Software Engineering

Introduction

Duke
UNIVERSITY

Welcome to 651: Software Engineering

- Professors
  - Section 1: Dr. Drew Hilton
  - Section 2: Dr. Shani Daily
- TAs:
  - Siyu Chen
  - Guanwen (Gary) Wang
  - Rijish Ganguly
  - Mengxi Wu
  - Liming Jin
  - Rui Sun

Duke UNIVERSITY

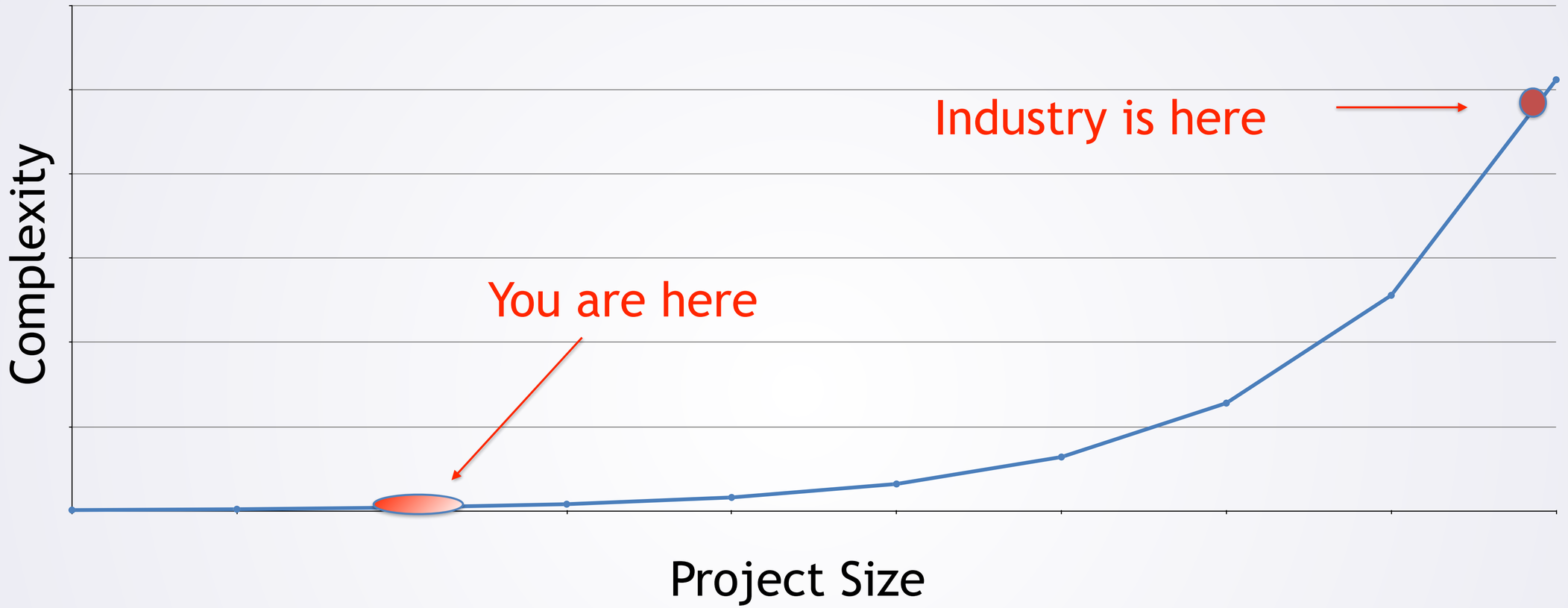
What is Software Engineering?

- Software Engineering is about **Managing Complexity**
  - Then again, so is pretty much everything in Computer Engineering..

What is Software Engineering?

- Current skill set: small projects, low complexity, one developer
  - A few classes with simple interfaces
  - Working from well-defined specs, often with design given

What is Software Engineering?



- Need to handle **a couple orders of magnitude** more complexity
  - Much larger projects, many developers
  - Specifications need refinement, must do significant design

Complexity: Big Hammer

Complexity

- Fortunately 551 + 550 have given you a big hammer to attack complexity…
  - Remind me what it is called?

Complexity: Big Hammer

Complexity

- Fortunately 551 + 550 have given you a big hammer to attack complexity…
  - Remind me what it is called? **Abstraction**

Abstraction

- Break big problems into small problems
  - Separate interface from implementation
- Tools you are familiar with for this:
  - Functions
  - Classes
- Now need to think about how to break large problems down
  - Into many classes
  - Possibly multiple programs (maybe on multiple computers)
    - May communicate by things other than function call (e.g., http)

OO Design

- One major topic of this course: OO Design
  - How do we split the task into (good) classes?
  - What are the interfaces between classes?
  - How do we make the project resilient to changes?
    - Real code changes.
    - Change is hard
    - Most reasons for what makes good vs bad code is change
- …but design is not the only aspect of software engineering…

Facets of Software Engineering

- Requirements Definition
- Design
- Implementation
- Testing
- Maintenance
- Working in Teams
- Process/Project Management

Facets of Software Engineering

- Requirements Definition
- Design
- Implementation
- Testing
- Maintenance
- Working in Teams
- Process/Project Management

I'm going to overview each briefly
As I do so, I want you all to think
about how **abstraction helps complexity**
in each topic.

Requirements Definition

- Customers often have a **vague** idea of what sw should do
  - "I need a program that lets students register for courses"
- However, you need a very **specific** specification with details
  - Should it be a web app? Mobile?
  - What rules does it need to enforce?
  - How does it handle full classes?
  - …

Design

- Design: determining what the pieces are and what they do
  - Pieces may be..
    - Services/programs
    - Classes
    - Functions
- Hierarchy: (also popular in 550, right?)
  - May do high level design (HLD) then refine
    - Split into services now, then design each of those
- Key: getting the right interfaces!
- Note: design does not generally involve writing code!

