Announcements

- Project 1 grades will be out tomorrow
  - Great job, everyone! ☺
- Extra Credit Opportunities today
- Project 1 Design due on 13th
- Project 2 has started. Code review counts from 10/10 onwards
- Project 2 Topic and MMFs assignment released
  - Due 10/22 before class. Mentors will check-in, grade the topic idea and MMFs in class, let you know if they are not substantial enough, discuss AI tool usage.
- Test assignment releasing 10/17 - related to Project 2
  - Using Copilot for unit and integration testing videos up on class website along with additional resources

AI research from Project 1: Summary

Benefits of AI Tools

1. **Increased Productivity and Efficiency**: Many groups reported a significant reduction in development time, particularly in tasks involving code generation, debugging, and routine coding tasks. Tools like GitHub Copilot and Continue were frequently mentioned for their role in speeding up the coding process.
2. **Enhanced Code Quality**: Several reports highlighted that AI tools helped reduce errors and improve the quality of code. AI suggestions were particularly useful in identifying bugs and proposing fixes.
3. **Improved Learning and Understanding**: AI tools, especially those integrated within IDEs like Visual Studio Code, facilitated better understanding of coding practices and solutions through explanations and step-by-step guidance.
4. **Support in Design and Architecture**: Some groups used AI tools to assist in the design phase, finding them useful for validating design patterns and generating UML diagrams or architecture ideas.

Drawbacks of AI Tools

1. **Dependency and Over-reliance**: Some students expressed concerns about becoming too dependent on AI tools, which might affect their fundamental coding skills.
2. **Inconsistencies and Errors**: AI tools sometimes generate incorrect or suboptimal code, requiring manual review and correction. This was particularly noted with complex coding tasks where the AI struggled to maintain context or produce logically sound outputs.
3. **Learning Curve and Integration Challenges**: While AI tools provided significant benefits, a learning curve was associated with effectively using them. Integration issues were also mentioned, particularly regarding setting up and configuring the tools within existing development environments.
4. **Limited by Context Understanding**: AI tools occasionally failed to grasp the larger context of the project, leading to suggestions that, while syntactically correct, did not align with project goals or specific requirements

Midterm Feedback

Best Aspect of the course:

- Tools of the trade lectures – has been continuously added/edited based on student suggestions (TA workshop, gcp and rest demos/assignments were new additions last semester)
- Small individual assignments based on tools
- Project – Practicality, teamwork, tools
- Quizizz for theoretical lectures
- 3 out of 70 students said they didn't like the course. I wont count that as representative.

Things that can be changed

- Recorded Lectures – **Agreed. All demo lectures have been recorded though. I will record all others going forward.**
- Quizizz extra credit should be partial grade – **Disagree**. It is extra credit so it can't be participation. **But more participation-based quizzes coming up. Also 1 wrong will be mostly accepted**

Midterm Feedback

Things that can be changed

- Don’t use AI for the course. I don’t believe in or like AI. 5 out of 70 students said this - **The goal of integrating AI into the course is to give you hands-on experience, so you can form a well-informed opinion. It's okay if you end up disliking AI, but I want you to have enough exposure and understanding to support that opinion with knowledge and experience.**
- Make attendance compulsory for in-class quick standup meetings – **agreed. I should have been stricter at the beginning of the course. 5 pm time also doesn’t help.**
- More TOTT , remove theoretical lectures – **Unfortunately, can’t change curriculum content. Trying to add more practicality while still adhering to curriculum policies**

Rate TOTT demos on a scale of 1 -5 : Average - 4.4/5

Midterm Feedback

ONE Change

- More flexibility with frameworks vs. step-by-step guide– Project 2 is for flexibility. Project 1 is to learn 1 set of tools and apply
- Step-by-step guide for in-class demos – **there is already written guide as well as recordings for all demos**

Others

- Project 1 requirements not detailed. Sprint requirements. More check-ins. **This course helps you get a step ahead of guided project-based courses (like 2340). More flexible. Rubrics should provide granularity details.**
- Let's have more project-based competitions – **Great suggestion. I will incorporate this next semester – leaderboard based on project code quality etc.**

