

Lecture 01

ECE 1145: Software Construction and Evolution

Course Introduction

Software Evolution, Quality,
Maintainability (CH 1, 3, 10)

Introductions

Instructor: Dr. Kara Bocan (she/her)

Graduate Teaching Assistant: Evan McKinney

Office Hours and Contact Info

Dr. Kara Bocan

Office Hours: Tu/Th 10:00 – 11:00 AM, or by appointment

Starting Thurs. Sept. 2

1206 Benedum Hall or online (see links on Canvas)

Email: knb12@pitt.edu (include “ECE 1145” in the subject line)

Evan McKinney

Office Hours: M 9:00 – 10:00 AM

1137 Benedum Hall or online (see links on Canvas)

Email: evm33@pitt.edu

Course Structure

Lectures: M/W 3:00 – 4:15 PM

1211A/B Benedum Hall and online (see links on Canvas)

→ Recorded

→ May turn off camera/microphone, hold questions if you do not wish to be recorded

Assignments (accessible and submitted online)

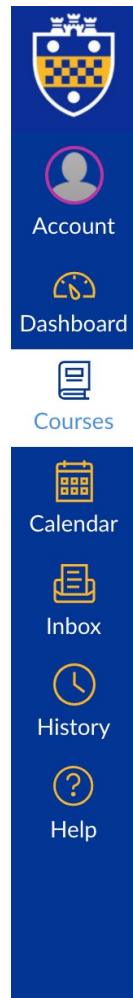
→ Weekly project iterations – submit code and report as a team (except Iteration 0)

→ Code Reviews

→ Schedule on Canvas

Canvas

canvas.pitt.edu



≡ 2221 ECE 1145 SEC1080

Fall Term 2021-2022

Home

Announcements

Modules

Assignments

Zoom

Panopto Video

Grades

Welcome to Software Construction and Evolution!

This course covers software engineering principles related to software construction, maintenance, and evolution, focusing on standards and techniques for developing maintainable, flexible software. Use the links below or in the Canvas sidebar to navigate course material. Start by reviewing the syllabus and schedule. Course materials and assignments will be posted in weekly modules.

Syllabus

Schedule

Modules

Lectures and Office Hours

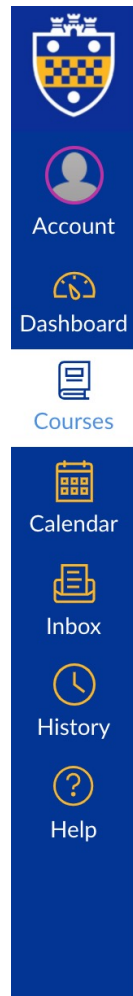
Lectures and office hours will be accessible in-person or online via Zoom. To attend via Zoom, start by [creating your university Zoom account](#). Then to connect to each meeting, click the links below, or use the Zoom meeting links in the Canvas sidebar or calendar, or go to pitt.zoom.us, click "Join", and enter the meeting information listed below. You must be signed in with your Pitt Zoom account to access the meeting.

Lectures

Canvas

canvas.pitt.edu

Home page: Syllabus, Schedule, Zoom meetings, Resources



≡ 2221 ECE 1145 SEC1080

Fall Term 2021-2022

Home

Announcements

Modules

Assignments

Zoom

Panopto Video

Grades

Welcome to Software Construction and Evolution!

This course covers software engineering principles related to software construction, maintenance, and evolution, focusing on standards and techniques for developing maintainable, flexible software. Use the links below or in the Canvas sidebar to navigate course material. Start by reviewing the syllabus and schedule. Course materials and assignments will be posted in weekly modules.

Syllabus

Schedule

Modules

Lectures and Office Hours

Lectures and office hours will be accessible in-person or online via Zoom. To attend via Zoom, start by [creating your university Zoom account](#). Then to connect to each meeting, click the links below, or use the Zoom meeting links in the Canvas sidebar or calendar, or go to pitt.zoom.us, click "Join", and enter the meeting information listed below. You must be signed in with your Pitt Zoom account to access the meeting.

Lectures

Canvas

canvas.pitt.edu

Navigation

2221 ECE 1145 SEC1080

Fall Term 2021-2022

- Home
- Announcements
- Modules
- Assignments
- Zoom
- Panopto Video
- Grades

Welcome to Software Construction and Evolution!

This course covers software engineering principles related to software construction, maintenance, and evolution, focusing on standards and techniques for developing maintainable, flexible software. Use the links below or in the Canvas sidebar to navigate course material. Start by reviewing the syllabus and schedule. Course materials and assignments will be posted in weekly modules.

[Syllabus](#) [Schedule](#) [Modules](#)

Lectures and Office Hours

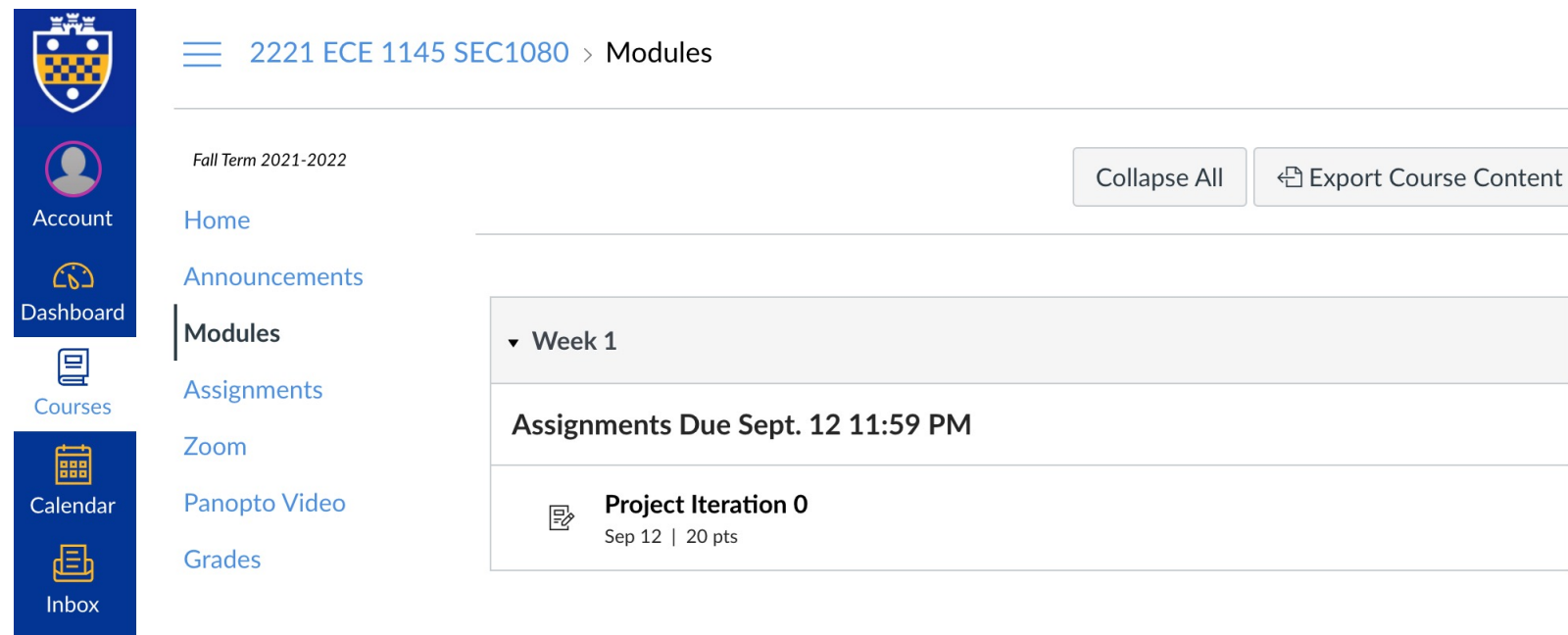
Lectures and office hours will be accessible in-person or online via Zoom. To attend via Zoom, start by [creating your university Zoom account](#). Then to connect to each meeting, click the links below, or use the Zoom meeting links in the Canvas sidebar or calendar, or go to pitt.zoom.us, click "Join", and enter the meeting information listed below. You must be signed in with your Pitt Zoom account to access the meeting.

