Intro to Systems Analysis and Design

CIS641

Erik Fredericks // frederer@gvsu.edu

Adapted from materials provided by Gregory Schymik and the textbook (Systems Analysis and Design 5th/6th Ed.)

Overview

- Welcome!
- Syllabus
- SAD

WELCOME

Syllabus

As always, the syllabus is **worth reading**

Important topics like:

- When are my exams?
- What is the grading breakdown?
- What time does this class meet?
- Where can I find the nifty textbook?

About the textbook

Required, **HOWEVER**:

As part of the GVSU SAVE program, your account was **auto-charged**

- You can opt-out by September 9th to remove the charge if you want to get it elsewhere
  - The 5th ed. should be fine if you can find it *online somewhere*
    - *I'll transpose the HW questions into Blackboard if they differ*
  - Or if you want a hard copy and it's cheaper...

- https://lakerstore.gvsu.edu/gvsusave

About the syllabus...

Let's go through it, shall we?

What we ***will*** cover

Systems analysis and design from an **IS perspective**

Practical software engineering

Learning how to read **research papers** AND
And
And
Draw conclusions from them *quickly and efficiently*

What we will ***not*** cover

In-depth programming

- For programming tasks, I'm going to make the assumption that you know a base level of programming

What do you want to cover?

Knowledge you are **assumed to have**

Ability to program in **some language**

- You are a graduate-level CIS student, I am expecting you to write a program
- (If you don't know how to program, there are options as well)

Ability to work **on a team**

- You are a graduate-level CIS student, I also expect maturity and even effort-sharing
  - If you don't want to work on a team, I expect you to be able to function independently

Ability to **talk to me if you have problems**

- Don't be nervous, I am very approachable

How class will *generally* work

Our classes are (either) 1.5 or 3 hours long

- Part of class: me talking
- Part of class: you talking
- Part of class: us discussing together

Be prepared when you come to class to actively participate in discussions!

- I'll generally tell you papers to read, etc.

Our tech stack

Class is **synchronous in-person**, meaning:

1) Class runs at specific days/times (I will not be recording)
2) Office hours exist

Class website:	Blackboard
Chat (offline):	Discord
Term projects:	GitHub

If you want to get a hold of me for questions:

- Ping me in discord
- Email me
- Visit during office hours

PROJECTS

You will develop a semester-long project

Goals of this project:

1) Have something portfolio-worthy at the end

2) Develop it like a real-world project
   a) Project roles
   b) Software artifacts
   c) etc.

Project steps

1) Form teams

2) Submit a project proposal

3) Setup a GitHub repository **and** website (github.io)

4) Project checkpoints

5) Project delivery
    a) Evaluate other teams' artifacts

Project?

I want you to create something that is **interesting to you**

Are you a software developer with latent tendencies for video game design?

- Make a video game!

Aspiring app developer?

- Make that app!

Terminal wizard?

- Make a Bash/PowerShell script environment that will make your sysadmin blush!

No clue!?!?

- Let's talk

Team?

I am allowing teams of **1 to 4 people**

- Though I would **strongly** recommend being in a group
  - Since you are masters students and most likely have real-life I'll let 1 slide
  - Though, I'll be expecting a full project regardless!

Things you'll have to do:

- Requirements specifications
- Artifacts
- Project checkpoints
- Presentations
- Etc.

Things to focus on for the **project**

1) Does it have to work?
   a) Yes! But … it doesn't have to be a game-changing multi-million LOC project
   b) Just make sure it functions and I'll be happy

2) The focus on this class is the **IS perspective**
   a) So, put a larger focus on the artifacts and models of your project

Research discussions

This is a **600-level class**

- Meaning it's 100 levels higher than a 500-level class!

We'll talk research throughout the semester as well

- Meaning *you'll* talk research throughout the semester :)

Since we have so many people, here's how that will work:

- Each week, 2-4 people will present a paper to the class
    - *It is up to you to sign up!!!*

- You'll each take a turn discussing the paper, generally on some particular point
- (Next week I'll lead to show you what I mean)

Questions so far?

burgertv:

> Tina asking the important questions

Source: burgertv

What programming languages do you know?

Introduce yourselves!

This class tends to be very interactive, so let’s do the thing of introducing each other to each other

Systems Development Lifecycle (SDLC)

Let's get into some material for the next hour!

