
}

class AscendingOrder implements Order {
    public boolean lessThan(int i, int j) { return i < j; }
}

class DescendingOrder implements Order {
    public boolean lessThan(int i, int j) { return i > j; }
}

static void sort(int[] list, Order order) {
    ...
    boolean mustSwap =
        order.lessThan(list[j], list[i]);
    ...
}
```

Sorting with a configurable order, version B'

```
const ASC = function(i: number, j: number): boolean {
  return i < j;
}
const DESC = function(i: number, j: number): boolean {
  return i > j;
}

function sort(
  list: number[],
  order: (number, number) => boolean) {
  ...
  boolean mustSwap = order(list[j], list[i]);
  ...
}
> sort(list, ASC);
```

Which version is better?

Version A:

```
static void sort(int[] list, boolean ascending) {  
    ...  
    boolean mustSwap;  
    if (ascending) {  
        mustSwap = list[i] > list[j];  
    } else {  
        mustSwap = list[i] < list[j];  
    }  
    ...  
}
```

Version B':

```
interface Order {  
    boolean lessThan(int i, int j);  
}  
class AscendingOrder implements Order {  
    public boolean lessThan(int i, int j) { return i < j; }  
}  
class DescendingOrder implements Order {  
    public boolean lessThan(int i, int j) { return i > j; }  
}  
  
static void sort(int[] list, Order order) {  
    ...  
    boolean mustSwap =  
        order.lessThan(list[j], list[i]);  
    ...  
}
```

it depends

it depends

Depends on what?
What are scenarios?
What are tradeoffs?

it depends

**Depends on what?
What are scenarios?
What are tradeoffs?**

**In this specific case, what
would you recommend?
(Engineering judgement)**

"Software engineering is the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems, preferentially by applying scientific knowledge, developing software systems in the service of mankind.

"**Software engineering** is the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems, preferentially by applying scientific knowledge, developing software systems in the service of mankind.

Software engineering entails making **decisions** under constraints of limited time, knowledge, and resources. [...]

Engineering quality resides in engineering judgment. [...]

Engineering requires reconciling conflicting constraints. [...]

Engineering skills improve as a result of careful systematic reflection on experience. [...]

Costs and time constraints matter, not just capability. [...]

Software Engineering for the 21st Century: A basis for rethinking the curriculum
Manifesto, CMU-ISRI-05-108

Goal of software design

- Think before coding
- For each desired program behavior there are infinitely many programs
 - What are the differences between the variants?
 - Which variant should we choose?
 - How can we synthesize a variant with desired properties?
- Consider qualities: Maintainability, extensibility, performance, ...
- Make explicit design decisions

Tradeoffs?

```
static void sort(int[] list, boolean ascending) {  
    ...  
    boolean mustSwap;  
    if (ascending) {  
        mustSwap = list[i] > list[j];  
    } else {  
        mustSwap = list[i] < list[j];  
    }  
    ...  
}
```

```
interface Order {  
    boolean lessThan(int i, int j);  
}  
class AscendingOrder implements Order {  
    public boolean lessThan(int i, int j) { return i < j; }  
}  
class DescendingOrder implements Order {  
    public boolean lessThan(int i, int j) { return i > j; }  
}
```