Lectures

Canvas

canvas.pitt.edu

Weekly Modules: lecture slides and recordings, assignments (homework, projects), any other resources



The screenshot shows the Canvas LMS interface for the course 2221 ECE 1145 SEC1080. On the left is a blue sidebar with icons and labels for Account, Dashboard, Courses, Calendar, and Inbox. The main content area has a header with a hamburger menu, the course ID, and the title 'Modules'. Below the header is a section for 'Fall Term 2021-2022' with buttons for 'Collapse All' and 'Export Course Content'. A vertical list of navigation links includes Home, Announcements, Modules (which is highlighted with a vertical bar), Assignments, Zoom, Panopto Video, and Grades. The main content area displays a 'Week 1' section with a heading 'Assignments Due Sept. 12 11:59 PM'. Below this heading is a list item 'Project Iteration 0' with a document icon, due on 'Sep 12' for '20 pts'.

2221 ECE 1145 SEC1080 > Modules

Fall Term 2021-2022

Collapse All Export Course Content

Home

Announcements

Modules

Assignments

Zoom

Panopto Video

Grades

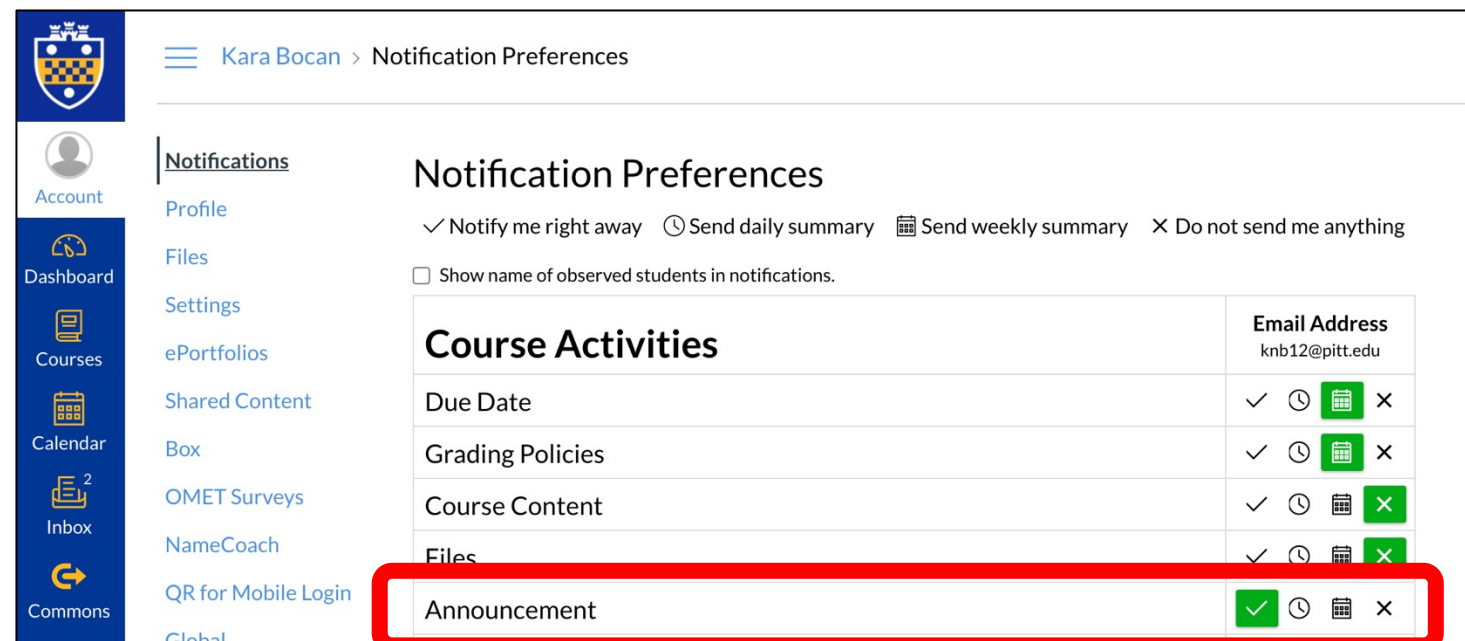
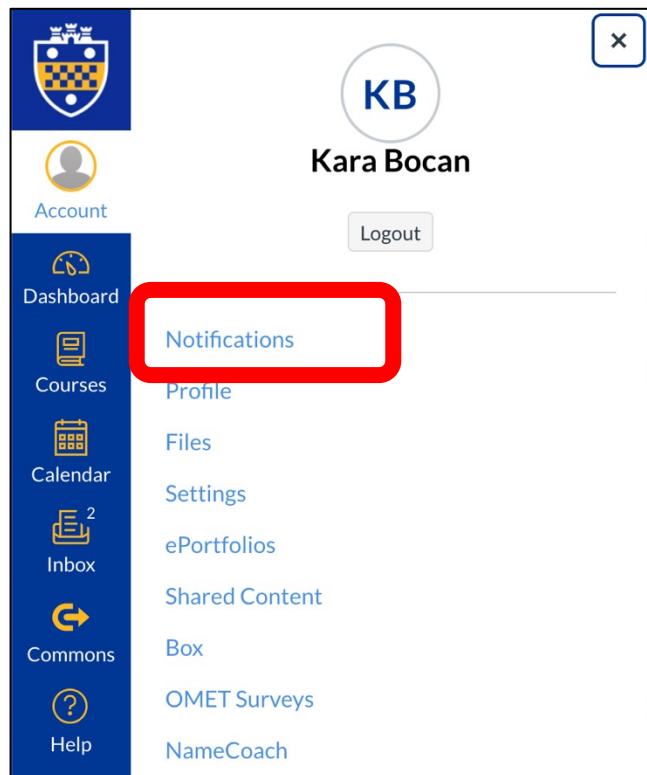
▼ Week 1

Assignments Due Sept. 12 11:59 PM

Project Iteration 0
Sep 12 | 20 pts

Canvas

Enable notifications



Communication

How will I communicate with you?

→ Lectures

→ Canvas announcements

→ Surveys

(turn on notifications!)

Kara Bocan > Notification Preferences

Notification Preferences

✓ Notify me right away ⌚ Send daily summary 📅 Send weekly summary ✕ Do not send me anything

☐ Show name of observed students in notifications.

| Course Activities | Email Address |
|-------------------|----------------|
| Due Date | knb12@pitt.edu |
| Grading Policies | knb12@pitt.edu |
| Course Content | knb12@pitt.edu |
| Files | knb12@pitt.edu |
| Announcement | knb12@pitt.edu |

Communication

How can you communicate with me/us?