Implementation

- Implementation: given a small enough"piece" make it work
  - This is what you all are good at from 551
  - Here is where you write code.
- Piece too large? Refine design
  - Break into more pieces

Testing

- Remind us about testing from 551?

Testing

- Find **presence** of bugs.
  - Become more confident that software is correct as bugs harder to find.
- In 551, you did **unit testing**
  - Testing individual functions/classes
- Other kinds of testing we'll learn about
  - **Regression testing**: did you break it with this change?
  - **Integration testing**: do the pieces fit together?
  - **System testing**: does the whole thing work?
  - **Acceptance testing**: should the customer say "you are done"?

Maintenance

- After we are "done" we aren't really done.
- Changes, monitoring, and support after "done" are maintenance
  - Bug fixes
  - New features
  - Changes to how features should work
  - Monitoring behavior
  - Recovering from outages
  - …

Working In Teams

- So far: develop individually
- Real software: 10s to 100s (or 1000s..) of developers
  - 15,600 developers have contributed to Linux since 2005.
  - Internet estimates about 1000 developers on Windows 7.
- How do you work on a team of 20? 100? 500? 1000?

Process/Project Management

- Need to not just make software…
  - But make it **on time**
  - And correct.
- What process do you follow to get all this stuff done?
  - Especially with your team of 100 people…
- We'll talk about some common models, e.g.
  - Waterfall
  - Agile

Facets of Software Engineering

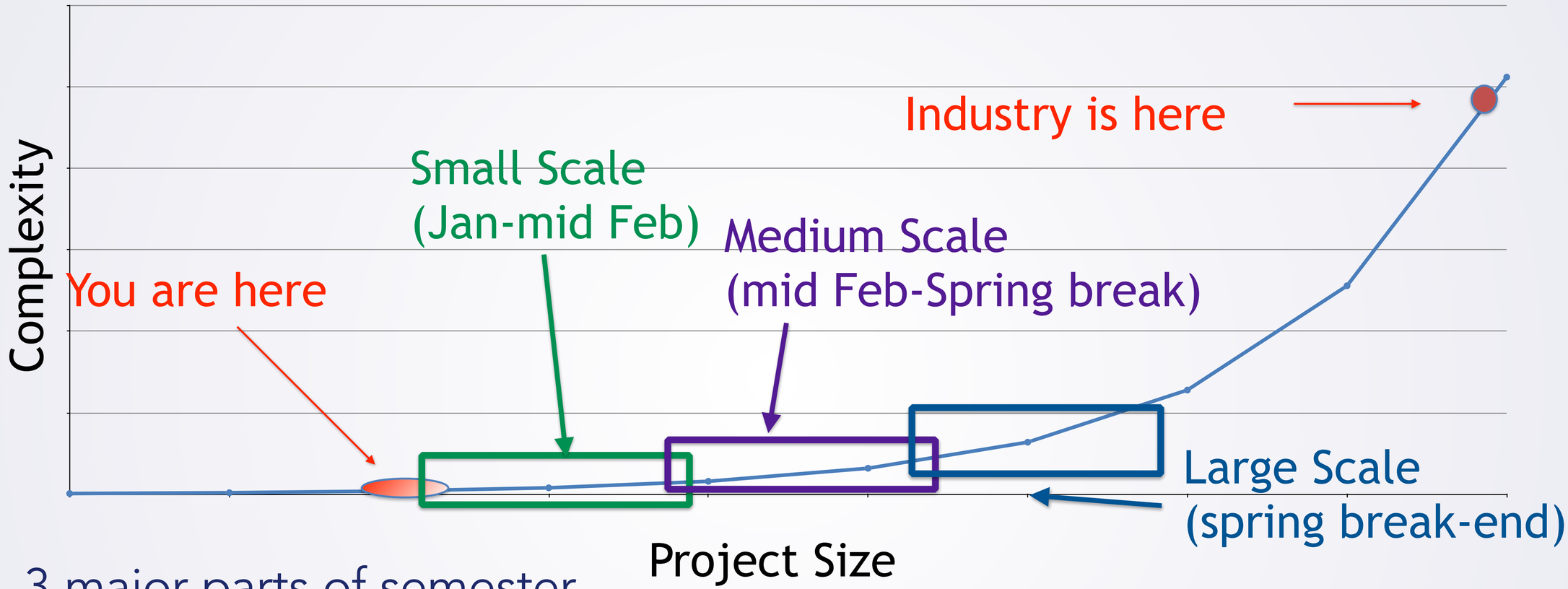
- Requirements Definition
- Design
- Implementation
- Testing
- Maintenance
- Working in Teams
- Process/Project Management

Think, pair, share!

You all thought about how abstraction helps in each, discuss your thoughts with the person next to you.