- Four phases of SDLC
- Team roles and skillsets
- Design processes
- A bit o' UML

Why is a formal process absolutely necessary?

Let's look at a few case studies:

Case Study : Ariane 5

Ariane 5 rocket -- created by European Space Agency
Required **10 years** and **$7 billion** to build

In under a minute after launch, the rocket exploded

As a result of...
Trying to put a 64-bit number into a 16-bit variable

A bug and a crash, J. Gleick, New York Times, Dec 1996

Case Study : Toyota Prius Braking

Delay in brake software caused vehicles to brake slower than expected

If traveling 60mph, the brakes didn’t trigger until ~90ft beyond initial brake press

Disconnect in the anti-lock brake system (ABS) -- < 1 second lag

CNN Feb 4, 2010

Case Study : Inadvertent Overdosing of Radiation

Let's talk about the **Therac-25**

- Radiation therapy machine (chemotherapy)
- Iteration over the well-used Therac-6
  - Now completely software-controlled
    - .... Including safety mechanisms
    - Hey, its probably cheaper to have everything be software-controlled, right?

- Machine was extensively tested and verified over several years
  - A single programmer wrote all the code
  - Unit testing and general SW testing was minimally done

Leveson 1993
http://www.cs.umd.edu/class/spring2003/cmsc838p/Misc/therac.pdf

Therac-25

11 units were installed in the US and Canada

- **Six** massive overdoses of radiation were reported

Why?

- Program faults from the Therac-20 software codebase that were not previously detected
  - Well, why weren't they detected?
    - Because the **hardware interlocks** prevented the machine from ever entering a state
    - Now that it was completely SW-controlled, well, no safety mechanism existed to prevent overdose

Therac-25

Effectively this problem results from **race conditions** and **concurrency errors**

But, the human element is also to blame

- Medical staff initially did not believe that their machine was causing the problems
- Improper training led staff/nurses to use shortcuts or rapid key entries that caused the machine to go into the wrong power mode
- From Wiki (https://en.wikipedia.org/wiki/Therac-25#Additional_reading):
    - The failure only occurred when a particular nonstandard sequence of keystrokes was entered on the VT-100 terminal which controlled the PDP-11 computer: an "X" to (erroneously) select 25 MeV photon mode followed by "cursor up", "E" to (correctly) select 25 MeV Electron mode, then "Enter", all within eight seconds.[5] This sequence of keystrokes was improbable, and so the problem did not occur often and went unnoticed for a long time.[3]

What does this tell us?

Not that the engineers were incompetent. A **lot** of effort went into designing and testing these systems.

It tells us that testing is **never** complete and **it is impossible to consider every single conceivable issue**

Could a Lint test have solved the Ariane-5 explosion? Possibly
Could extensive human testing have solved the <X> catastrophe? Maybe, maybe not.

Therefore, it is important to verify and test as much as possible prior to release!

So software engineering…

Historically, probably some of the driest classes you will get (unless if you have one of our nifty professors, naturally)

But that’s all stuff that you do in the working world, right?
Specifications, testing, verification. That’s the bland side of programming.

Where’s the excitement? Where’s the sex appeal?

Well…

This is also why I feel that software engineers should have the same requirement as Mechanical Engineers to be tested for a Professional Engineer license.

People’s lives can and will be at stake as a result of your programming and design processes.

Software Engineer’s Responsibilities

Assume you programmed the Therac-25

You are **directly (or indirectly) responsible** for causing severe radiation burns, at least one of which resulted in a breast amputation

The patient died on November 3, 1985, of an extremely virulent cancer. An autopsy revealed the cause of death as the cancer, but it was noted that had she not died, a total hip replacement would have been necessary as a result of the radiation overexposure. An AECL technician later estimated the patient had received between 13,000 and 17,000 rads.

