DePaul University

Course Overview

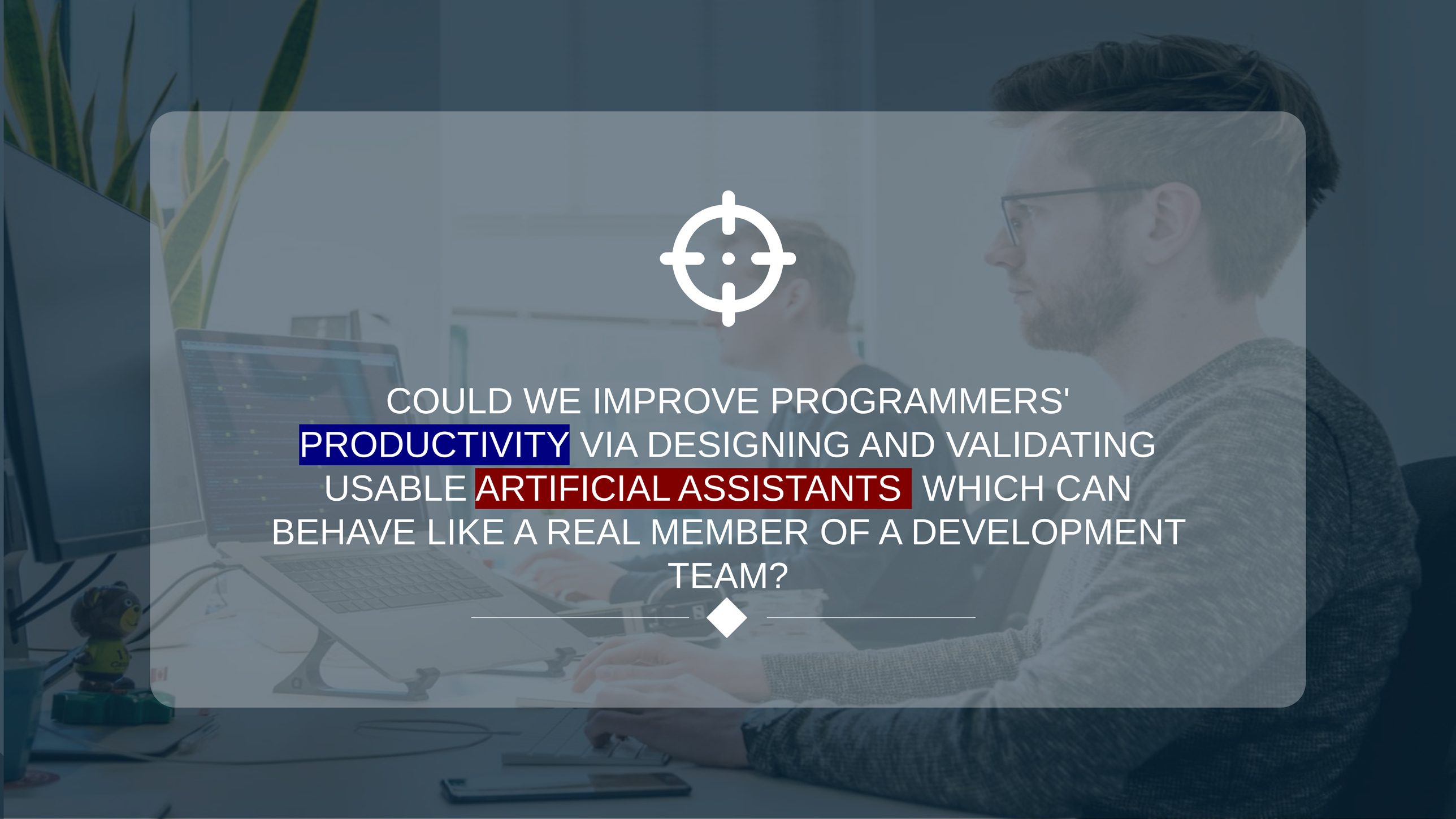
Object-oriented Software Development
SE 350– Spring 2021

Vahid Alizadeh

Week 1.1
March 30, 2021

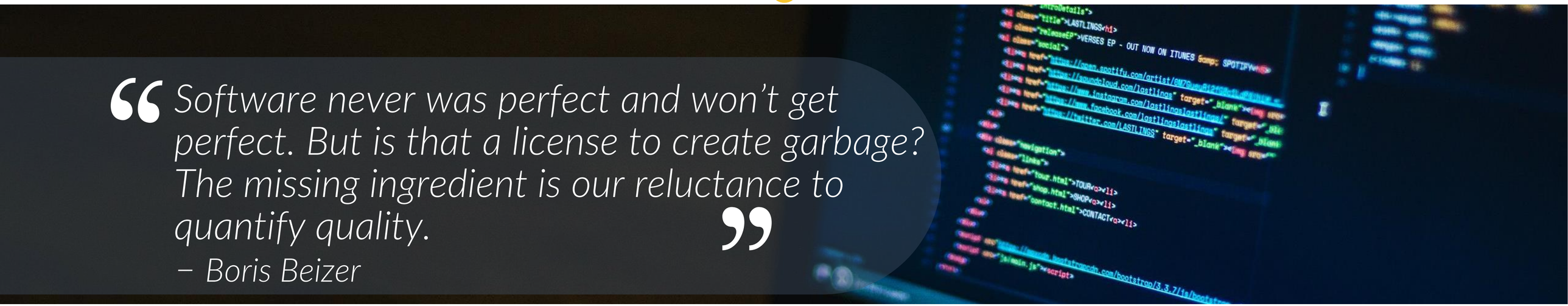
Introduction

Who am I? Who are you?