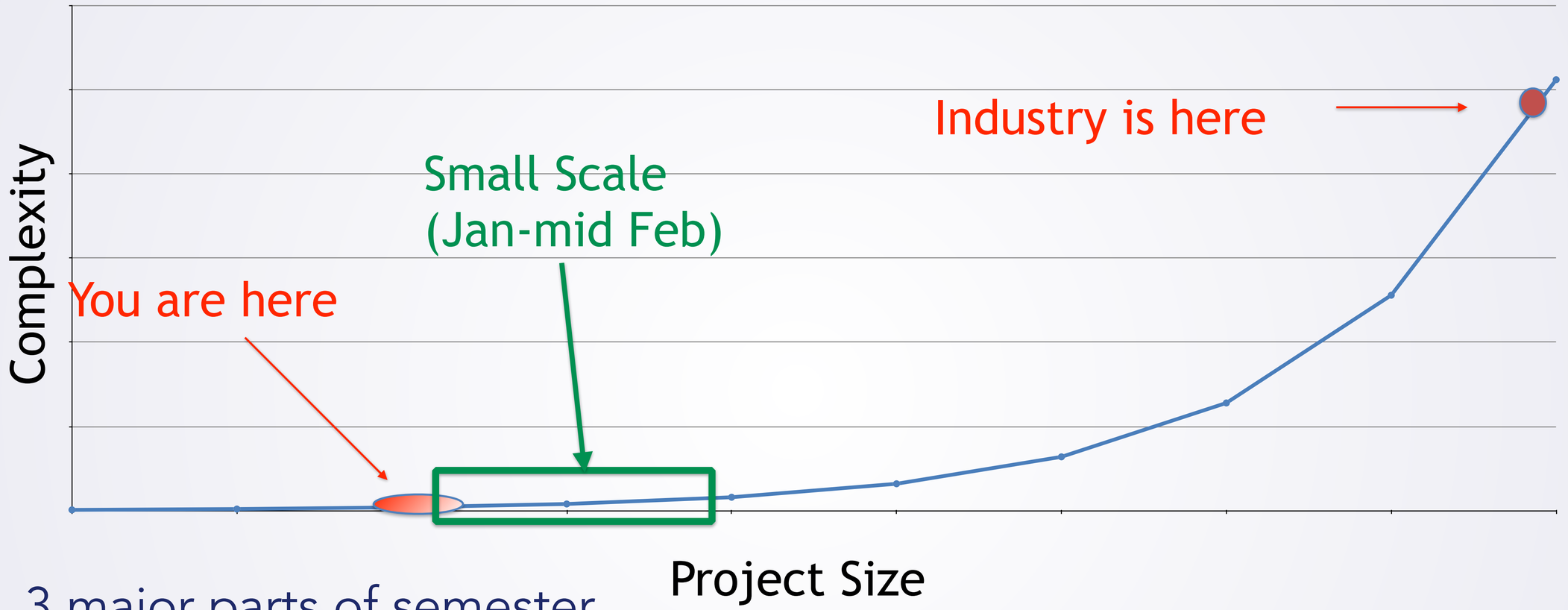
In a few minutes, we'll have people report back…

Roadmap



- 3 major parts of semester
  - Small Scale: A few classes (1-~10)
  - Medium Scale: Modules: many classes (10+)
  - Large Scale: Systems: multiple components/programs interacting

Roadmap



- 3 major parts of semester
  - Small Scale: A few classes (1-~10)
  - Medium Scale: Modules: many classes (10+)
  - Large Scale: Systems: multiple components/programs interacting

Roadmap From Here

- First: **key principles**
  - What guides our design?
  - How do we know if something is good or bad?
- These will underpin everything else we do
  - They are your **vocabulary** for **discussing** software engineering ideas
  - Discussion is key. I expect you all to talk
    - Why? **Think pair share**…