→ In lectures (in-person or online)

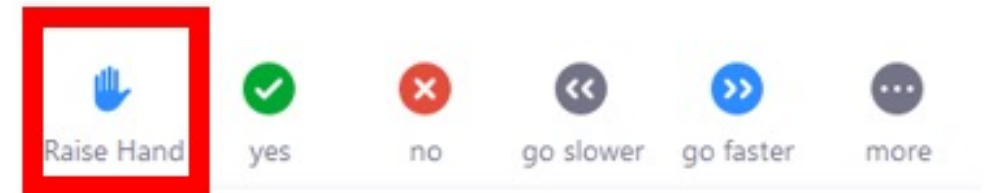
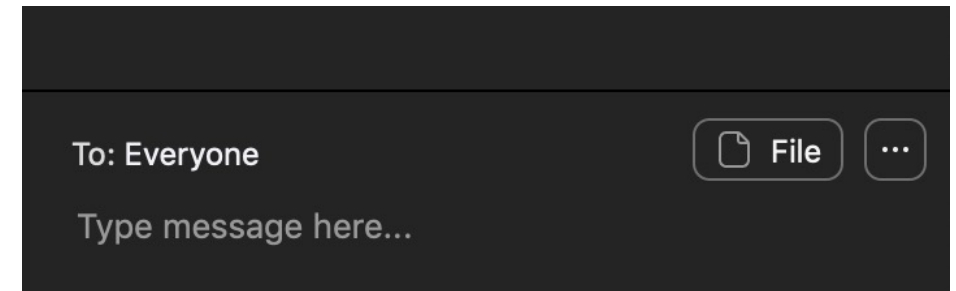
→ Chat to all or just me

→ If a private message, I will still address the question but won't say your name

→ "[Raise hand](#)" in Zoom

→ Office hours – scheduled and by appointment

→ Email – expect response same day if during business hours (8:00a – 6:00p), otherwise next day



Textbook

Flexible Reliable Software: Using Patterns and Agile Development,
Henrik B. Christensen, Andrew McGettrick, and John Impagliazzo,
2010 CRC Press LLC

Available free via Pitt library:

https://pitt.primo.exlibrisgroup.com/permalink/01PITT_INST/1sjtb5p/alma9998550973506236

→ Access link on Canvas home page

Textbook sections corresponding to lecture topics are listed on the schedule and lecture slides

Assignments and Grading

| | |
|----------------------------|-----|
| Code Reviews | 5% |
| Midterm | 12% |
| Final | 13% |
| Project (Iterations 1 – 8) | 70% |

Iterations 1, 2 : 5%

Iterations 3, 4 : 8%

Iterations 5, 6 : 10%

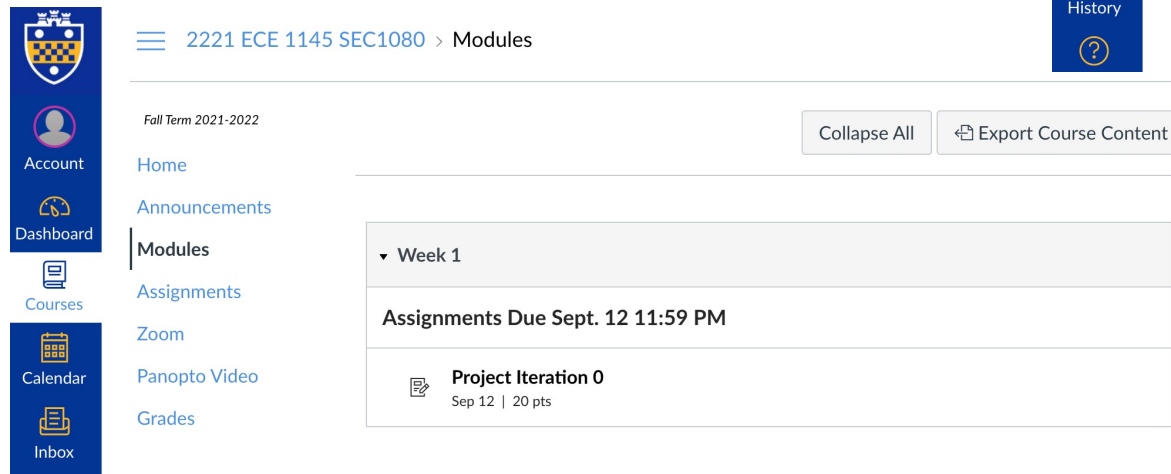
Iterations 7, 8 : 12%



Assignments

All accessible and submitted via Canvas

Due Sundays 11:59PM
(on due date)



2221 ECE 1145 SEC1080 > Modules

Fall Term 2021-2022

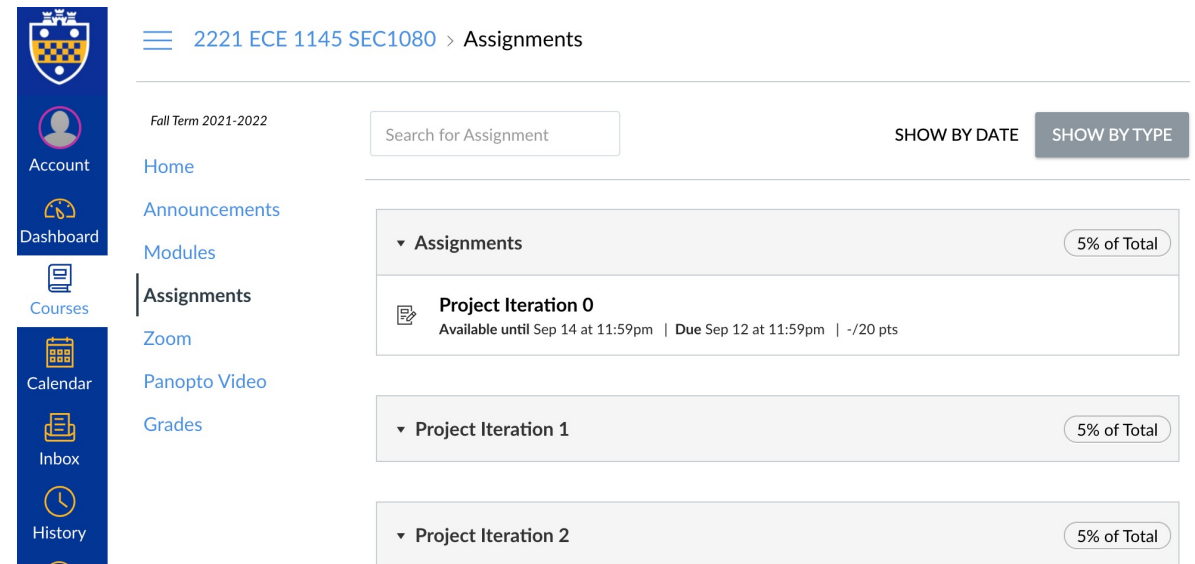
Home Announcements Modules Assignments Zoom Panopto Video Grades

Collapse All Export Course Content

▼ Week 1

Assignments Due Sept. 12 11:59 PM

Project Iteration 0
Sep 12 | 20 pts



2221 ECE 1145 SEC1080 > Assignments

Fall Term 2021-2022

Search for Assignment

SHOW BY DATE SHOW BY TYPE

▼ Assignments 5% of Total

Project Iteration 0
Available until Sep 14 at 11:59pm | Due Sep 12 at 11:59pm | -/20 pts

▼ Project Iteration 1 5% of Total

▼ Project Iteration 2 5% of Total

Late Work Policy

Late work accepted up to 48 hours after the due date

→ 30% penalty

→ Tues. 11:59pm

Extension requests granted within reason if prior to the due date

Project is iterative – iterations do not have to be perfect, just need to demonstrate the learning goal for that iteration.