Software/Quality Engineer’s Responsibilities

Now that (I assume) you’re scared straight

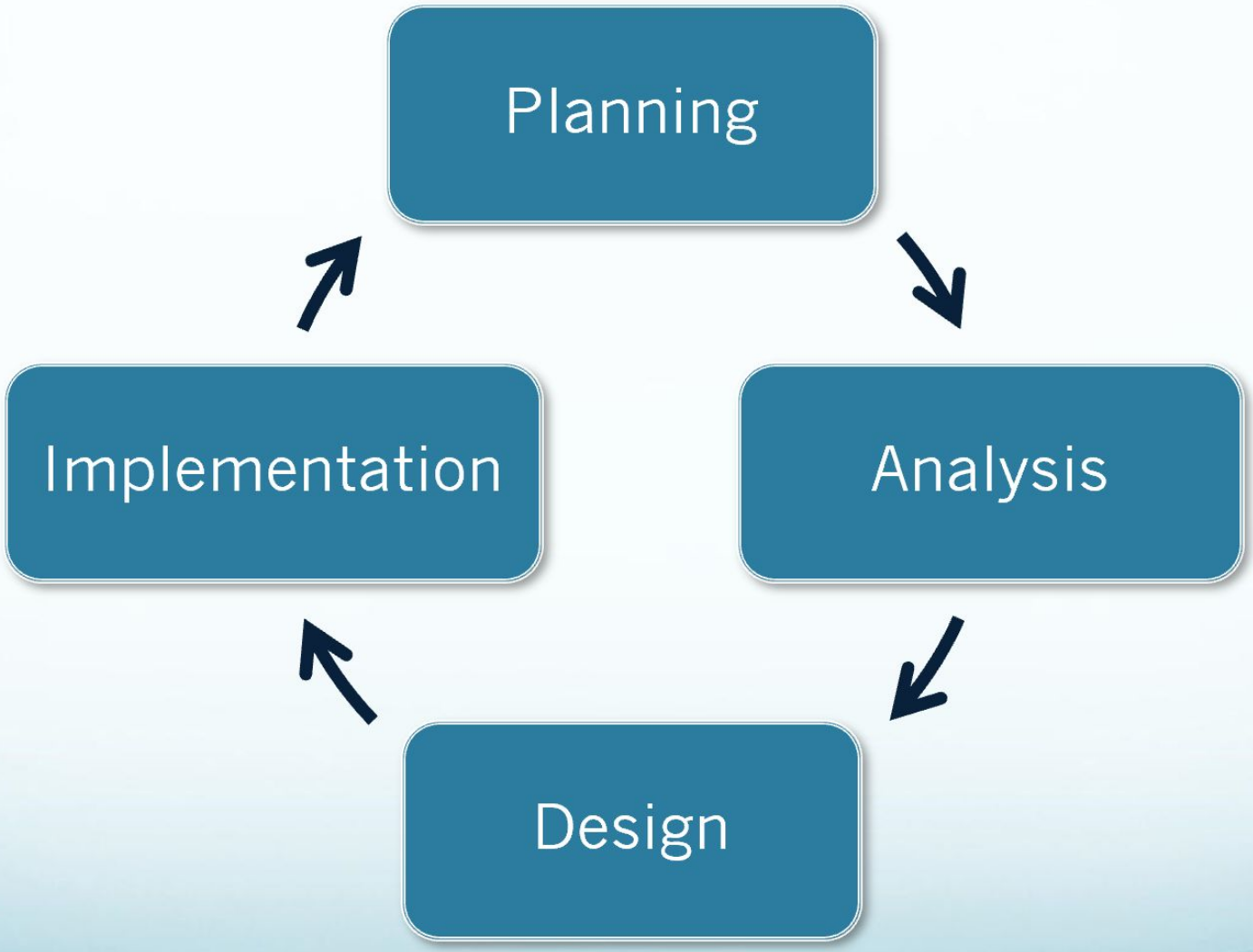
Be responsible when you program. It’s not a game. People’s lives will be affected by your actions.

/end soapboxing

So now that we’re all mortified…

In this class, we will consider the real-world implications of software engineering concepts in terms of real-world concerns

- So let's talk about SDLC!