```
static void sort(int[] list, Order order) {  
    ...  
    boolean mustSwap =  
        order.lessThan(list[j], list[i]);  
    ...  
}
```

Some qualities of interest, i.e., *design goals*

Functional correctness	Adherence of implementation to the specifications
Robustness	Ability to handle anomalous events
Flexibility	Ability to accommodate changes in specifications
Reusability	Ability to be reused in another application
Efficiency	Satisfaction of speed and storage requirements
Scalability	Ability to serve as the basis of a larger version of the application
Security	Level of consideration of application security

Source: Braude, Bernstein,
Software Engineering. Wiley 2011

Using a design process

- A design process organizes your work
- A design process structures your understanding
- A design process facilitates communication

Semester overview

- Introduction to Object-Oriented Programming
- Introduction to **design**
 - **Design** goals, principles, patterns
- **Designing** objects/classes
 - **Design** for change
 - **Design** for reuse
- **Designing** (sub)systems
 - **Design** for robustness
 - **Design** for change (cont.)
- **Design** for large-scale reuse

Crosscutting topics:

- Building on libraries and frameworks
- Building libraries and frameworks
- Modern development tools: IDEs, version control, refactoring, build and test automation, static analysis
- Testing, testing, testing
- Concurrency basics

Preview: Design goals, principles, and patterns

- **Design goals** enable evaluation of designs
 - e.g., maintainability, reusability, scalability
- **Design principles** are heuristics that describe best practices
 - e.g., high correspondence to real-world concepts
- **Design patterns** codify repeated experiences, common solutions
 - e.g., template method pattern

Software Engineering at CMU

- 17-214: “Code-level” design
 - extensibility, reuse, concurrency, functional correctness, medium-size to large programs
- 17-313: “Human aspects” of software development
 - requirements, team work, balancing qualities, scheduling, costs, risks, business models
- 17-413 Practicum, Seminar, Internship
- SE electives: SE4Startups, Program Analysis, Machine Learning in Production
- Various master-level courses on requirements, architecture, software analysis, deep learning for SE, etc
- SE Minor/Concentration: <https://www.s3d.cmu.edu/education/undergrad>

**This is not a
Java/JavaScript course**

This is not a Java/JavaScript course

**but you will write a
lot of
Java/JavaScript code**

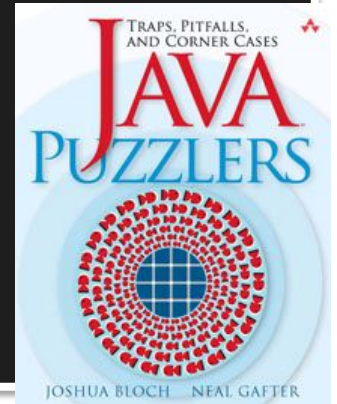
```
int a = 010 + 3;  
System.out.println("A" + a);
```

```
int a = 010 + 3;  
System.out.println("A" + a);
```

```
const a = 010 + 3;  
console.log("A" + a);
```

```
int a = 010 + 3;  
System.out.println("A" + a);
```

```
const a = 010 + 3;  
console.log("A" + a);
```



Java + JavaScript / TypeScript

Focus on design concepts and cross-cutting skills, not programming language

Language proficiency through practice and homeworks

Lectures show examples in pseudo code, Java, JavaScript, TypeScript, and other languages

Both Java and TypeScript for homeworks (sometimes your choice)