COULD WE IMPROVE PROGRAMMERS' PRODUCTIVITY VIA DESIGNING AND VALIDATING USABLE ARTIFICIAL ASSISTANTS WHICH CAN BEHAVE LIKE A REAL MEMBER OF A DEVELOPMENT TEAM?

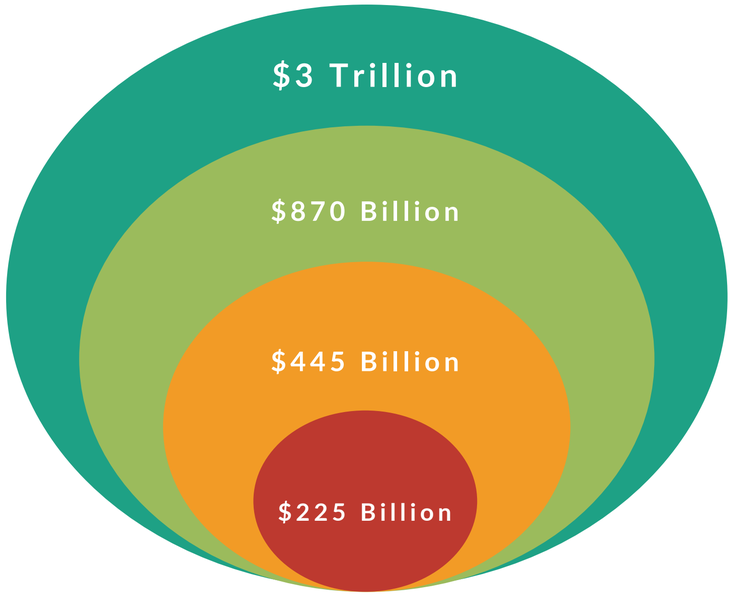
The Importance of Software Quality



> *Software never was perfect and won't get perfect. But is that a license to create garbage? The missing ingredient is our reluctance to quantify quality.*
>
> *– Boris Beizer*

Software quality can have a serious effect on the bottom-line.

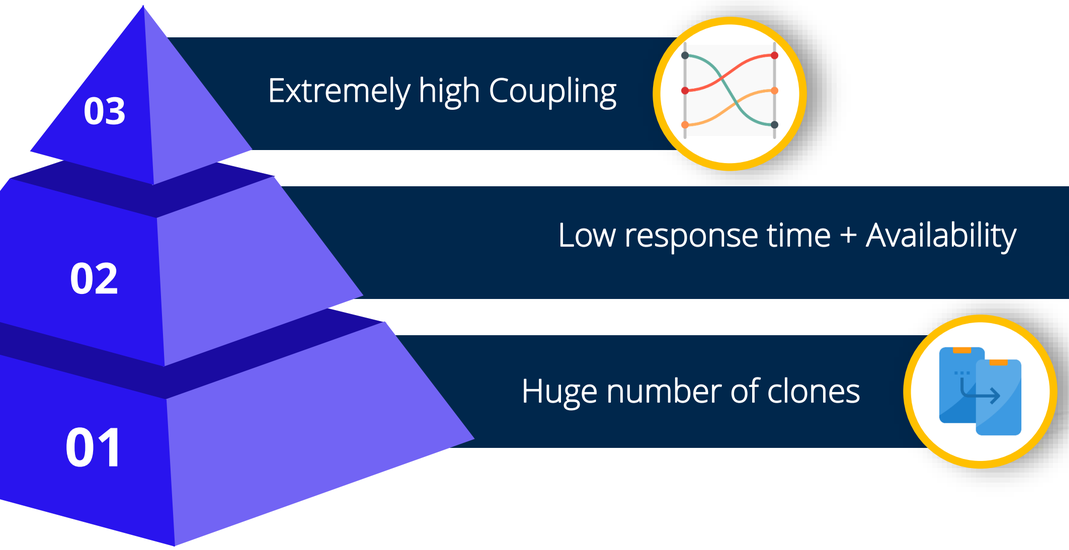
- malfunctions caused by defective software
- Transnational organized crime
- Cybercrime
- Air Pollution Deaths