Planning
Analysis
Design
Implementation

Many different types of SDLCs

Consider it as a **system**

Inputs → SYSTEM → Outputs

Each phase (and steps within) must be **documented**

- Also known as your **deliverables**

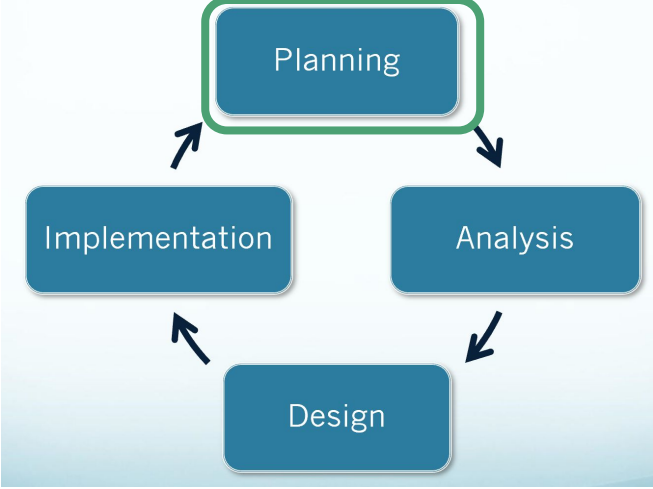
Phases are executed in some sort of **pattern**

- Sequentially, iteratively, etc.
- Different styles (Waterfall, Agile, etc.)

Ability to Develop Systems	Structured Methodologies		RAD Methodologies			Agile Methodologies	
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	SCRUM
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Good	Good
That Are Complex	Good	Good	Good	Poor	Excellent	Good	Good
That Are Reliable	Good	Good	Good	Poor	Excellent	Excellent	Excellent
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Excellent	Excellent

% *Systems Analysis and Design 5th Ed.*

Planning



Key ingredients:

- Why should we build it?
- What value does it bring?
- How long will the build require?

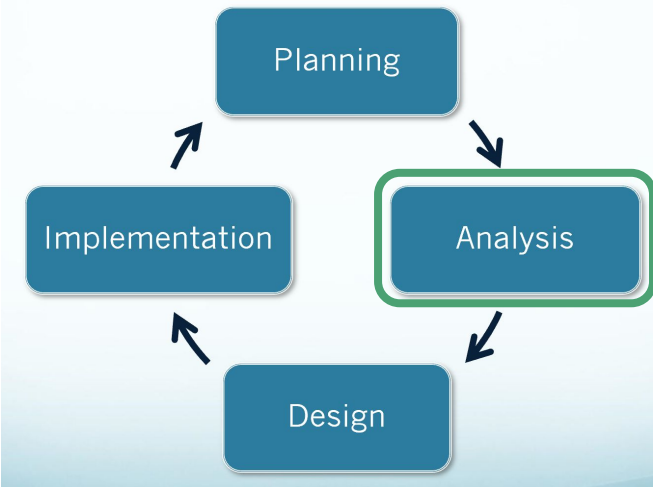
Project initiation:

- Develop and/or receive request
- Feasibility analysis

Project management:

- Create a work plan
- Add required staff
- Monitor/control project

Analysis



Key ingredients:

- Who will use it?
- What should the system do?
- When/where will it be used?

Analysis strategy:

- Model and formulate system

Gather requirements:

- System concept
- Business model
- Develop system proposal

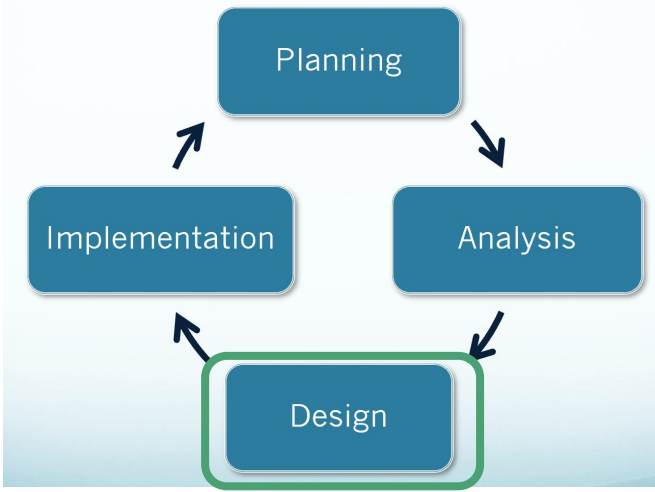
Design

Key ingredients:

- How will we build it?