→ Better to turn in working code and then make improvements / add missing features on the next iteration.

Project Iteration 0

Due Sunday Sept. 12, 11:59PM

- Set up development environment, practice Java, submit practice report with screencast, play a turn-based strategy game
- Submit reports via Canvas
 - Use provided templates for reports, submit via Canvas
 - Complete digitally or print and scan as pdf
 - [How to Scan Work on a Mobile Device](#)
 - Code on private GitHub repository

Collaboration Policy

For project iterations, work with your teammates

You may discuss with others **generally** how to approach problems / debug, but your team must do your own work

- **Practice learning, problem solving, and debugging** – these skills are sometimes more important than the actual class material!
- In reports, cite references and others involved in discussions

When in doubt, ask us!

Schedule

Link on Canvas
Home Page

Will be updated as
needed

ECE 1145 Fall 2021

Last Updated:

8/25/21

| Week | Date | # | Lecture Topic | Text Reference | Assignments Due |
|------|-------|----|--|--------------------|---|
| 1 | 8/30 | 1 | Course Intro, Software Evolution, Quality, Maintainability | CH 1, 3, 10 | |
| | 9/1 | 2 | Course Project Overview, Tools and Environment | CH 36.1, 36.2 | |
| 2 | 9/6 | | No Class - Labor Day | | PI0: Environment Setup |
| | 9/8 | 3 | Test-Driven Development | CH 2, 4, 5 | |
| 3 | 9/13 | 4 | Configuration Management | CH 33 | PI1: Test-Driven Development |
| | 9/15 | 5 | Build and Release Management | CH 6 | |
| 4 | 9/20 | 6 | Design Patterns Overview, Strategy Pattern | CH 7, 8, 9 | PI2: TDD and Git |
| | 9/22 | 7 | Refactoring and Integration Testing | CH 8 | |
| 5 | 9/27 | 8 | Coding Standards, Documentation, Maintainability | CH 3, 10 | PI3: Strategy Pattern and Refactoring |
| | 9/29 | 9 | Code Review | | |
| 6 | 10/4 | 10 | Variability Management, State Pattern, Test Stubs | CH 11, 12 | Code Review 1 |
| | 10/6 | 11 | Time In-Class for Code Review | | |
| 7 | 10/11 | 12 | Abstract Factory Pattern, Pattern Fragility | CH 13, 14 | PI4: Code Quality |
| | 10/13 | 13 | Midterm Review | | |
| 8 | 10/18 | | Midterm | | Midterm |
| | 10/20 | 14 | Compositional Design, Roles and Responsibility | CH 15, 16 | |
| 9 | 10/25 | 15 | Multi-Dimensional Variance | CH 17, 18 | PI5: Test Stubs and More Patterns |
| | 10/27 | 16 | Design Pattern Catalog | CH 19 - 23, 25, 27 | |
| 10 | 11/1 | 17 | Systematic Testing | CH 34 | PI6: Compositional Design |
| | 11/3 | 18 | Code Coverage | | |
| 11 | 11/8 | 19 | More Patterns | CH 26, 28, 29, 31 | PI7: Blackbox Testing and Pattern Hunting |
| | 11/10 | 20 | Framework Theory | CH 32 | |
| 12 | 11/15 | 21 | MiniDraw | CH 30 | Code Review 2 |
| | 11/17 | 22 | HotCiv GUI | CH 36.7 | |
| 13 | 11/22 | | No Class - Thanksgiving Recess | | |
| | 11/24 | | No Class - Thanksgiving Recess | | |
| 14 | 11/29 | 23 | Exception Handling | | |
| | 12/1 | 24 | TBD | | |
| 15 | 12/6 | 25 | TBD | | PI8: Frameworks |
| | 12/8 | 26 | Final Review | | |
| 16 | 12/13 | | No Class - Finals Week | | Final |
| | 12/15 | | No Class - Finals Week | | |

Exams

Open book and notes

Small number of short answer questions, open-ended coding problems, project reflection

Studying recommendations:

- Review lectures, try example problems on your own, ask questions
- Review project iterations
- Make a summary sheet to organize concepts
- Use your resources! We are here to help

COVID-19 Safety Guidelines

coronavirus.pitt.edu for up-to-date information

If in-person: face covering required

→ **Do not attend in-person if you feel sick!** We will allow you to make up the work.

If you get sick:

→ [Contact Student Health Services](#) for instructions

→ Contact me – your health is important, we will work something out

→ Continue remotely if possible, or arrange to complete coursework later

Accessibility and Inclusivity

If there is any way we can make this course more accessible or the environment more inclusive, please let us know!

- Contact [Disability Resources and Services](#) (DRS) to request accommodations:
140 William Pitt Union, (412) 648-7890, drsrecep@pitt.edu,
(412) 228-5347 for P3 ASL users
- Contact us with any needed adjustments for religious holidays or cultural practices

If you experience anything that makes this learning environment unsafe for you, let us know

- Our job is to promote an environment that supports everyone in their learning

Support and Counseling Resources

[University Counseling Center](#), telehealth appointments available

[Therapy Assistance Online](#), a private online library of programs

[Thriving Campus](#), a directory of off-campus licensed mental health clinicians (many specialize in working with students)

[resolve Crisis Services](#), call any time 1-888-7-YOU-CAN (1-888-796-8226)

- For information and non-urgent matters, call 412-864-5004 or [email resolve Crisis Services](#).
- Walk-in Crisis Center at 333 North Braddock Ave., Pittsburgh, PA 15208

Questions for Today

What is software evolution?

What is software quality? Why is it important?

How can we design flexible software?

What are we doing here, exactly?

- This is a (relatively) new course, adapted from:
 - Software Engineering and Architecture: <https://baerbak.cs.au.dk/c/swea/menu1.html>
 - Software Construction and Evolution: <https://www.rose-hulman.edu/class/csse/csse375/>
- Focus: Software Evolution and the importance of Software Quality, Flexibility, Maintainability
 - Iterative development, refactoring, test-driven development, patterns
- Feedback is appreciated!

Software Development Concepts

Requirements

- Collect and document user's/customer's expectations for the software

Design

- Partition and structure the software to make it more communicable

Realization

- Program the software to fulfill requirements while adhering to design

Testing/Quality Assurance

- Verify that the system conforms to requirements and expectations

Deployment

- Ensure that the software executes correctly at the user's location/environment

Maintenance

- Ensure product is corrected and enhanced in response to bug reports or feature requests



Software Development Concepts

Requirements

- Collect and document user's/customer's expectations for the software

Design

- Partition and structure the software to make it more communicable

Realization

- Program the software to fulfill requirements while adhering to design

Testing/Quality Assurance

- Verify that the system conforms to requirements and expectations

Deployment

- Ensure that the software executes correctly at the user's location/environment

Maintenance

- Ensure product is corrected and enhanced in response to bug reports or feature requests



How are these activities organized in a typical programming course?

How are these activities organized in a company?