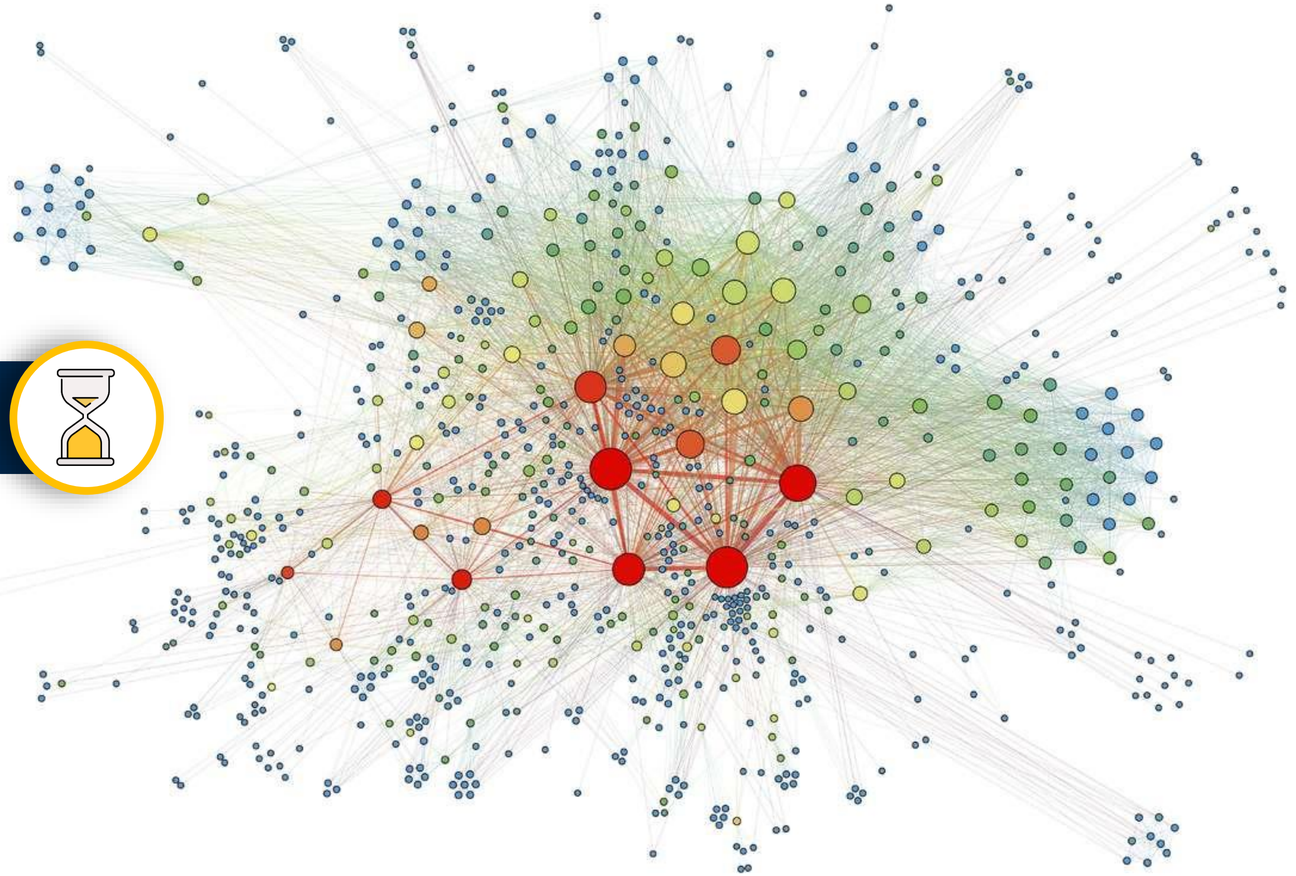


Global Economy Costs per Year

Real-World Example

Services Architecture





Context

Software projects evolve and change continuously

Increases the complexity of project

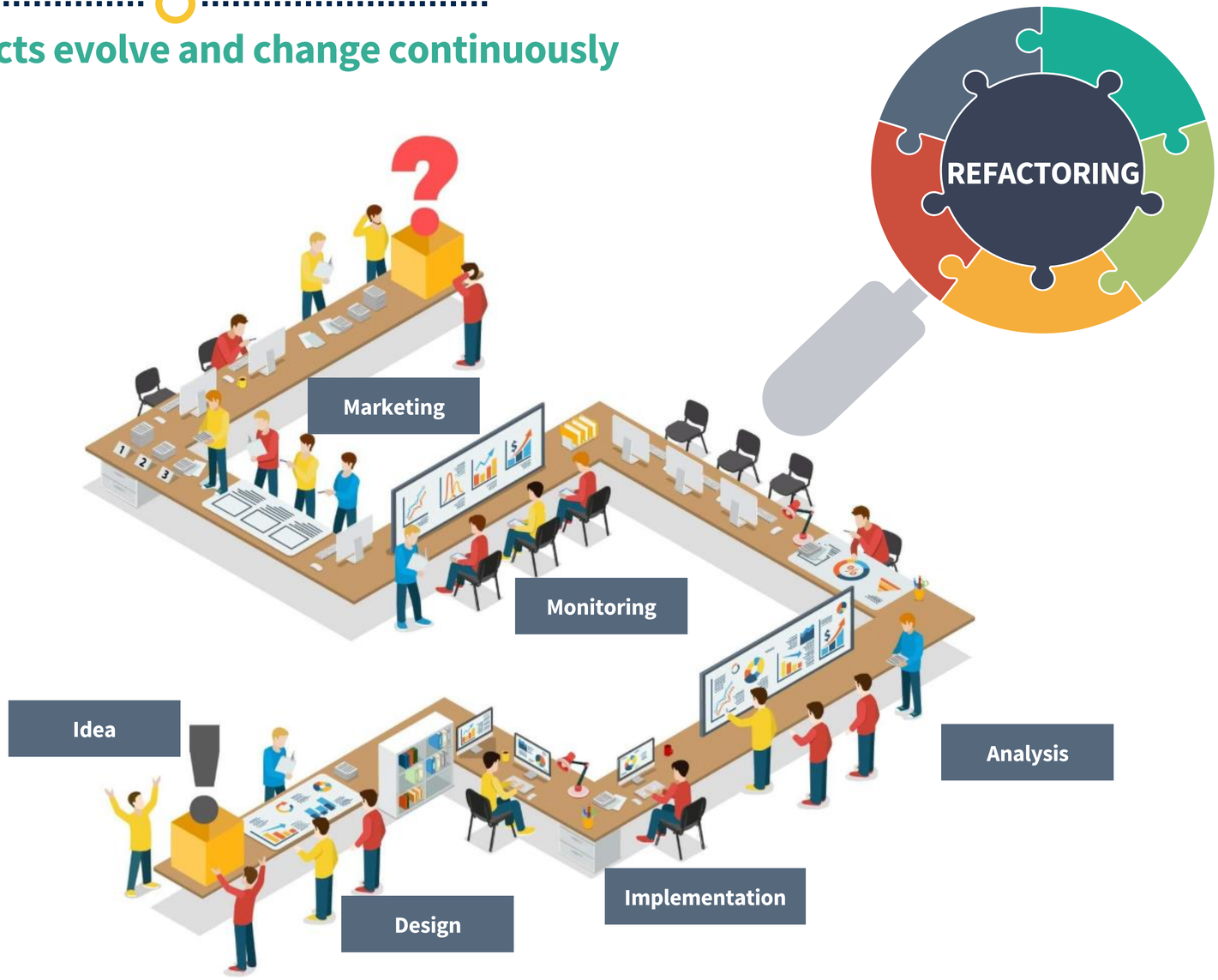
- 60% of developers' time in understanding the existing software