```
int a = 010 + 3;  
System.out.println("A"
```


```
const a = 010 + 3;  
console.log("A" + a);
```

Java **AND** TypeScript/JavaScript

- HW 1&2: Both
 - Flashcard learning app (command line)
- HW 3: Java + HW 5: TypeScript/JavaScript
 - Board game with web interface (could also be a mobile app)
- Your choice:
 - HW4: Static website generator / CMS (command line application)
 - HW6: Data analysis and visualization tool (desktop/web application)

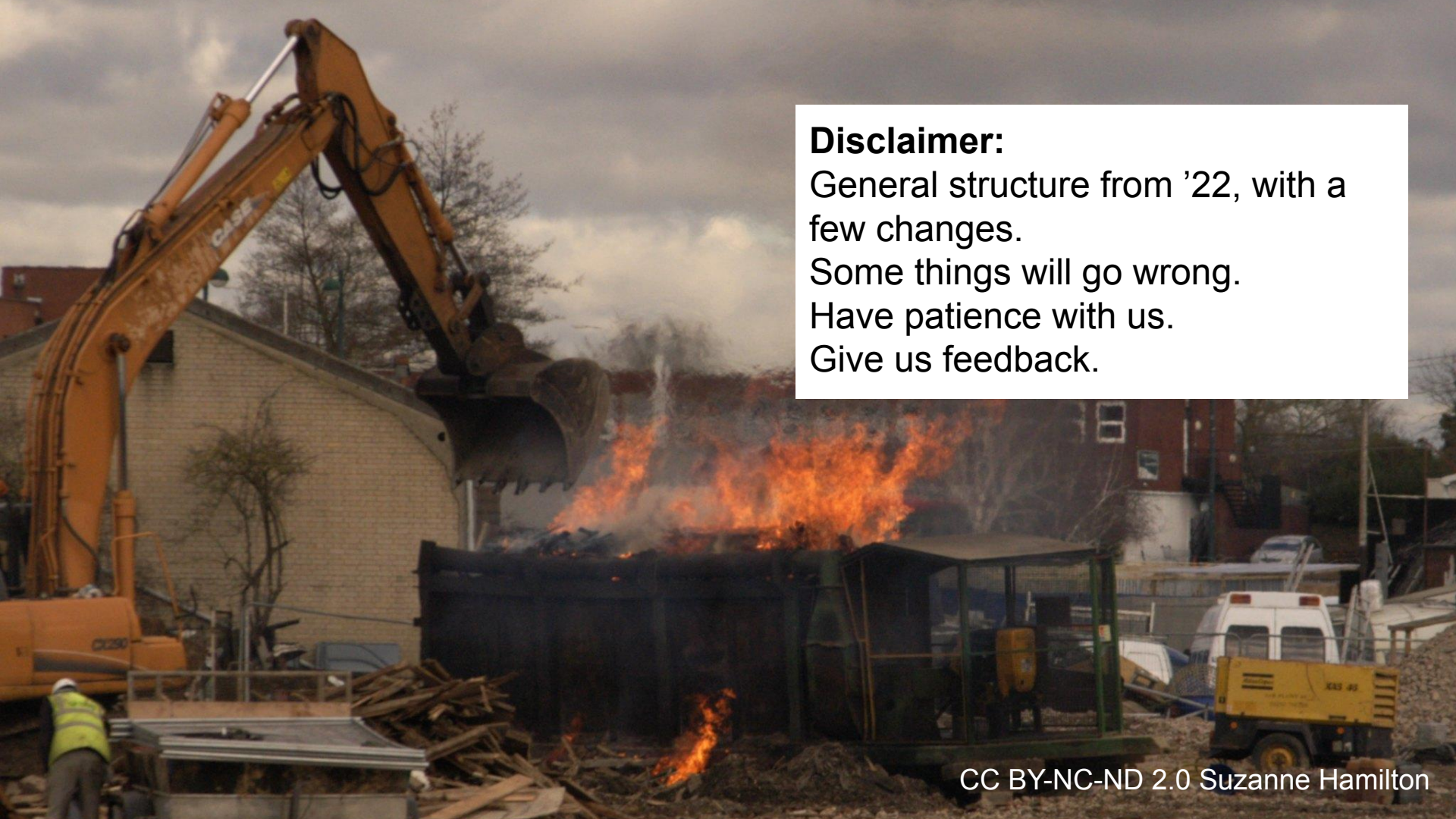
Recitations will provide tools/examples in both languages.

COURSE ORGANIZATION

A photograph of a large fire with bright orange and yellow flames and thick, dark smoke. In the foreground, a wooden sign stands on a grassy area. The sign has three horizontal bars. The top bar reads "SENIOR CENTER". The middle bar is a black banner with white text that reads "WEAR A MASK", "WASH YOUR HANDS", "SOCIAL DISTANCE", and "STAY SAFE". The bottom bar reads "COME JOIN US".

Trying to get back to normal with ...
gestures widely everything

Talk to us about concerns and
accommodations



Disclaimer:
General structure from '22, with a few changes.
Some things will go wrong.
Have patience with us.
Give us feedback.

Course materials

Course website (syllabus, slides, calendar): <https://cmu-17-214.github.io/s2023/>

Discussions, questions, announcements: Piazza

Assignments, readings, and grades: Canvas (and Gradescope)

Homework submission: GitHub (signup instructions in assignment) and other tools

GitHub ID/start of class survey.

Please fill out: <https://forms.gle/wdcaD4C3F3FWMBET8>

- Will also put on Piazza

If you don't have a github account, signing up is fast. Do that right now and fill out the survey.

Are you finished? Have you set up pushing via ssh key-pair or PAT?

- <https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh>
- <https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token>

It's OK if you don't finish setting up push right now, but you'll need to do it to do the homework. We discuss this in recitation, or can help via Piazza or Office Hours.



Course preconditions, enrollment

- 15-122 or equivalent: Basic programming skills in any language, algorithms and data structures (lists, graphs, sorting, binary search)
- 21-127 or equivalent: Basic discrete math concepts, logic
- If you are on the waitlist, or want to change sections, email this person:
Samantha Mudrinich smudrini@andrew.cmu.edu
- Note: you may have to unregister from other classes that create schedule conflicts. Also from “easiest to get into” to “hardest to get into” the section list is: D, A, E, C, G, F, B

Course staff

Bogdan Vasilescu

vasilescu@cmu.edu, TCS 326

Jonathan Aldrich

jonathan.aldrich@cs.cmu.edu, TCS 422

Teaching assistants:

Matt, Jake, Hamza, Can, Ruiying, Ariane, Eileen, Terry, Terry, Haozhe,
Meixuan, Stella, Teresa, Minyi



Course meetings

- Lectures: Tuesday and Thursday 3:30 – 4:50pm