We are going to need:

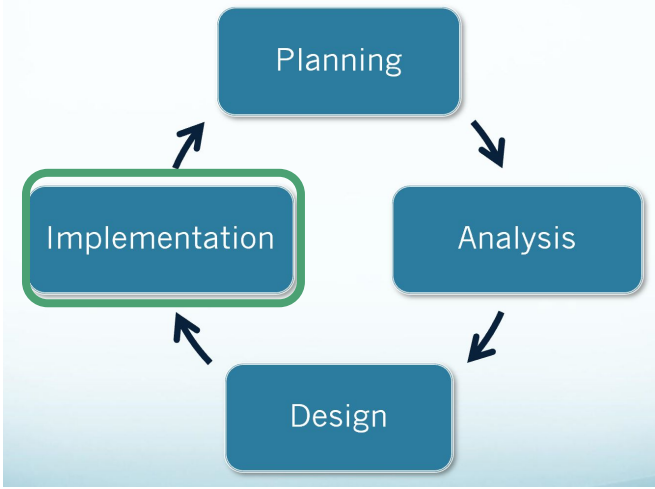
- A design strategy
- System architecture and interfaces
- File/database specifications
- Program design for:
  - Necessary programs and what they do!



Implementation

Key ingredients:

- Construct system
- Install system
- Support system (maintenance)



WHAT IF SOMETHING WENT WRONG?!?!?!?!?

Where do we go in the cycle?

Discussion time!

Here is a case study

- It's the attachment in Blackboard!
- Ethics articles ➔ Boeing 737-MAX!
  - (The Therac articles you'll read as homework)

I am going to give you **20-25 minutes** to read it over, discuss, and then we'll come back and talk about it!

- Breakout rooms:
  - Nominate **1 person** to record your group's answers
  - Nominate **1 person** to share the slide with the questions (that person should share that slide in your room (it is the next one after the coffee pour))

@gerik

Discussion time!

1) What did you think about it (in general)?

2) Who was at fault?

3) Why are they at fault?

4) What could have been done to prevent this (if anything)?

5) How does this compare to Therac?

```
$> start.here.sh
```

Back to work

Skills o' the analyst

You are the **agent of change**

- Identify methods for ***improving the organization***
- Motivate others
- Train others!

Skills o' the analyst

Your skills:

- Technical:
  - Understand technologies involved
- Business:
  - Understand business processes and work with/around them!
- Analytical:
  - Be a problem solver (not just an identifier)
- Communications:
  - Speak with technical and non-technical people alike
- Interpersonal:
  - Leadership and management
- Ethics:
  - Deal fairly with others
  - Protect confidential information

Object-oriented (OO) systems analysis/design

Balance data and process

Uses Unified Modeling Language (UML) and the "unified" process

- Use-case driven
- Architecture-centric
- **Iterative and incremental**

Characteristics of OO systems

Classes and objects:

- Object (instance): instantiation of a class
- Attributes: information describing aspect of a class
- State: values and relationships at some point in time

Methods and messages:

- Methods: class behavior
- Messages: information sent to object to trigger a method

Characteristics of OO systems

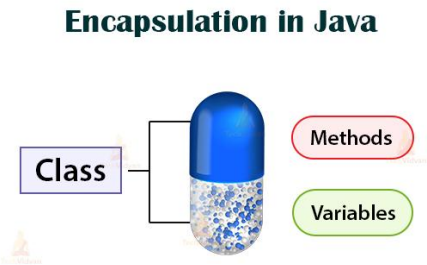
Encapsulation and information hiding:

- Encapsulation: combining process and data
- Information hiding: functionalities abstracted

Inheritance:

- General classes created (superclasses)
- Sub-classes inherit from general classes