Reduces the productivity

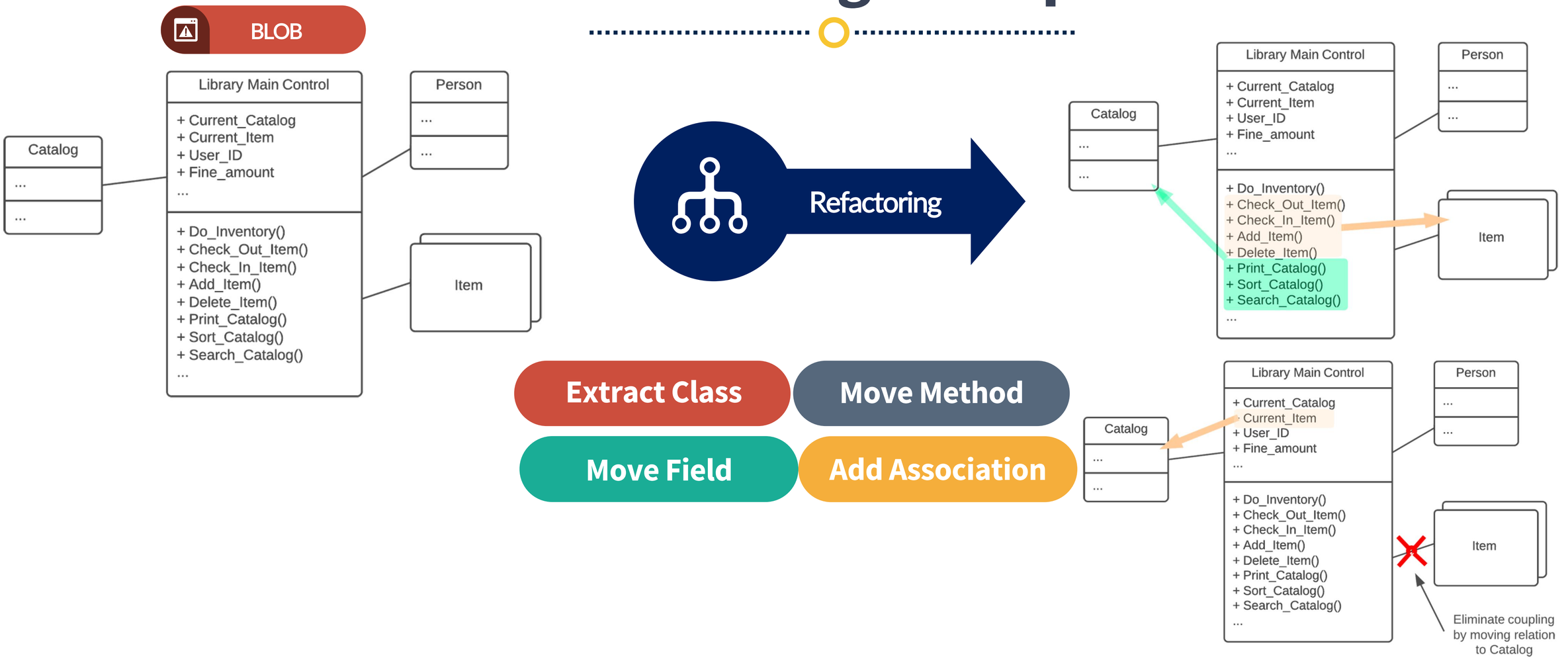
- 76% of Software Engineers working on maintenance tasks

Increases cost of maintenance

- 67% of Software Budget allocated to maintenance



Refactoring Example



"A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior"