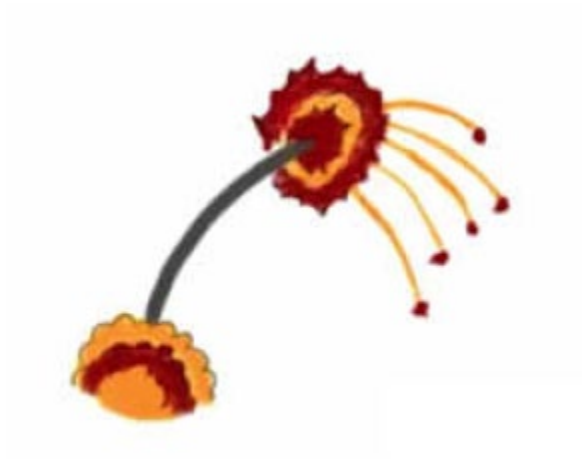
CS3300 Introduction to Software Engineering

Lecture 13: Project 2 Description and Software Testing

Dr. Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

Some Examples…



Ariane 5 Failure:
https://www.youtube.com/watch?v=gp_D8r-2hwk

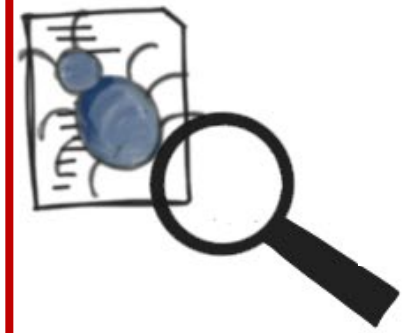
Slide adapted from Alessandro Orso

Testing is a part of Verification and Validation…

Requirements Engineering

Design

Implementation



Verification & Validation

Maintenance

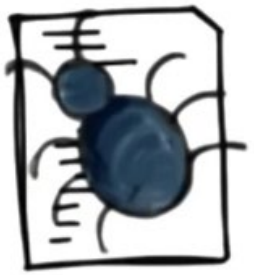
Software is Buggy!

- Cost of bugs: $ 60 B/year
- On average, 1-5 errors per 1K LOC
- Windows 10
  - 50M LOC
  - 63,000 known bugs at the time of release
  - 1.25 per 1,000 lines

- For mass market software 100% correct SW development is infeasible, but
- We must verify the SW as much as possible

Failure, Fault, Error

Failure: Observable incorrect behavior of a program. Conceptually related to the behavior of the program, rather than its code.



Fault (bug): Related to the code. Necessary (not sufficient!) condition for the occurrence of a failure.

Error: Cause of a fault. Usually a human error (conceptual, typo, etc.)

Failure, Fault, Error: Example

```
1.  double doubleValue(int param) {
2.      double result;
3.      result = (double) param * param;
4.      return(result);
5.  }
```

A call to double(3) returns 9. What is this?	The result 9 is a failure- it is an observable behavior
Where is the fault?	Line 3
What is the error that caused the fault?	N/A. Maybe typo, erroneous copy paste, or conceptual. Only the developer knows.

Approaches to Verification

- **Testing** (dynamic verification): exercising software to try and generate failures
- **Static analysis**: identify (specific) problems statically, that is, considering all possible executions
- **Inspections/reviews/walkthroughs**: systematic group review of program text to detect faults
- **Formal verification** (proof of correctness): proving that the program implements the program specification