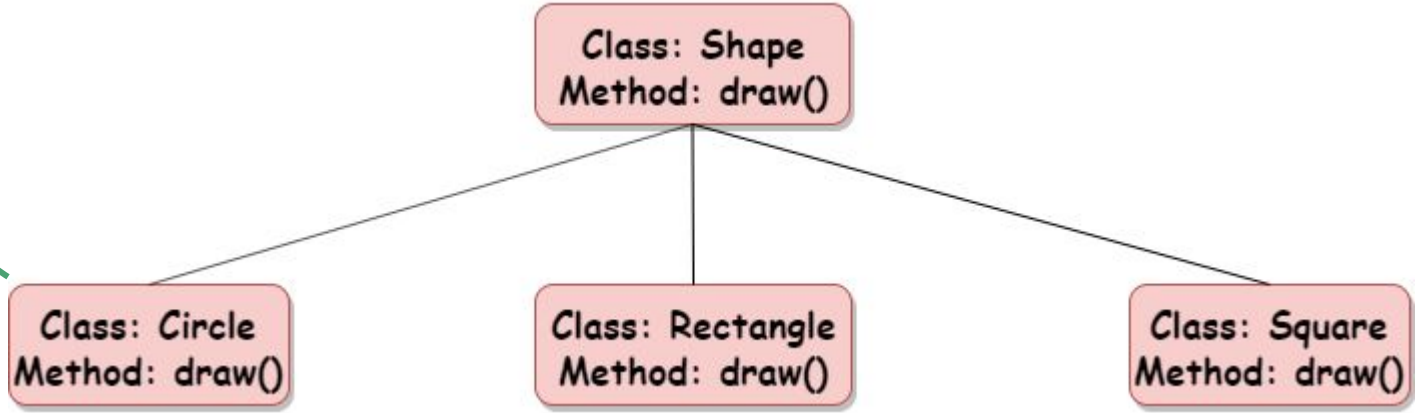
`Dog IsA Animal`, etc.



Characteristics of OO systems

Polymorphism and dynamic binding:

- Polymorphism: same messages have different meanings
- Dynamic binding: object type not determined until run time
  - Static binding: object type set in stone



Characteristics of OO systems

Use case driven:

- Defines system behavior
- Each use case focuses on some *business aspect*

Architecture-centric:

- Functional (external): user's perspective
- Static (structural): attributes, methods, classes, relationships
- Dynamic (behavioral): messages between classes and behaviors

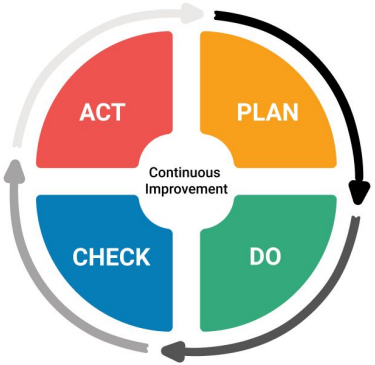
Characteristics of OO systems

Iterative and incremental:

- Continuous testing / integration, refinement
- Analysts understand system better over time!

Benefits of this approach!

- Complexity broken down into manageable chunks
- Individual modules worked on … individually