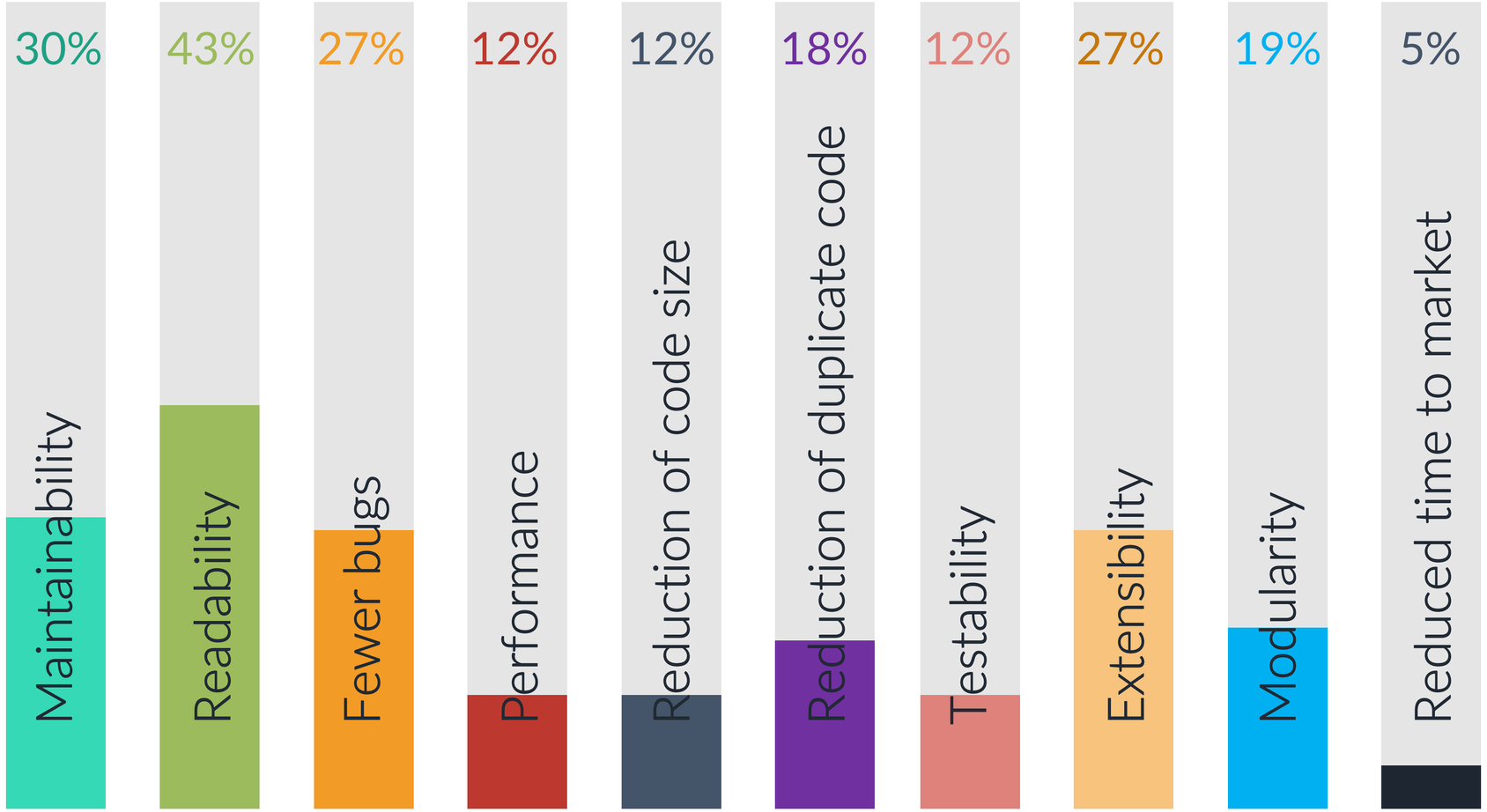
– M. Fowler [1999]

Which one is easier to work with?

Refactoring Effect

An Empirical Study of Refactoring Challenges and Benefits at Microsoft

Miryung et al. "An empirical study of refactoring challenges and benefits at Microsoft." IEEE Transactions on Software Engineering (2014)



But Refactoring is…

Time-consuming

Boring

Error-prone

What do experienced developers need:

> *"I don't feel a great need for automated refactoring tools, but I would like* ***code understanding and visualization tools*** *to help me make sure that my manual refactorings are valid."*

> *""I'd love a tool that could* ***estimate the benefits of refactoring****. Also, it'd be awesome to have better tools to help figure out who knows a lot about the existing code"*

Refactoring Problems

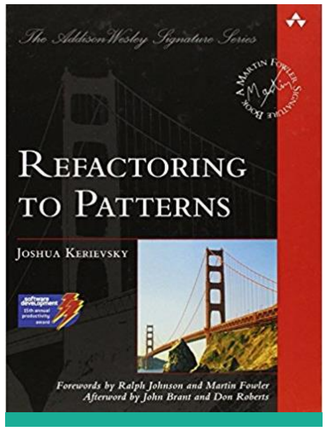
WHAT to Improve in Software Health?

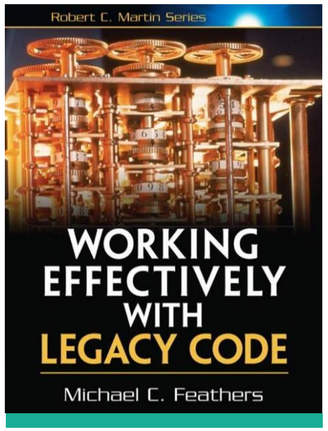
WHEN to Improve Software Health?

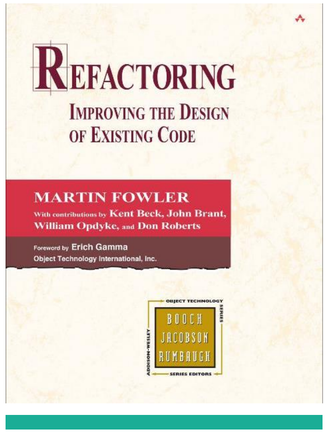
HOW to Improve Software Health?