Course Applications

In-class examples: Pay station



<http://pittsburghparking.com/>

Project: Turn-based strategy game
“HotCiv”



Software Construction and Evolution

Building vs. Growing



Software Construction and Evolution

From the *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 2004 edition:

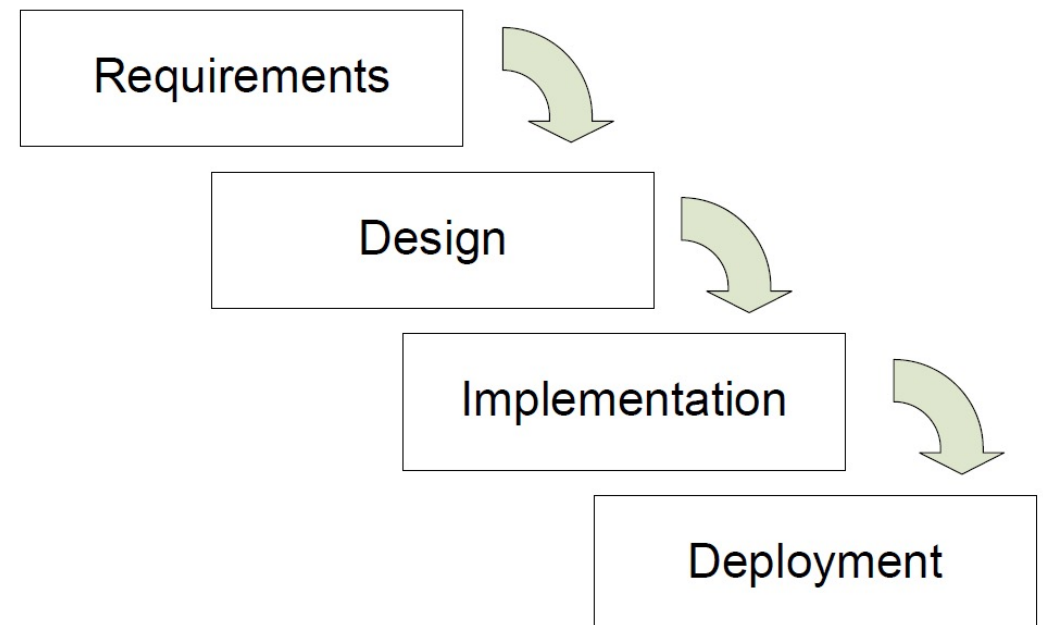
Software construction refers to the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

Software Construction and Evolution

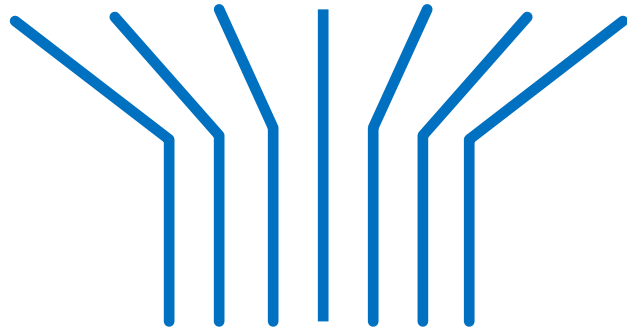
From the *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 2004 edition:

Software construction refers to the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

Waterfall model



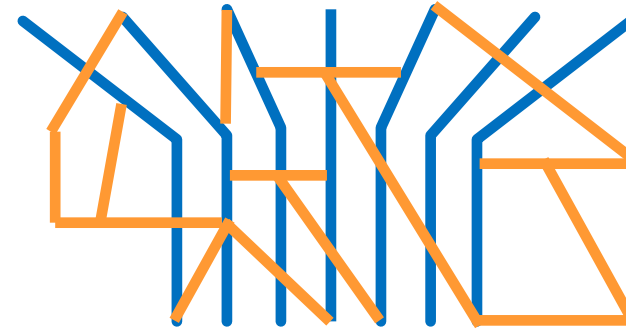
Software Evolution



The Original Software Design...

Easy to understand

Well-isolated components are easy to change, easy to validate changes



...after a few Updates

Increased size and complexity
...but it works, right? (for a while)

Reliability degrades, errors creep in...

At some point, it becomes
unmaintainable – cost to make the next
change becomes too high

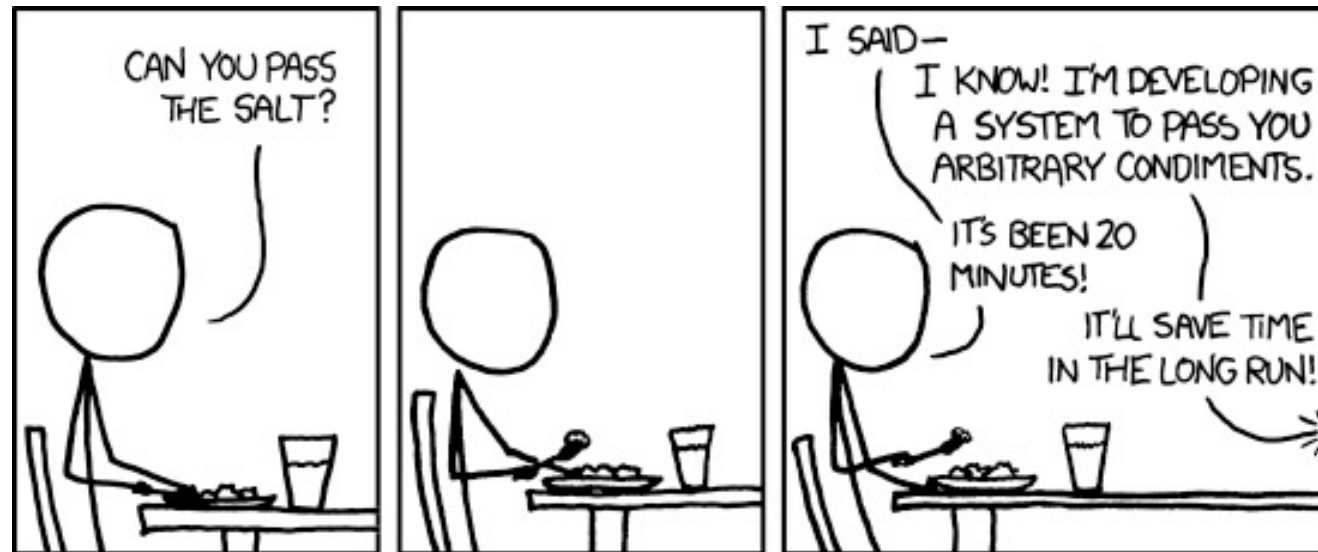
Software Evolution

“Growing” software does not mean we can’t have a plan or design
It means we are **agile** and prepared to change the plan (even dramatically) as we **learn from growing the software!**



Why Change?

Who/what drives changes in software?



Why Change?

Who/what drives changes in software?

- Users/clients, management, developers, government
- Add functionality, bug fixes, updated assumptions, process change, code clean-up, improve usability, generalize, improve performance, adapt to new technology...