Testing

Input Domain D

Software

Output Domain O

Test Case: {i ∈ D, o ∈ O}

Test Suite: A set of Test Cases

Static Verification

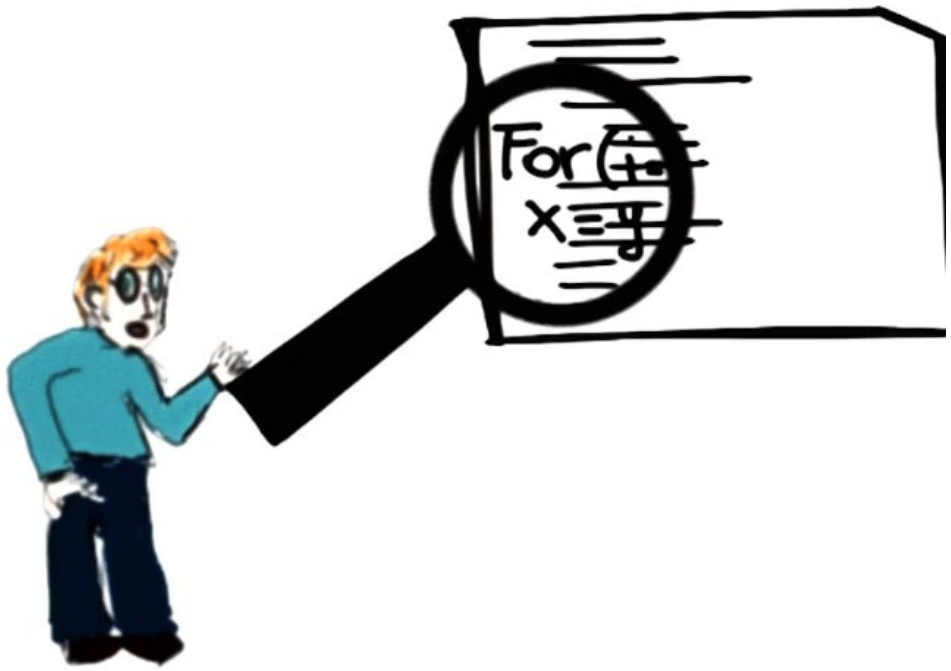
Input Domain D

Software

Output Domain O

Considers all possible inputs (execution/behaviors)

Inspections/Reviews/Walkthroughs



Human intensive activity
Manual
Group activity
Inspect defects in the artifacts by identifying faults

Formal Proof (Of correctness)

Program

Specification

Given a formal specification, checks that the code corresponds to such specification

Sophisticated mathematical analysis

Comparison among the 4 techniques

PROS

CONS

Testing	No False Positives	Highly Incomplete
Static Verification	Considers all program behaviors, Complete	False Positives, Expensive
Inspections	Systematic, Thorough	Informal, Subjective
Formal Proofs of Correctness	Strong Guarantees	Complex, Expensive to build/prove a mathematical basis

Today, Quality Assurance (Verification) is mostly Testing

“50% of my company employees are testers, and the rest spend 50% of their time testing”.

- Bill Gates

What is Testing?

Testing == To execute a program with a sample of the input data

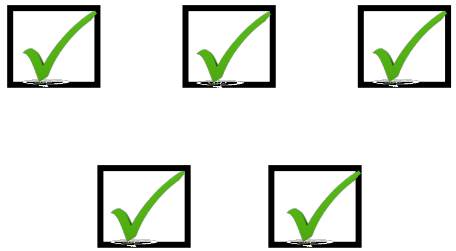
- Dynamic technique: program must be executed
- Optimistic approximation:
  - The program under test is exercised with a (very small) subset of all the possible input data
  - We **assume** that the behavior with any other input is consistent with the behavior shown for the selected subset of input data

Successful Tests

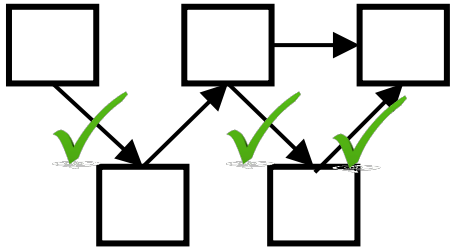
-Goodenough and Gerhart (1985). “Towards a Theory of Test data selection”. *IEEE Transactions of Software Engineering*, Jan 1985