Current State of Refactoring

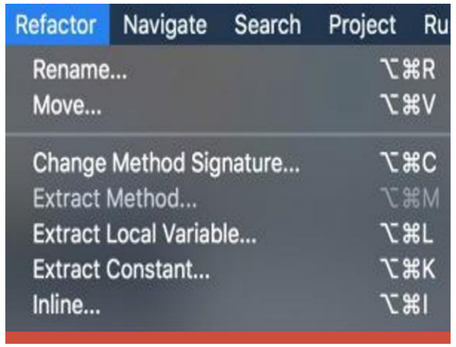
Approaches / Challenges







1999

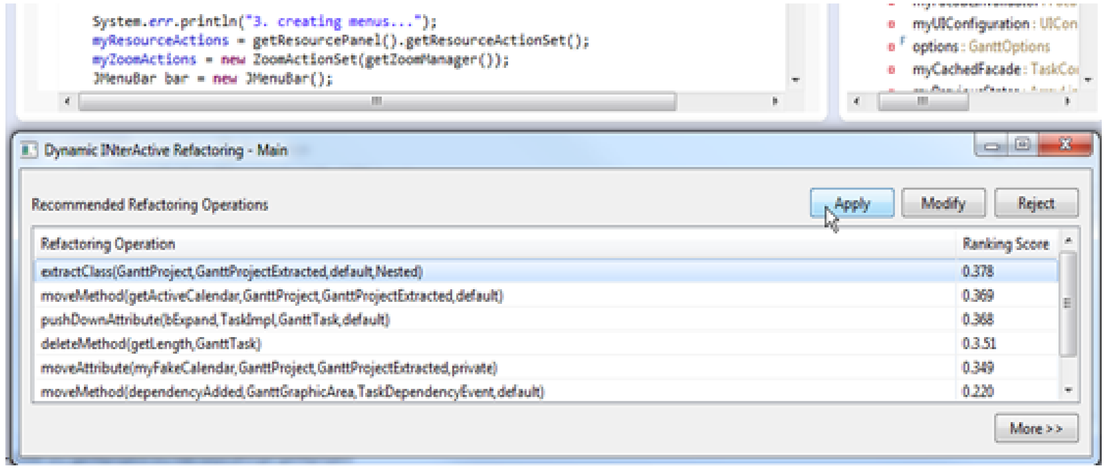


Manual

- error-prone
- time consuming
- not scalable
- not useful for radical refactoring

Fully-automated

- lacks flexibility
- fails to consider developer perspective
- proposes a long static list of Refactorings
- not always lead to the desired architecture



Semi-automated/Interactive

- Expensive process
- Evaluation of many refactorings
- Have to consider many quality attributes

Refactoring Tools/Techniques are not aligned with *Continuous Integration*

Your turn!

- **Introduce yourself**
  - Your background / experiences
  - Future plan (Dream job)
  - Your interests
  - Why did you take this course?
  - What is your course load this semester?
  - How do you feel about Online course modality?
  - Any fun fact about yourself?
- **Any questions about this course?**

Object-oriented Software Development

Course Overview

Class Schedule

- **Object-oriented Software Development – SE 350**
- Spring 2021
- Online Zoom Sessions
  - **Required** for **Sync** (601-602) sections
  - **Optional** for **Async** (610-611) sections
  - Recordings will be available for all sections in D2L
  - You can find the schedule and links in D2L
- **Section 601-610:**
  - **Tuesdays - Thursdays** 1:30 PM – 3:00 PM
- **Section 602-611:**
  - **Tuesdays - Thursdays** 3:10 PM – 4:40 PM

My Contact

- **Email**
  - V.alizadeh@depaul.edu
- **Microsoft Team**
  - Private message
  - Or mention me in Discussion channel
- **Office**
  - #829 CDM
  - Gated community (offices behind locked doors)
  - Need to call for access (phone next to reception table)
  - Office Phone
    - 312-362-6248

Office Hours

- **Zoom Office Hours**
  - Wednesdays
    - 2:00 – 5:00 PM
    - Join URL:
      - https://depaul.zoom.us/j/7240757601?pwd=Tjl3djJ0aWY3R244R09uZ1VMWG1Fdz09
      - There is waiting room.
  - Better to schedule an appointment
    - You can contact me (email or Teams) to schedule a time.
    - Or use BlueStar
    - We can have individual or group meeting upon your request.

Course Forum: Microsoft Teams

- **Everyone is required to join the Microsoft Teams group**
- **Team Code:**

SE 350 - Spring 2021

9vz3uvy

- **Where do I enter the code?**
  - Click Join or create a team below your teams list and look for the Join a team with a code card
- **We use MS Teams as a Discussion forum, Sharing files, Announcements, Scheduling the meetings, Communications, ….**
- **Some materials and activities will be replicated in D2L as well.**

Microsoft Teams

What is this course about?

- This course studies **Principle, techniques and tools** of object-oriented **modeling**, **design**, and **implementation** of large-scale software systems. The students will learn the real-world aspects of object orientation by putting the concepts into practice.
- **Topics include:**
  - Principles of object-oriented programming languages
  - Principles of object-oriented design
  - UML diagrams (class, object, sequence,...)
  - Quality metrics and Refactoring of object-oriented design
  - Anti-patterns
  - Testing methodologies
  - Design Patterns

What is this course about?