Software Development: Agile

Manifesto for Agile Software Development

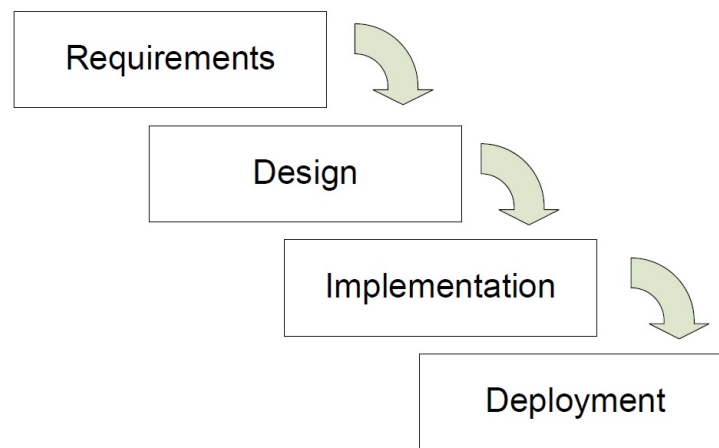
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

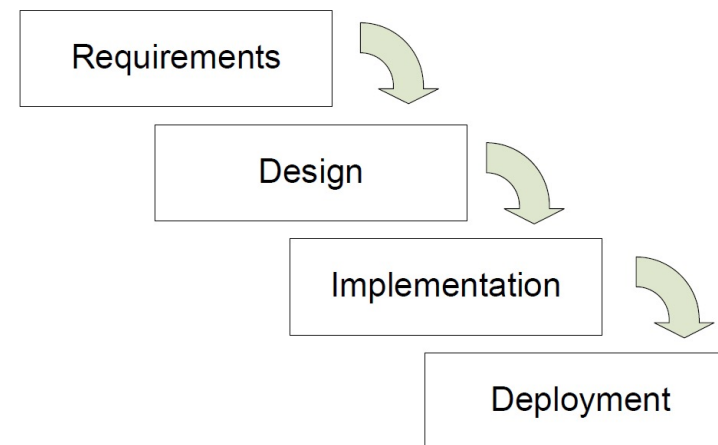
Extreme Programming (XP)

Waterfall



1-3 years

XP



Extreme Programming

Pair Programming

Roles: programmer and overseer

Collective code ownership (!= no ownership)

Coding standards

Automated testing

Test-driven development (to ensure tests are written)

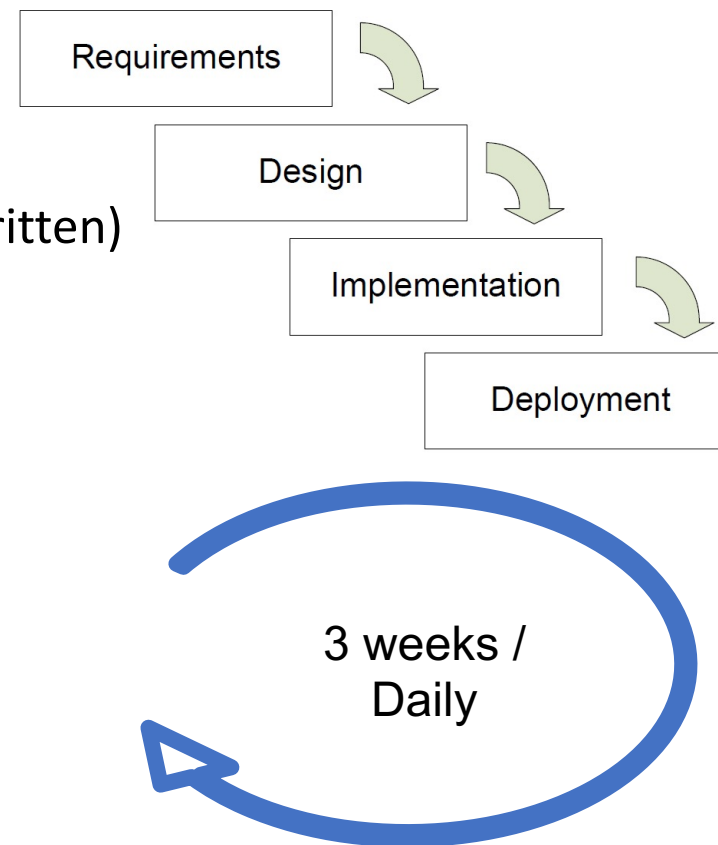
Small releases (Scrum: Sprints)

Continuous integration (“nightly builds”)

The process is

→ Incremental: **growing** software

→ Iterative: learning while doing



Extreme Programming

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Pair Programming

Roles: programmer and overseer

Collective code ownership (!= no ownership)

Coding standards

Automated testing

Test-driven development (to ensure tests are written)

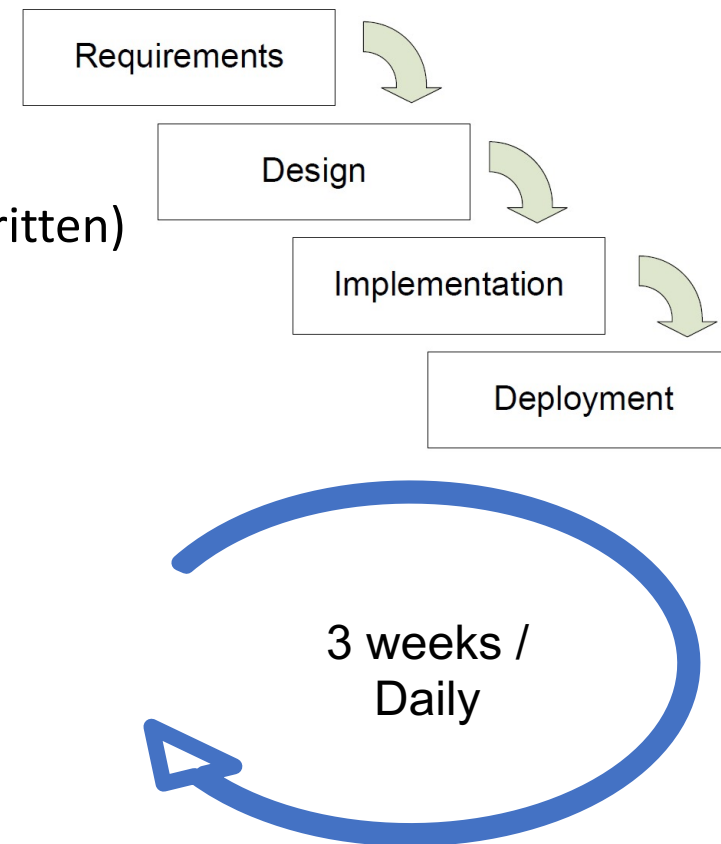
Small releases (Scrum: Sprints)

Continuous integration (“nightly builds”)

The process is

→ Incremental: **growing** software

→ Iterative: learning while doing



Emphasis on effective and efficient **communication**

Simplicity and control of scope

Feedback to keep focus and track new features

Courage to act appropriately when things aren't working or aren't needed!

Software Quality

“Good” code is **easier to change**

→ What does “good” mean?

Example: Is this *good* or *bad* software?