Testing Granularity Levels

Unit Testing

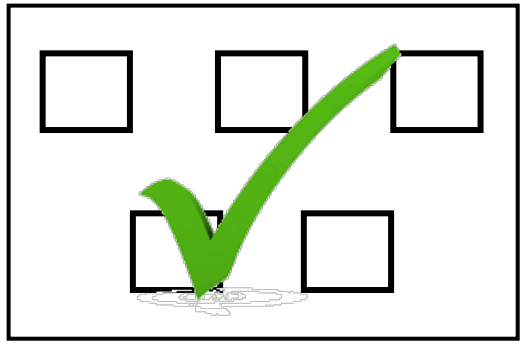


Integration



Big Bang

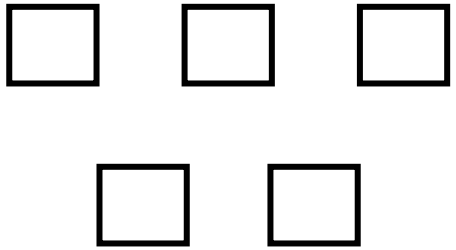
System



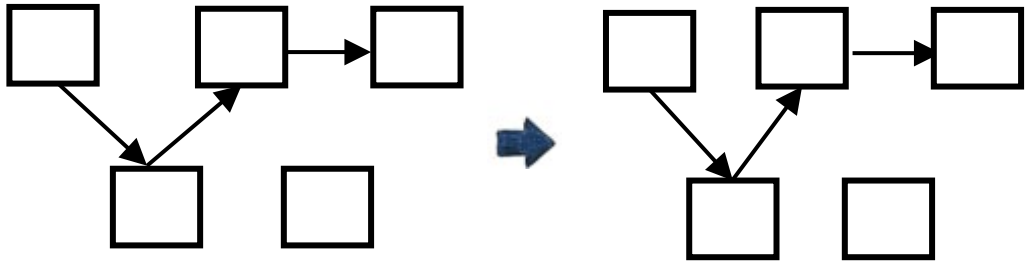