- **Not just programming**
  - Designing programming solutions
  - What does it mean?
- **Common approaches**
  - Patterns
- **School vs. Real World**
- **Writing maintainable, extensible, high quality code**

Course Objectives & Learning Outcomes

- **Object-Oriented Design Principles**
  - You will have a deeper understanding of object-oriented concepts and how to use them, and will be able to design and develop software applications using object-oriented design principles.
- **Visual Modeling**
  - You will be able to model a software solution visually using UML sequence and class diagrams.
- **Design Patterns**
  - You will be able to design and implement an executable solution to a given problem in a programming language using the most suitable set of common software and architectural design patterns.
- **Object-Oriented Programming Paradigm Principles**
  - You will be able to effectively translate design patterns and object-oriented design principles into an object-oriented programming language.
- **Implementation**
  - You will be able to apply advanced object-oriented programming language features/capabilities to design and implement a given software application using object-oriented best practices. You will be proficient with incremental/iterative development, refactoring, and writing higher-quality codes.

Prerequisite

- **What do you need to have for this class?**
  - Understanding how to create and manage a software project.
  - **Java** and **UML** will be used for source code examples, assignments, and the exams.
  - Some experience programming in **Java** or another C-like language is required. This is NOT an introductory Java course.
- **Courses:**
  - **Required:**
  - **CSC 301** | DATA STRUCTURES II
    - (OR equivalent courses on data structures)
  - **Optional and Useful:**
  - **SE 333** | SOFTWARE TESTING
  - **SE 330** | OBJECT ORIENTED MODELING

Grading

- **Assignments (50%)**
  - **5%** - In-class and Forum Participation
  - **5%** - Quizzes
  - **40%** - Home works and Programming Projects
    - Must be submitted via D2L. No email or cloud sharing submissions is allowed.
- **Exams (50%)**
  - **20%** - Mid-term Exam
  - **30%** - Final Exam
- **Optional Bonus (10%)**
  - **10%** - Research OR Implementation Project
- **Grading Scale**

A 93-100	B- 80-82	D+ 65-68
A- 90-92	C+ 77-79	D 60-64
B+ 87-89	C 73-76	F <60
B 83-86	C- 69-72	

Tentative Schedule

Date	Activity	Lecture

Week 1
Mar30 & Apr1
	Environment Setup	• Introduction to OO
		• UML Diagrams
		• OO Principles
		  ◦ Abstraction
		  ◦ Encapsulation
		  ◦ Inheritance
		  ◦ Polymorphism
		  ◦ Composition
		  ◦ Interface
		  ◦ Delegation
		• Design with Exceptions
		• Principles and Foundations of OO Design
		  ◦ SOLID Principles
		  ▪ Single Responsibility
		  ▪ Open-Closed
		  ▪ Liskov Substitution
		  ▪ Dependency Inversion
		  ▪ Interface Segregation
		• Design Patterns
		  ◦ Creational
		  ◦ Structural
		  ◦ Behavioral
		• Anti-patterns
		• Refactoring OO Design
		• JUnit
		• OO Design & Development Research Trends

Week 2 Apr6 & Apr8		
Week 3 Apr13 & Apr15	**Assignment 1**	
Week 4 Apr20 & Apr22	**Assignment 2**	
Week 5 Apr27 & Apr29	**Mid-term Exam**	
Week 6 May4 & May6		
Week 7 May11 & May13	**Assignment 3**	
Week 8 May18 & May20		
Week 9 May25 & May27	**Assignment 4**	
Week 10 Jun1 & Jun3	**Bonus Projects Due**	
Week 11 (Finals) Jun8 & Jun10	**Final Exam**	

Important Dates

Monday April 5, 2021 Last day to add classes to SQ2021 schedule

Friday April 9, 2021 Last day to drop SQ2021 classes with no penalty

Monday April 12, 2021 Last day to select pass/fail option for SQ2021 classes

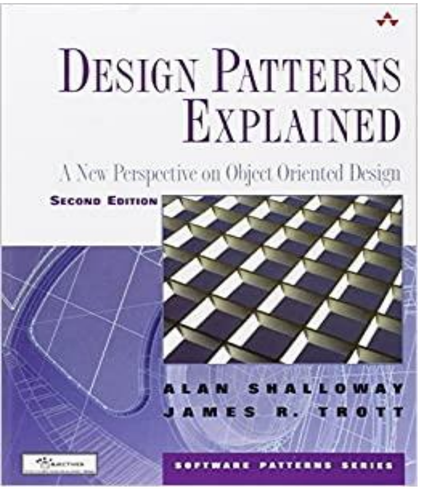
Friday May 14, 2021 Last day to withdraw from SQ2021 classes

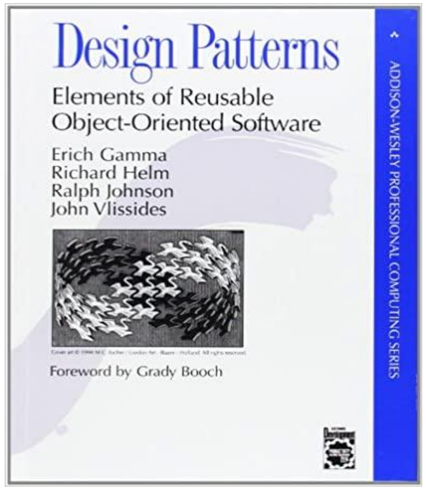
Note *Please visit DePaul Academic Calendar-Spring Term for the complete list.*