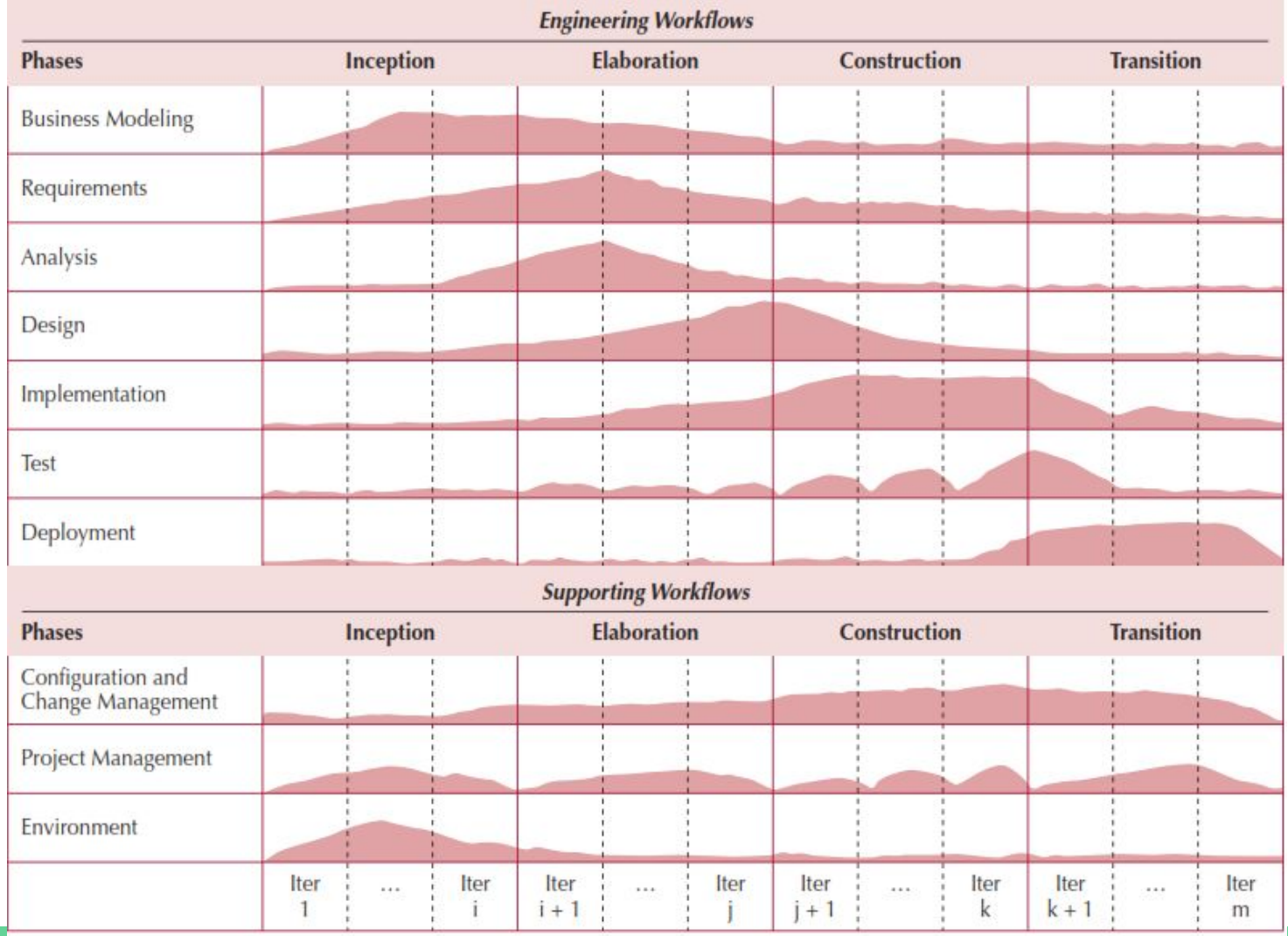
THE PROCESS THE PROCESS THE PROCESS THE PROCESS THE PROCESS THE PROCESS THE PROCE

Specific methodology that maps modeling (UML) between OO analysis and design

Comprises phases and workflows:

- Phases ➔ time periods of development
- Workflows ➔ tasks within each phase

- Activities between will overlap!


Engineering Workflows
Phases
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis
Design
Implementation
Test
Deployment
Supporting Workflows
Phases
Inception
Elaboration
Construction
Transition
Configuration and Change Management
Project Management
Environment
Iter 1
...
Iter i
Iter i + 1
...
Iter j
Iter j + 1
...
Iter k
Iter k + 1
...
Iter m

Phases (Unified process)

Inception
- Feasibility analyses
- Workflows will vary, but should focus on business modeling / requirements gathering

Elaboration
- Analysis and design

Construction
- Implementation

Transition
- Testing and deployment

Workflows (Engineering)

- Business modeling
- Requirements
- Analysis
- Design
- Implementation
- Testing
- Deployment

... is this a sequential process?

Supporting workflows

- Project management
- Configuration / change management
- Environment
- Operations / support
- Infrastructure management

Same question!

What does the unified process **not** include?

- Staffing
- Budgeting
- Contract management
- Maintenance
- Operations
- Support
- Cross/inter-project issues

Unified Modeling Language (UML)

What is it, other than a perceived PITA?

- "Provides a common vocabulary of object-oriented terms and diagramming techniques rich enough to model any systems development project from analysis through implementation"