1. Argue that this is good software!

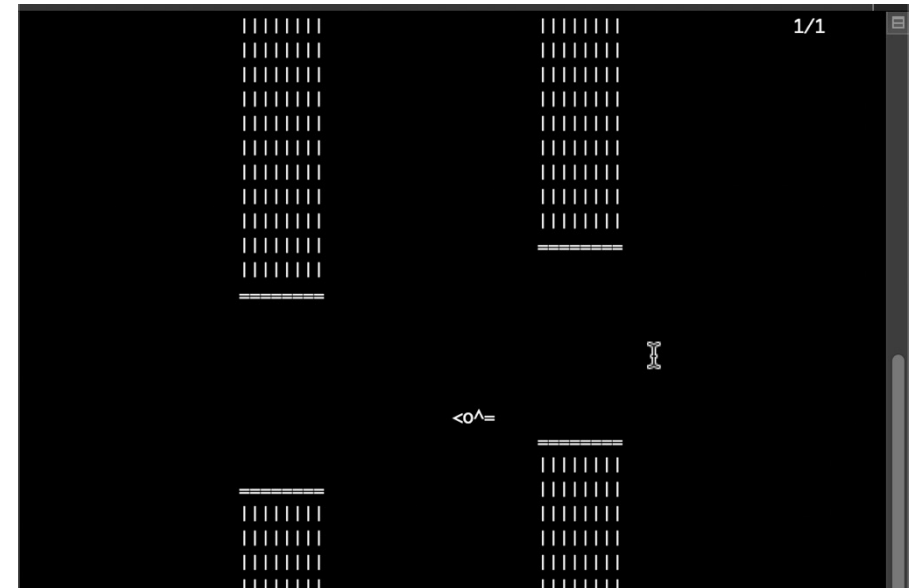
2. Argue that this is bad software!

```
#define P(a,b,c) a##b##c
#include/*****<urses.h>
int c,h, v,x,y,s, i,b; int
main() { initscr( ); P(cb,
rea, k)() ;///
P(n, oec, ho)(
)/* */ ;for (curs_set(0); s= x=COLS/2
; P( flu, shi, np)()){ timeout(y=c= v=0);///
P(c, lea, r)() ;for (P (
mva, d, dstr )(2, 3+x,
G) ; ; P( usl, eep, )(U)){//
P(m, vad, dstr )( y >>8,x,///
" "); for(i=LINES; /* */ i
; mvinsch(i,0,0>(~c|i-h-H &h-i )?' '
:(i- h|h- i+H) <0?'|' : '=' ));
if(( i=( y +=v= getch( )>0?I:v+
A)>>8)>=LINES| mvinch(i*= 0<i,
!=mvinch(i,3+x))break/*&% &*/;
>>8, x,0>v ?F:B ); i--s
/-W; P(m, vpr, intw)(0,
COLS-9," %u/%u ",(0<i)*
b); refresh(); if(++ i,b=b<i?i:
c==D){ c
-=W; h=rand()%(LINES-H-6
)+2; } } flash(); }}
```

Software Quality

The International Obfuscated C Code Contest

<https://www.ioccc.org/winners.html>

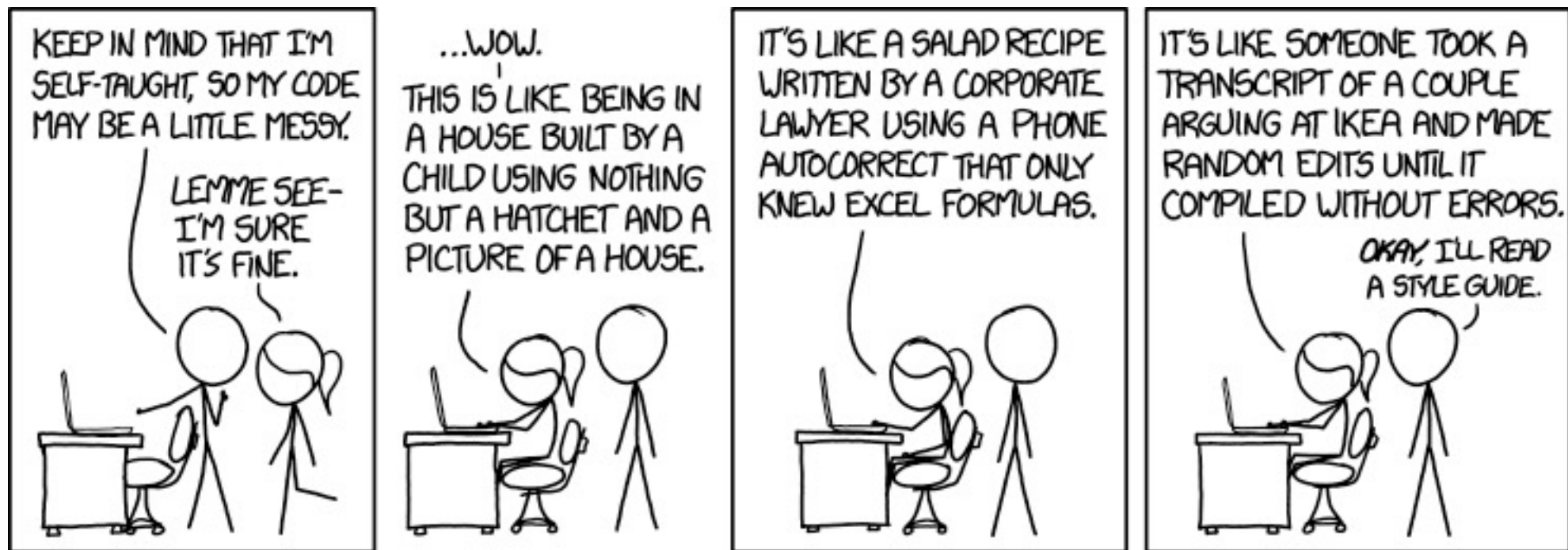


“Tappy Bird”

```
#define P(a,b,c) a##b##c
#include/*****<urses.h>
int c,h, v,x,y,s, i,b; int
main () { initscr( ); P(cb,
    rea, k()) ;///
    P(n, oec, ho)(
    )/* */ ;for (curs_set(0); s= x=COLS/2
    ; P( flu, shi, np()){ timeout(y=c= v=0);///
    P(c, lea, r)() ;for (P (
    mva, d, dstr )(2, 3+x,
    G) ; ; P( usl, eep, )(U)){//
    P(m, vad, dstr )( y >>8,x,//
    " "); for(i=LINES; /* */ i
    ; mvinsch(i,0,0>(~c|i-h-H &h-i
    :(i- h|h- i+H) <0?'|' : '=' ));
    if(( i=( y +=v= getch( )>0?I:v+
    A)>>8)>=LINES||mvinsch(i*= 0<i,
    !=mvinsch(i,3+x))break/*&% &*/;
    >>8, x,0>v ?F:B ); i--s
    /-W; P(m, vpr, intw)(0,
    COLS-9," %u/%u ",(0<i)*
    b); refresh(); if(++ i,b=b<i?i:
    c==D){ c
    -=W; h=rand()%(LINES-H-6
    )+2; } } flash(); }}
```


Software Quality Metrics

What are some measures of software quality?



Software Quality Metrics

1991 - 2011

ISO 9126:

Usability
Functionality
Efficiency
Portability
Reliability
Maintainability



https://en.wikipedia.org/wiki/ISO/IEC_9126

Software Quality Metrics

2011 - Present

ISO 9126:

Usability
Functionality
Efficiency
Portability
Reliability
Maintainability

ISO 25010:

Usability
Functional Suitability
Performance Efficiency
Portability
Reliability
Maintainability

Security
Compatibility

SOFTWARE PRODUCT QUALITY

Functional Suitability

- Functional Completeness
- Functional Correctness
- Functional Appropriateness

iso25000.com

Performance Efficiency

- Time Behaviour
- Resource Utilization
- Capacity

Compatibility

- Co-existence
- Interoperability

Usability

- Appropriateness
- Recognizability
- Learnability
- Operability
- User Error Protection
- User Interface Aesthetics
- Accessibility

Reliability

- Maturity
- Availability
- Fault Tolerance
- Recoverability

Security

- Confidentiality
- Integrity
- Non-repudiation
- Authenticity
- Accountability

Maintainability

- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

Portability

- Adaptability
- Installability
- Replaceability

Software Quality Metrics

Many qualities are in direct conflict

→ They must be balanced!