Functional/Non-functional

Customer

Acceptance Testing



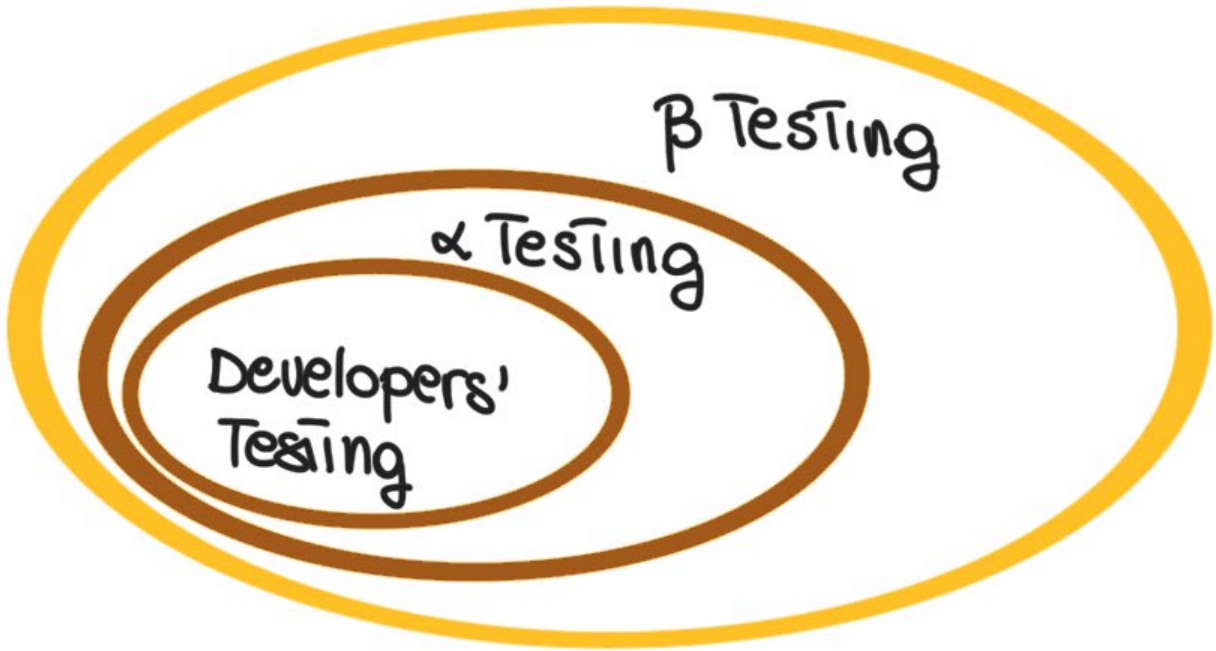
Regression Testing



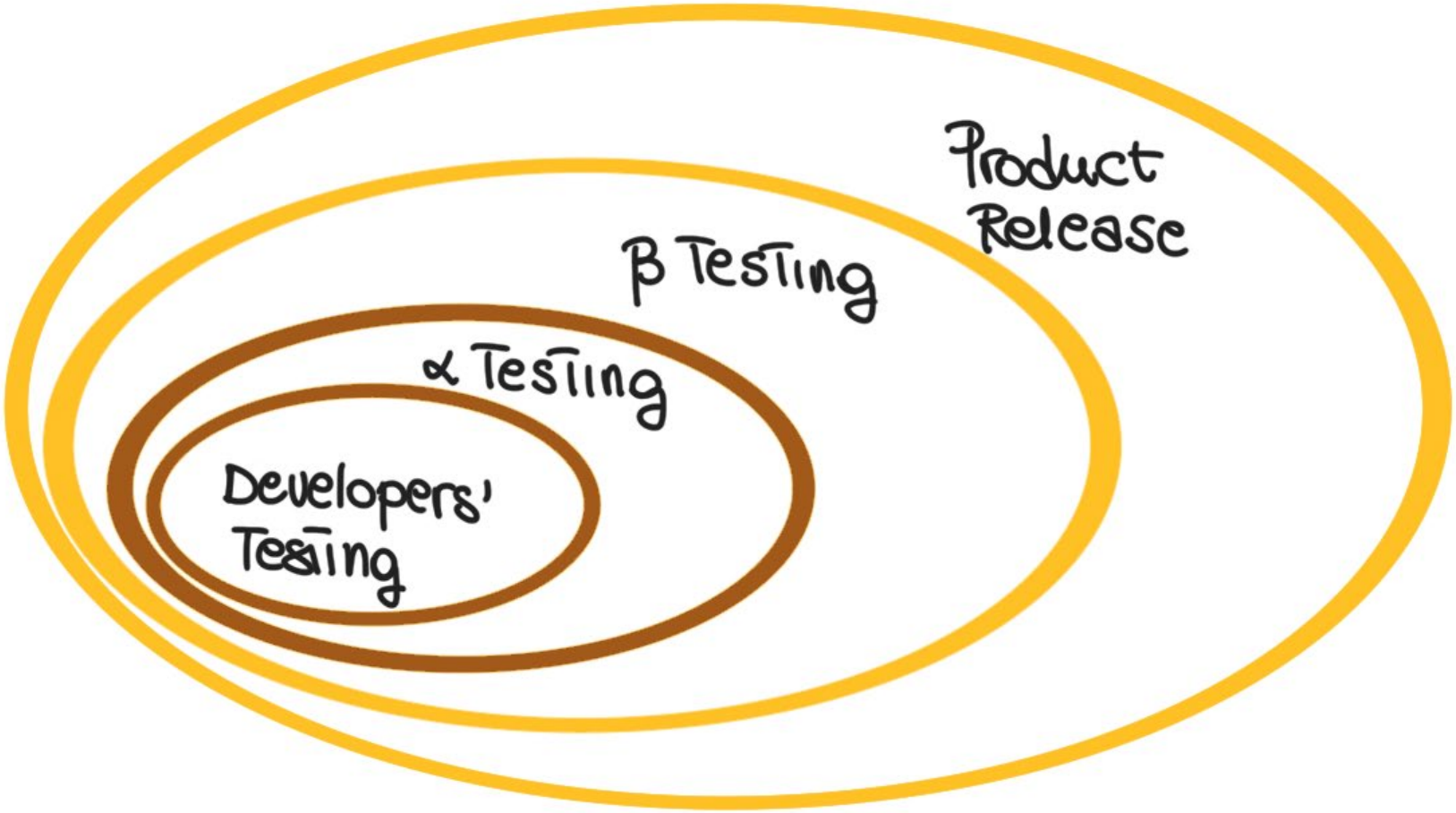
Testing Stages

Testing Stages