- We will record and post recordings of lectures to Canvas, but we won't live stream.
- Recitations: Wednesdays 9am - ... - 7:50pm
 - Preparing for homeworks, hands-on practice, supplementary material
 - Starting tomorrow! (setting up environments -- relevant for HW1)
 - Recitations are not recorded, but handouts will be on Piazza.
- Office hours: see course web page

*Recitation
attendance
is required*

Homework & Exams

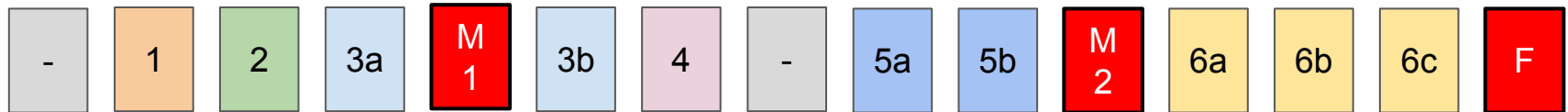
6 homeworks, 4 smaller + 2 large (with milestones), 1000 points total

(1) intro, (2) testing, (3) first design, (4) fixing design,
(5) extensibility + GUI, (6) framework and API design

Homeworks and milestones usually due Mondays (but some Fridays), see course calendar

Homework 1 due January 30

Two midterms + final

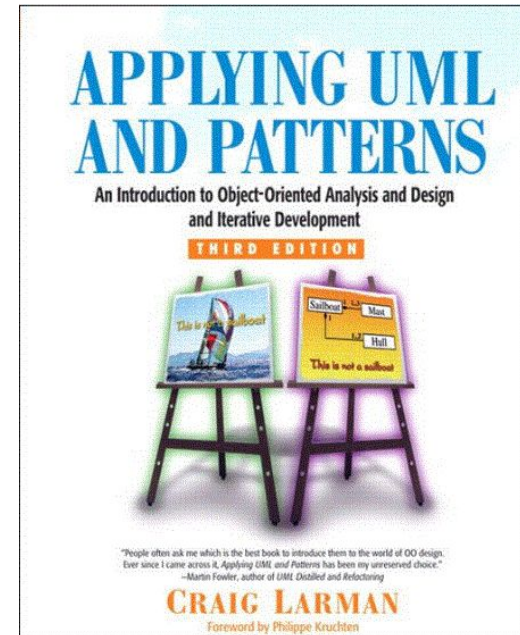
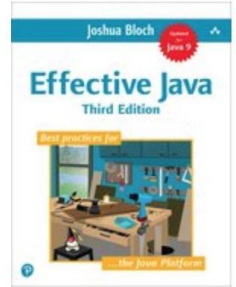


Late day policy

- See syllabus on course web page for details
- 2 possible late days per deadline (some exceptions may be announced)
 - 5 total free late days for semester (+ separate 2 late days for assignments done in pairs)
 - 10% penalty per day after free late days are used
 - but we won't accept work 3 days late
- Extreme circumstances – talk to us

Textbooks

- Craig Larman. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.** 3rd Edition. Prentice Hall. 2004. ISBN 0-13-148906-2
- Joshua Bloch. **Effective Java**, Third Edition. Addison-Wesley, ISBN 978-0-13-468599-1.
- Selective other readings throughout the semester
- Webpage also has pointers for references for Java and Typescript!
- Occasional in-class reading quizzes after reading assignment due
- **Electronic versions are all available for free through CMU library**



Approximate grading policy

- 50% assignments
- 20% midterms (2 x 10% each)
- 20% final exam
- 10% quizzes and participation.
 - Quizzes will be given in Lecture. You can miss a small number of quizzes without penalty. Please reach out if your circumstances are extenuating.

We reserve the right to be flexible with mappings between percentage and letter grades, but it will be no stricter than A=90-100%, B=80-90%, etc, and we don't have a hard curve.

Collaboration policy

- See course web page for details!
- We expect your work to be your own
- Do not release your solutions (not even after end of semester)
- Ask if you have any questions
- If you are feeling desperate, please reach out to us
 - Always turn in any work you've completed before the deadline
- We run cheating detection tools. Trust us, academic integrity meetings are painful for everybody

10% quizzes and participation / attendance

- Recitation participation counts toward your participation grade
- Lecture has in-class quizzes

The key to your success in this course is your regular engagement with course activities, staff, and other students

Summary

- Software engineering requires decisions, judgment
- Good design follows a process
- You will get lots of practice in 17-214!