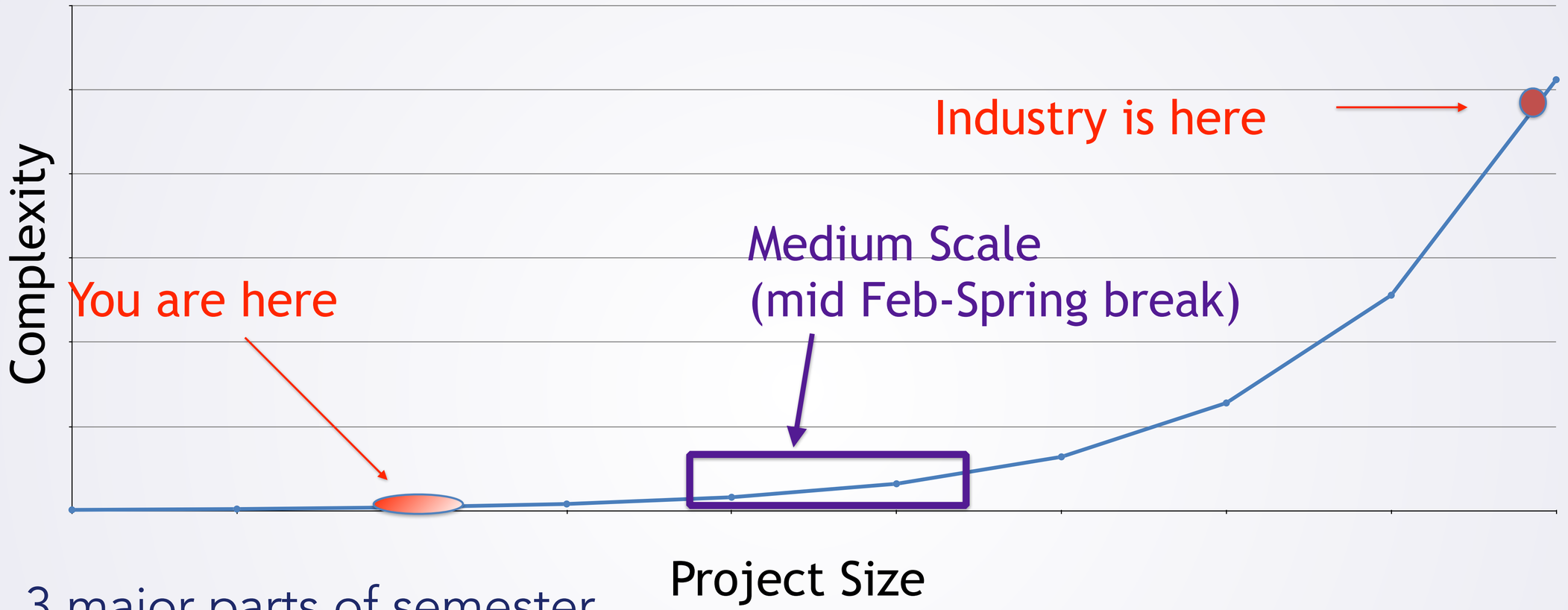
Discussing Software Engineering

- This is a key skill for your jobs
  - Advocate for your design. Give feedback on your co-worker's
  - Interview? Design questions…
- Analytical skills -> deep understanding
  - Nothing in CE is about memorization
  - Deep understanding: how, why?
  - Contemplate new things never seen before

Roadmap Cont'd

- **After principles**: Java for C++ programmers
  - Analyze language differences in framework of our design principles
  - Java came after C++
    - Why did they consider changes an improvement?
- **Then**: "small scale" software engineering
  - Process/project management
  - Design (especially design patterns)
  - Quality (testing, code review, technical debt, refactoring

Roadmap

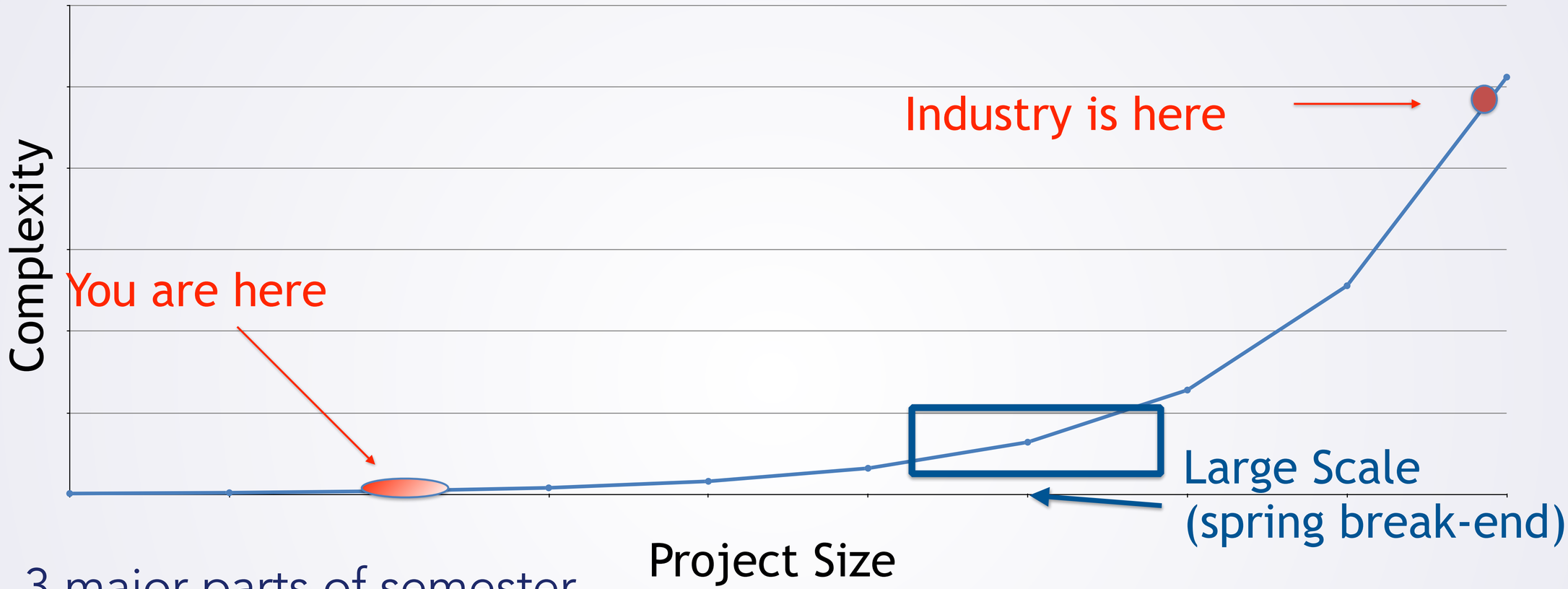


- 3 major parts of semester
  - Small Scale: A few classes (1-~10)
  - Medium Scale: Modules: many classes (10+)
  - Large Scale: Systems: multiple components/programs interacting

Roadmap Cont'd

- **After that**: "medium scale" software engineering
  - Process revisited
    - Teamwork
    - CI/CD
  - UI/UX
  - Designing modules + the interfaces between them
  - More testing!
    - Including breaking serialization across teams