Modifiability vs. performance

→ Delegation adds costs in execution speed and memory footprint

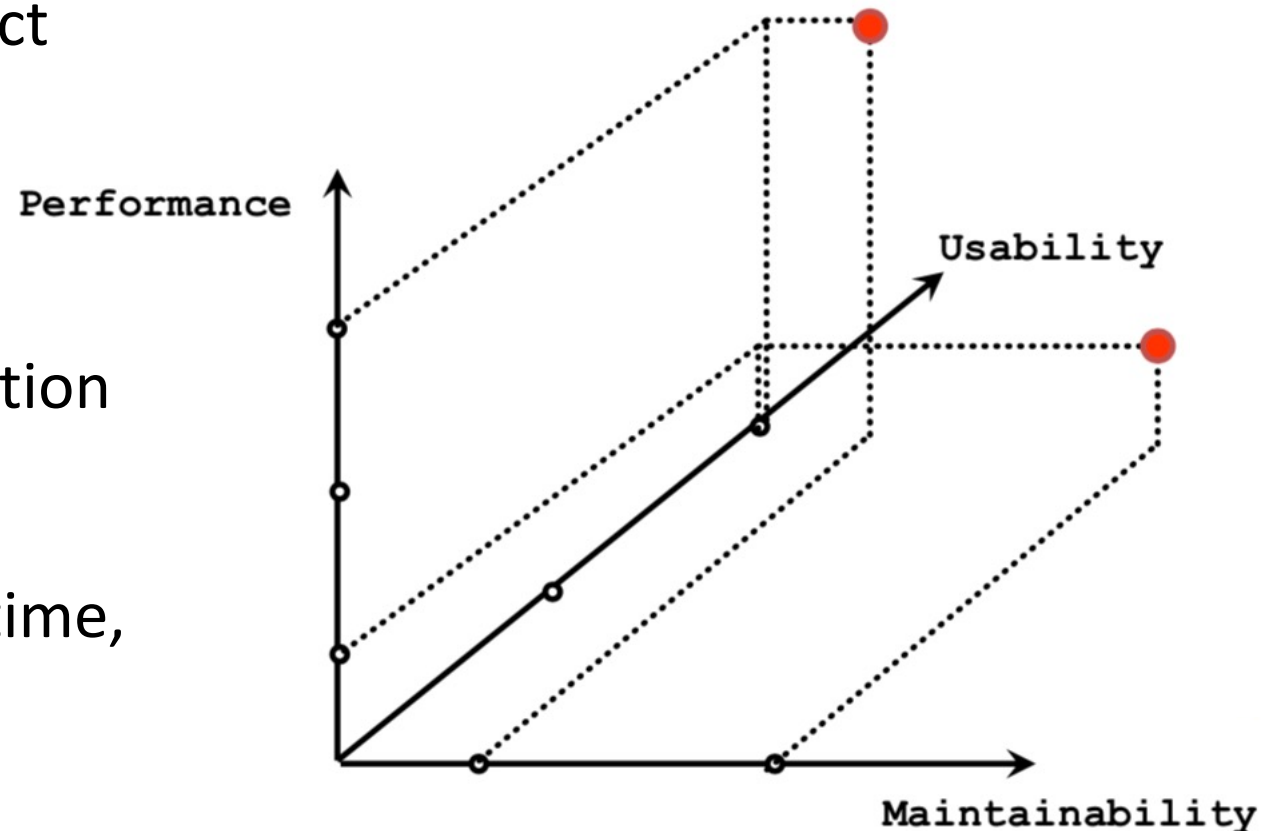
Cost vs. reusability

→ Highly flexible software takes time, effort, and money

Security vs. availability

→ Availability through redundancy increases opportunities of attack

etc.



Different priorities for developers, managers, customers...

Software Quality Metrics



Software Quality Metrics

Definition: Reliability (ISO 9126)

The capability of the software product to maintain a specified level of performance when used under specified conditions.

Definition: Maintainability (ISO 9126)

The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

Software Quality Metrics

```
public class X{private int y;public X(){y = 0;}public int z(){  
return y;}public void z1(int z0){y += z0;}public static void main(  
String[] args){X y=new X();y.z1(200);y.z1(3400);System.out.println  
("Result is "+ y.z());}}
```

Customers / users of software likely don't care if the code is readable, understandable, well documented... as long as it **works**.

→ Reliability

Code quality is usually most important to **developers** to ensure **maintainability**.

→ Although, good code also tends to have fewer bugs, because it is more **testable** (and the bugs are more fixable, due to **analyzability** and **modifiability**) ... more later

Course Project Iterations

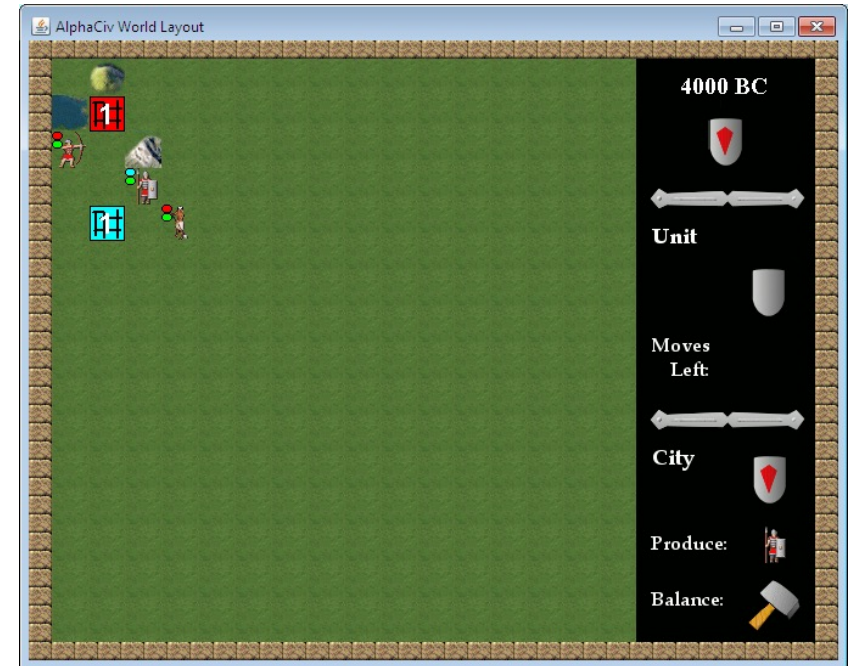
In this course, we will practice iterative development of reliable, maintainable (and flexible) software

... by developing a turn-based strategy game (iteratively)!



Course Project Iterations

- 0: Development Environment
- 1: Test-Driven Development (TDD)
- 2: TDD and Git
- 3: Strategy Pattern, Refactoring
- 4: Code Quality (w/ Code Review)
- 5: Test Stubs, More Patterns
- 6: Compositional Design
- 7: Blackbox Testing, Pattern Hunting
- 8: Frameworks and MiniDraw



Course Project Iterations

0: Development Environment – Due Sept. 12 11:59 PM

- Set up Java, Gradle, Git, IntelliJ
- Get the project starter code
- Practice Java
- Practice preparing/submitting a report
- Play a (turn-based strategy) video game!
- Read 36.1, 36.2 to prepare for Iteration 1