Structural diagrams
Behavioral diagrams
Etc.
(different versions have different types)

https://online.visual-paradigm.com/

100%

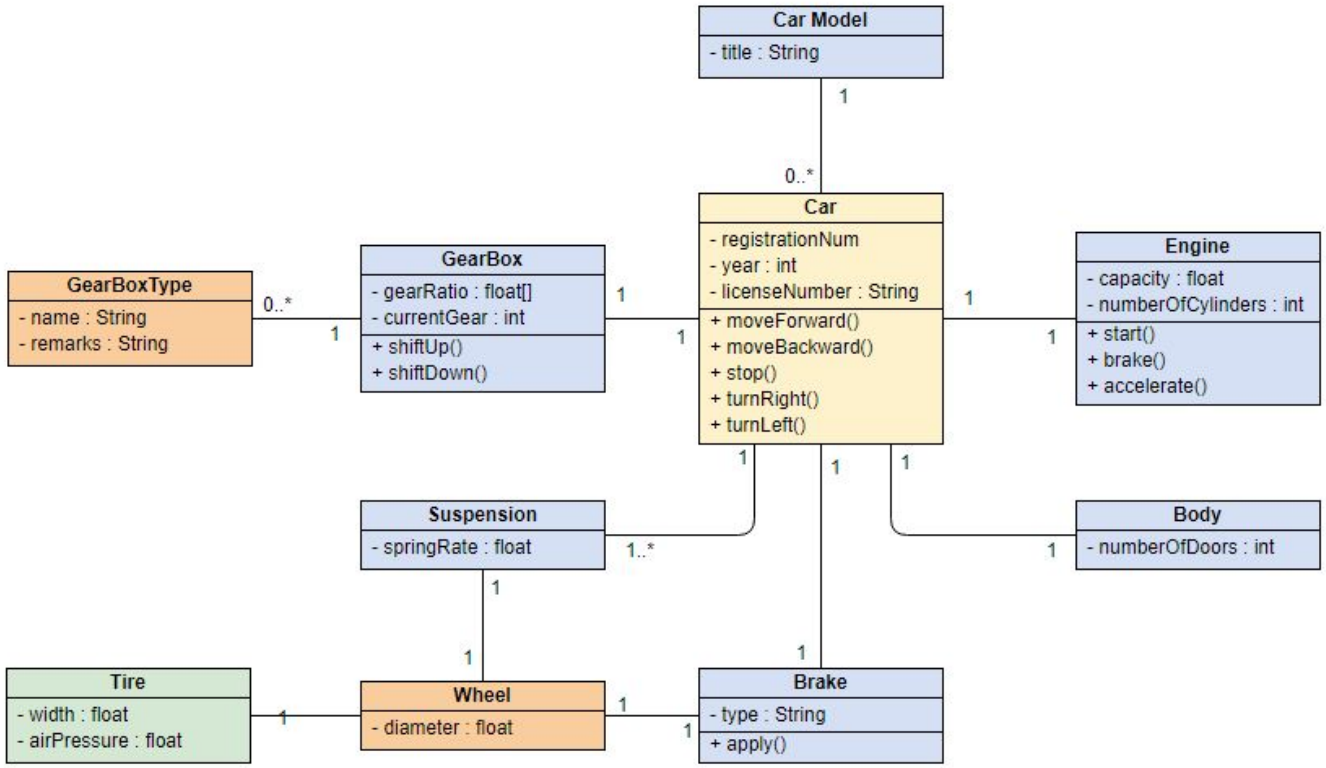
Search Shapes

Scratchpad

Drag elements here

Class

More Shapes...



What types of projects are you interested in working on?

For next time:

- If you don't already have them, create accounts on the following services (it is part of your homework assignment)
  - Discord
  - GitHub

- Check the Blackboard assignments page for your first assignment!
  - Hint: it is ethics related!

- Don't forget to read the research article posted ahead of next class!
  - It is in the sidebar under **Research Papers**

If time, group formation and project discussion

Let's spend some time talking term project, as I want it to be a **continuous focus** throughout the semester

- Meaning, if you start it with 2 weeks left you're probably not going to be happy with your grade

So what am I looking for?

1) A **publicly-available**, **functioning deliverable** (can be a prototype, but fully-realized would be better)
   a) Topic: group-specific
   b) You will need to submit a project proposal early-on however

2) **Software artifacts**
   a) Things like requirements, use cases, user stories, etc.

What shall we do now?

Let's form project teams!

In the syllabus I specified teams of **1 - 4**, however if you have grander schemes in mind I'll accept a larger group

- The effort level must scale with the # of team members though

If your team is fully-formed:

- Split off and start discussing ideas!

If you need a team:

- Try and join a team matching your interests (or looks like they need a person)

If you still need a team at the end of the lecture:

- I'll randomly assign you to a group