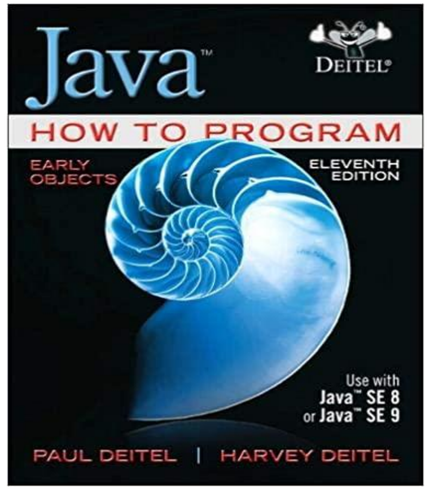
Textbooks

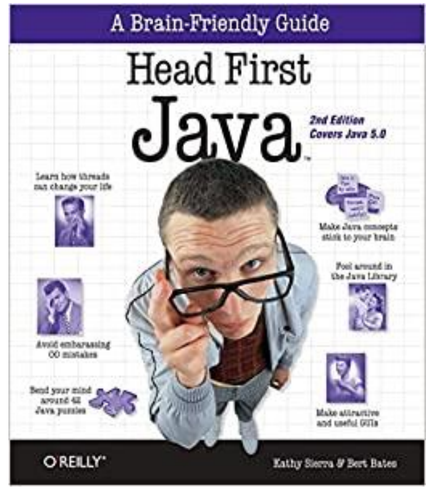
- None of the following books are required. You can use an alternate resource as a reference.

Additional course material (Articles, Lectures, Links, Videos, etc.) will be distributed in class.









Review the Syllabus Carefully

Object-oriented Software Development

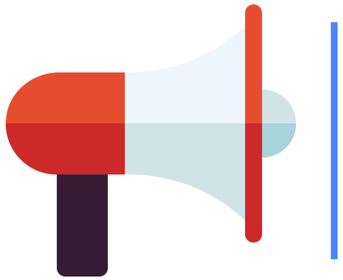
SE 350 (601-610)

Instructor: Vahid Alizadeh
Email: v.alizadeh@depaul.edu
Office: CDM 829
Office Hours: Wednesdays 2-5 PM - via Zoom / BlueStar Appointment
Lecture: Tuesdays & Thursdays, 1:30-3:00 PM - via Zoom
Course Forum: MS Teams
Quarter: Spring 2021

DEPAUL UNIVERSITY
COLLEGE OF COMPUTING AND DIGITAL MEDIA

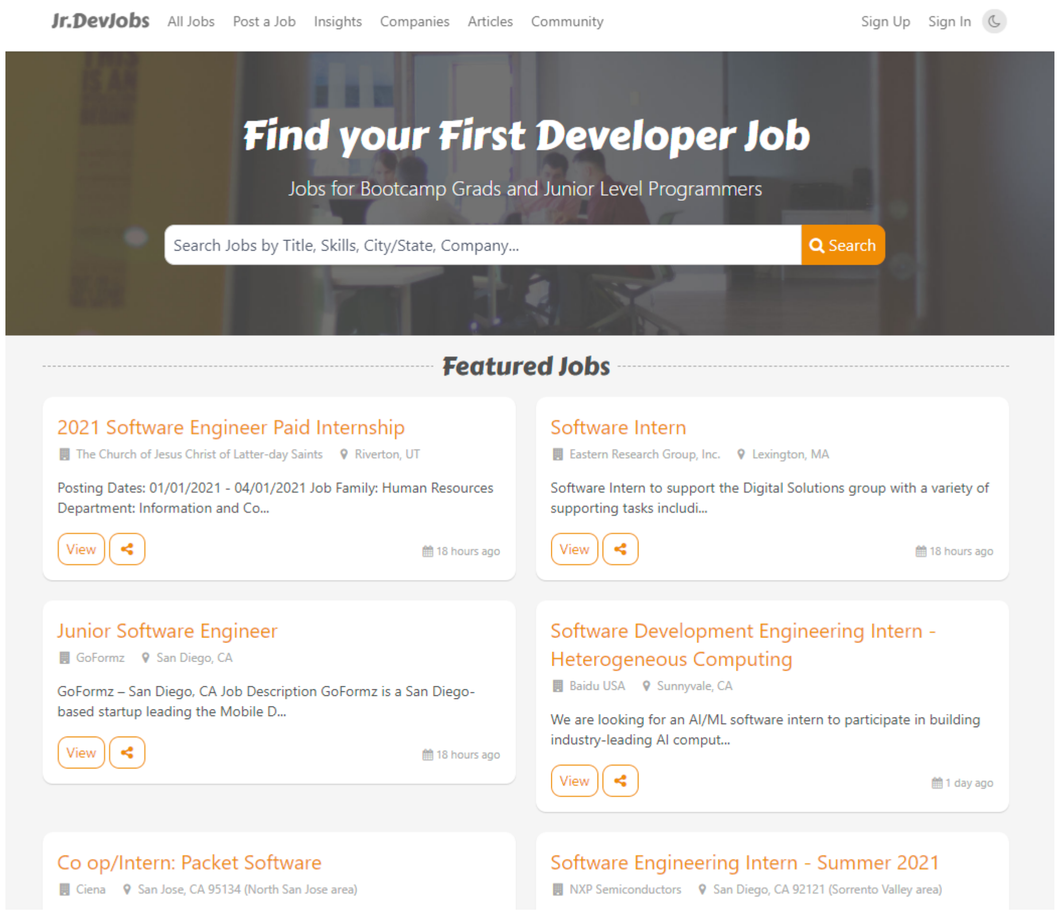
Last update: March 30, 2021

Contents



Announcements

Jobs for Bootcamp Grads and Junior Level Programmers



https://www.jrdevjobs.com/