Roadmap

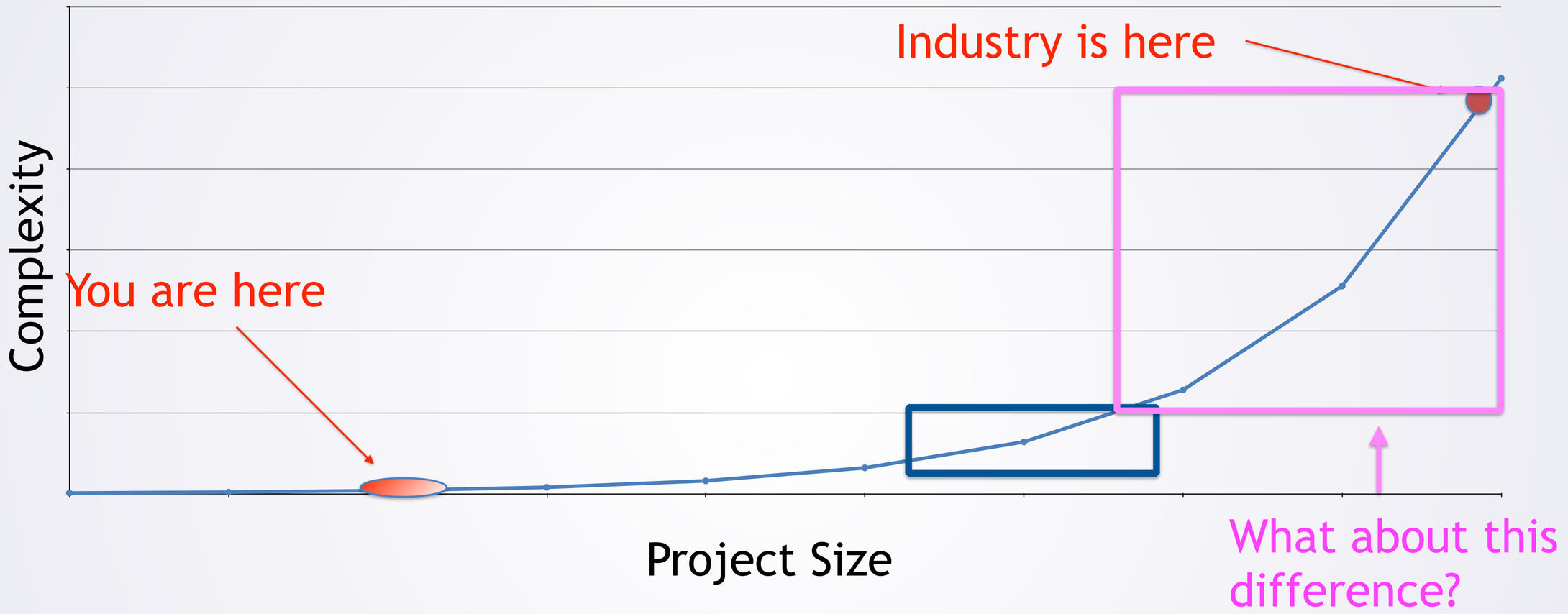


- 3 major parts of semester
  - Small Scale: A few classes (1-~10)
  - Medium Scale: Modules: many classes (10+)
  - Large Scale: Systems: multiple components/programs interacting

Roadmap Cont'd

- **Last**: "large scale" software engineering
  - System architectures
    - Monolith? Micro services? Event driven?
  - Components of large scale systems
  - Security
  - Maintenance + monitoring

Roadmap



- What about the remaining gap?
  - Don't really need any new techniques.. Same ideas, just at larger scale
  - Can't really have you all write a million lines of code this semester..

Logistics: Assignments

- You will have the following types of assignments:

- **Individual Programming**: design, write, test code yourself
- **Team Programming**: larger software project in 3 parts
- **Class Participation**: you need to engage + talk in class!
- **Exams**: midterm + final

Logistics: Programming Assignments

- **Individual Programming**: design, write, test code yourself
- **Team Programming**: larger software project

- First step for each assignment (should be done within 24–48 hours)
  - Plan: what are your sub-goals?
  - When are they going to be done?
  - How do you demonstrate them?
  - More on this as we talk about project management

Programming Assignment Late Policy

- Unusual/complex late policy:
  - Penalty is function of how many days and WHEN you ask for them

Programming Assignment Late Policy

Marginal Point Cost

Number of days after assn release day is requested

Nth late day requested	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	4	5	6	7	7	8	8	9	9	10	10	11	11
2	6	8	10	12	13	15	16	17	18	19	20	21	22	22
3	9	13	16	18	20	22	24	25	27	28	30	31	32	34
4	12	17	21	24	27	29	32	34	36	38	40	42	43	45
5	15	21	26	30	34	37	40	42	45	47	50	52	54	56
6	18	25	31	36	40	44	48	51	54	57	60	62	65	67
7	21	30	36	42	47	51	56	59	63	66	70	73	76	79

- Unusual/complex late policy:
  - Penalty is function of how many days and WHEN you ask for them

Duke UNIVERSITY

Programming Assignment Late Policy

Number of days after assn release day is requested

Nth late day requested	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	4	5	6	7	7	8	8	9	9	10	10	11	11
2	6	8	10	12	13	15	16	17	18	19	20	21	22	22
3	9	13	16	18	20	22	24	25	27	28	30	31	32	34
4	12	17	21	24	27	29	32	34	36	38	40	42	43	45
5	15	21	26	30	34	37	40	42	45	47	50	52	54	56
6	18	25	31	36	40	44	48	51	54	57	60	62	65	67
7	21	30	36	42	47	51	56	59	63	66	70	73	76	79

- Example:
  - Day after assignment comes out: "we need 1 extra day" = **-3 points**

Programming Assignment Late Policy

Number of days after assn release day is requested

Nth late day requested	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	4	5	6	7	7	8	8	9	9	10	10	11	11
2	6	8	10	12	13	15	16	17	18	19	20	21	22	22
3	9	13	16	18	20	22	24	25	27	28	30	31	32	34
4	12	17	21	24	27	29	32	34	36	38	40	42	43	45
5	15	21	26	30	34	37	40	42	45	47	50	52	54	56
6	18	25	31	36	40	44	48	51	54	57	60	62	65	67
7	21	30	36	42	47	51	56	59	63	66	70	73	76	79

- Example:
  - Day after assignment comes out: "we need 1 extra day" = -3 points
  - 5 days later (day 6) "oh no we are behind, we need 1 more day" = - (6 + 7) points = **-13 points**

Programming Assignment Late Policy

Number of days after assn release day is requested

Nth late day requested	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	4	5	6	7	7	8	8	9	9	10	10	11	11
2	6	8	10	12	13	15	16	17	18	19	20	21	22	22
3	9	13	16	18	20	22	24	25	27	28	30	31	32	34
4	12	17	21	24	27	29	32	34	36	38	40	42	43	45
5	15	21	26	30	34	37	40	42	45	47	50	52	54	56
6	18	25	31	36	40	44	48	51	54	57	60	62	65	67
7	21	30	36	42	47	51	56	59	63	66	70	73	76	79

- Example:
  - Day after assignment comes out: "we need 1 extra day" = -3 points
  - 5 days later (day 6) "oh no we are behind, we need 1 more day" = - (6 + 7) points = -13 points
  - On day 11: "oh my gosh, more behind 1 more day…" = - (9+15+10) = **-34 points**

Duke UNIVERSITY

Programming Assignment Late Policy

Number of days after assn release day is requested

Nth late day requested	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	4	5	6	7	7	8	8	9	9	10	10	11	11
2	6	8	10	12	13	15	16	17	18	19	20	21	22	22
3	9	13	16	18	20	22	24	25	27	28	30	31	32	34
4	12	17	21	24	27	29	32	34	36	38	40	42	43	45
5	15	21	26	30	34	37	40	42	45	47	50	52	54	56
6	18	25	31	36	40	44	48	51	54	57	60	62	65	67
7	21	30	36	42	47	51	56	59	63	66	70	73	76	79

- Example:
  - Contrast with "we need 3 extra days" on day 1 = -(3+6+9) = **-18 points**

Programming Assignment Late Policy

Number of days after assn release day is requested

Nth late day requested	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	3	4	5	6	7	7	8	8	9	9	10	10	11	11
2	6	8	10	12	13	15	16	17	18	19	20	21	22	22
3	9	13	16	18	20	22	24	25	27	28	30	31	32	34
4	12	17	21	24	27	29	32	34	36	38	40	42	43	45
5	15	21	26	30	34	37	40	42	45	47	50	52	54	56
6	18	25	31	36	40	44	48	51	54	57	60	62	65	67
7	21	30	36	42	47	51	56	59	63	66	70	73	76	79

- Example:
  - Contrast with "we need 3 extra days" on day 14 = -(11+22+34) = **-65 points**

Duke UNIVERSITY

Other rules about late policy

- Does not apply to **exceptional** situations
  - In hospital, death in family, etc.
  - Documentation may be required
  - Contact professor ASAP
- Once you (or your group) "buy" a late day you can **NOT** undo it.
  - Can't ask for 5 late days on day 1, then later say "nevermind, only need 3"
- For longer than 14 days or more than 7 days, marginal cost increase by 1 point per day (in each direction)

Why This Policy?

- **You all tell me**: why do I have this policy?

Why This Policy?

- **You all tell me**: why do I have this policy?
- Plan carefully and accurately!
  - If you realize you need more time on day 1, late days are "cheap"
  - If you ask at the last minute, they are expensive.
- Start early!
  - Realize you are behind? Make a plan to catch up or ask for late day **now**.
- Gives some flexibility
  - "Oh my gosh but that is due at the same time as [….]"
  - Plan!

Duke UNIVERSITY

Student Model of Project Management

1/14 650 Assn Released

1/15 568 Assn Released

1/24 651 Assn Released

2/4 650 Due

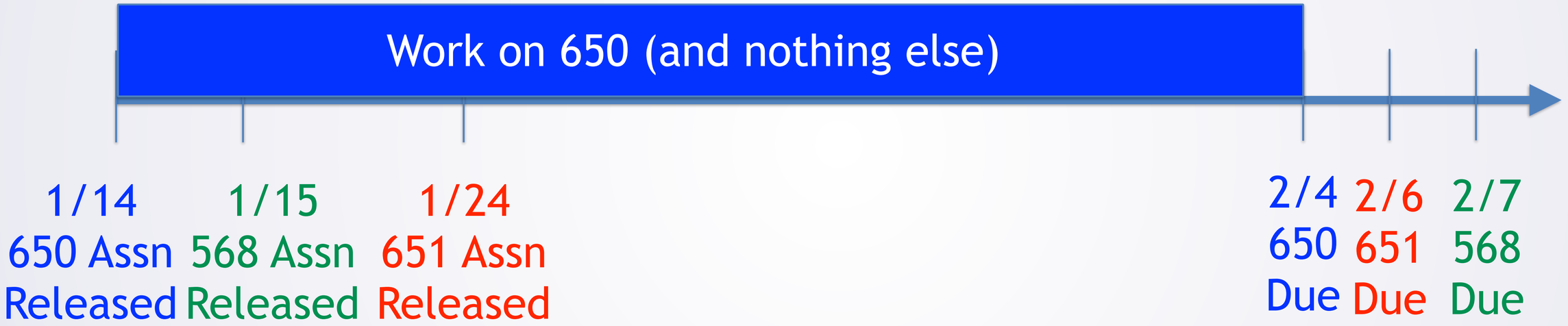
2/6 651 Due

2/7 568 Due

Actual Due Dates For Assn 1 In Each Class

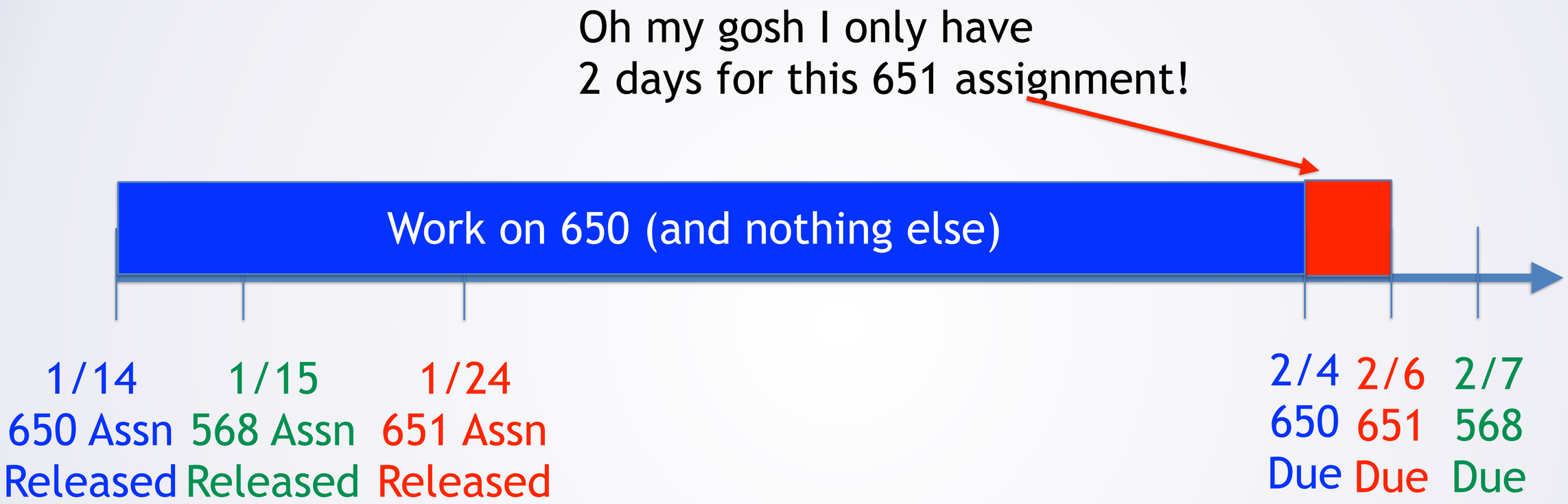
- I've learned this is how you all manage your time.

Student Model of Project Management



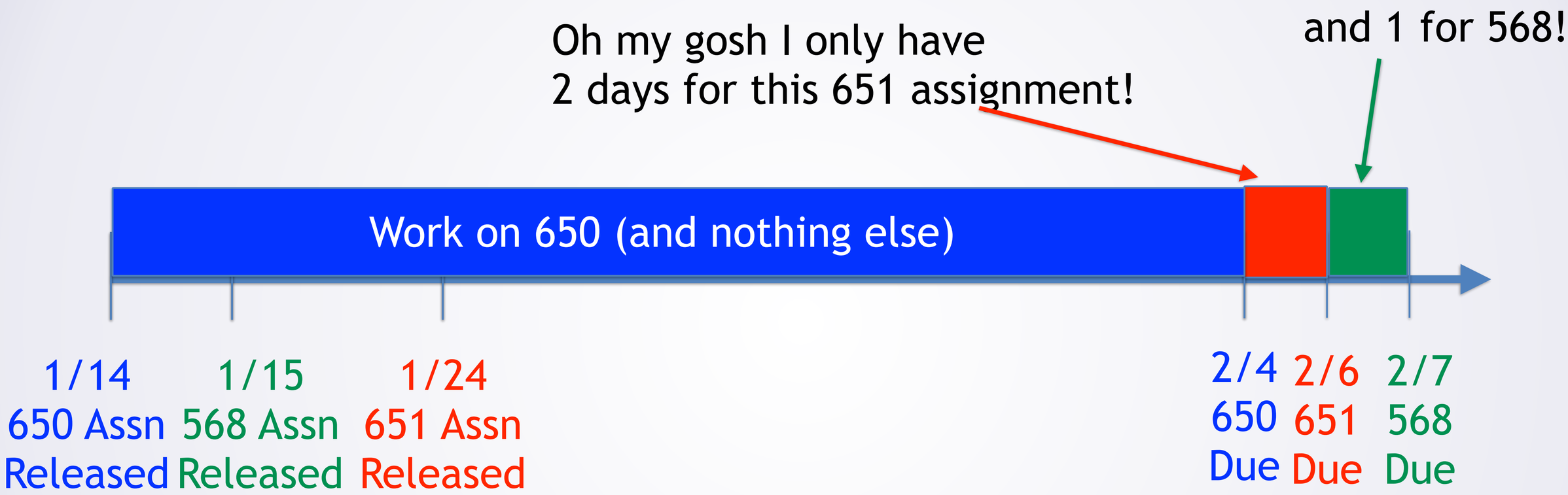
- I've learned this is how you all manage your time.

Student Model of Project Management



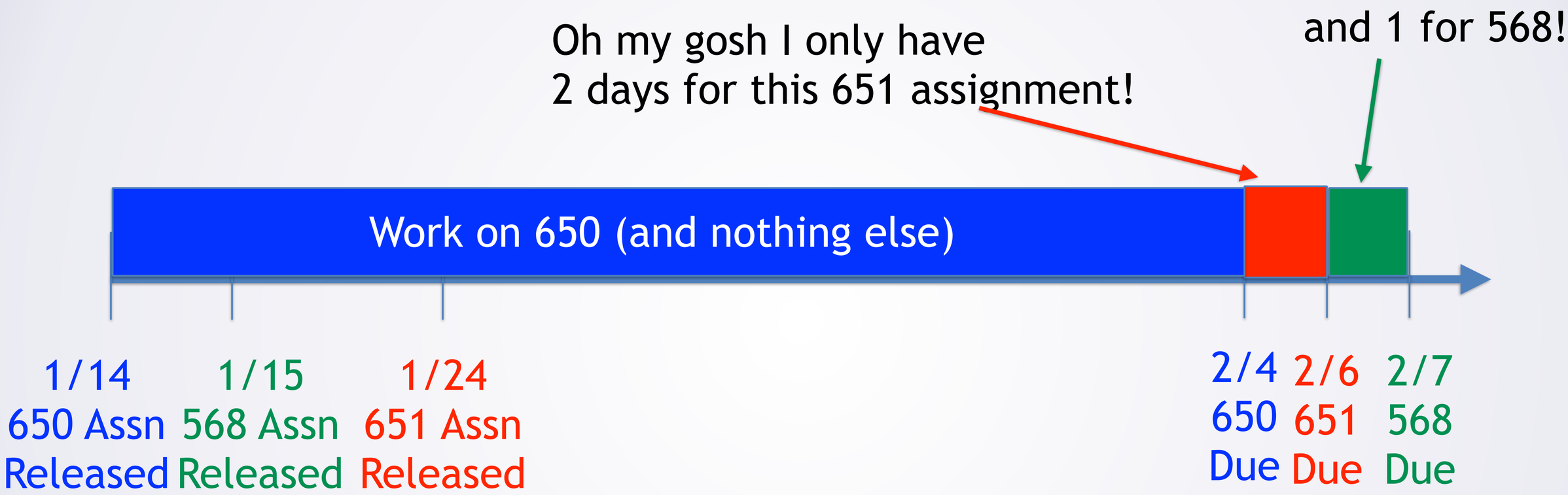
- I've learned this is how you all manage your time.

Student Model of Project Management



- I've learned this is how you all manage your time.

Student Model of Project Management



- I've learned this is how you all manage your time.

DO NOT DO THIS

What You Need To Do Instead

DO THIS INSTEAD

1/14 650 Assn Released

1/15 568 Assn Released

1/24 651 Assn Released

2/4 650 Due

2/6 651 Due

2/7 568 Due

- Why should you do this?
  - After all, EDF scheduling and context switch overhead make other seem good, right?

Student Model of Project Management

Oh I need to go to OH in 2 days because I realized I don't understand X...

I'm stuck until my group member does Y...

1/14 650 Assn Released

1/15 568 Assn Released

1/24 651 Assn Released

2/4 650 Due

2/6 651 Due

2/7 568 Due

I just have a mental block about this bug and need to step away for a bit.

- You will encounter delays..
  - Switching between tasks is good.

Other Things About Time Management

- If you plan for milestones across weeks, you can recover
  - Get a bit behind, time to fix
- If you plan for everything to happen in 2 days, you cannot recover
  - Falling one hour behind is a catastrophe
- In the Real World, you will need to handle multiple project at a time
  - Probably at least 3.

Academic Integrity

- You are expected to you own work in this class.
  - After all, you are here to learn.
  - If you can't do this, you can't do the job you want.
  - Your friends/the internet won't do it for you…
- 4 policies…
  - **Individual Programming**: **discuss, but write own code**
  - **Team Programming**: **team**
  - **Class Participation**: **open**
  - **Exams**: **individual**

Duke UNIVERSITY

Academic Integrity

- **Individual Programming**: **discuss, but write own code**
- I expect you to write your own code.
- Do not show your code to others or look at anyone else's code
  - Includes finding code on Internet
- Can have discussions such as
  - "I don't understand Factory pattern, can you explain it?"

Academic Integrity

- **Team Programming**: **team**
- Work done entirely by your team
  - Discuss with team mates
  - Share code with team mates
  - Do not look at other team's code, or on Internet
- Think of this like a company:
  - Do not expose your company to IP infringement lawsuit!

Academic Integrity

- **Class Participation**: **open**
  - Many are think-pair-share: expect you to talk to each other
  - About participation, not correctness
  - These are intended to be an open discussion, nothing against the rules

Academic Integrity

- **Exams**: **individual**
- Do not discuss at all with other students
  - Can ask professor/TA clarifying questions
- May bring one page of notes (handwritten by you)
  - Front and back
  - Standard 8.5"x11"
  - No magnifying glass

Assignment Particulars

	Percent	From	To
Class Participation	5	(always)	
Individual Programming 1	9	Fri 1/24	Thurs 2/6
Individual Programming 2	9	Fri 2/7	Thurs 2/20
Midterm	15	Tues 2/25 or Wed 2/26	
Team Project Evolution 1	10	Fri 2/21	Thurs 3/19
Team Project Evolution 2	10	Fri 3/20	Thurs 4/2
Project Presentations	7	Tues 4/7	Wed 4/15
Team Project Evolution 3	15	Fri 4/3	Wed 4/22
Final Exam	20	As stated by registrar	

Letter Grades

- Letter grades follow the standard 10 point scale with 3 points for - and 3 for +
- A+: [97, ∞) A: [93, 97) A-: [90, 93)
- B+: [87, 90) B: [83,87) B-: [80,83)
- C+: [77,80) C: [73,77) C: [70, 73)
- F: [0,70)

First Thing To Do

- Read All of Programming, Chapter 31: Java
  - Please do by 2 classes from now