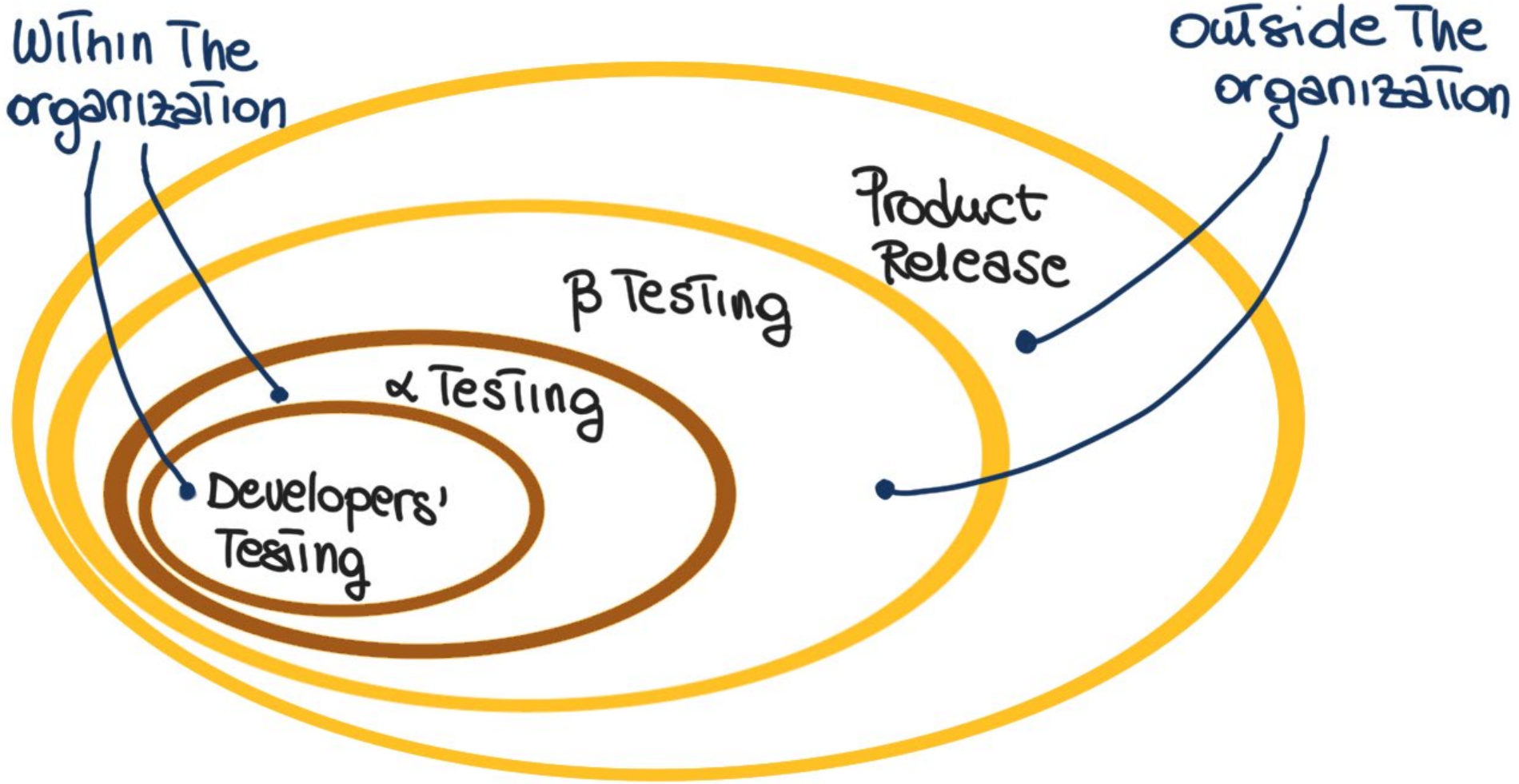
Testing Stages



Testing Stages



Testing Stages



Testing Techniques

There are several techniques

- Different processes
- Different artifacts
- Different approaches

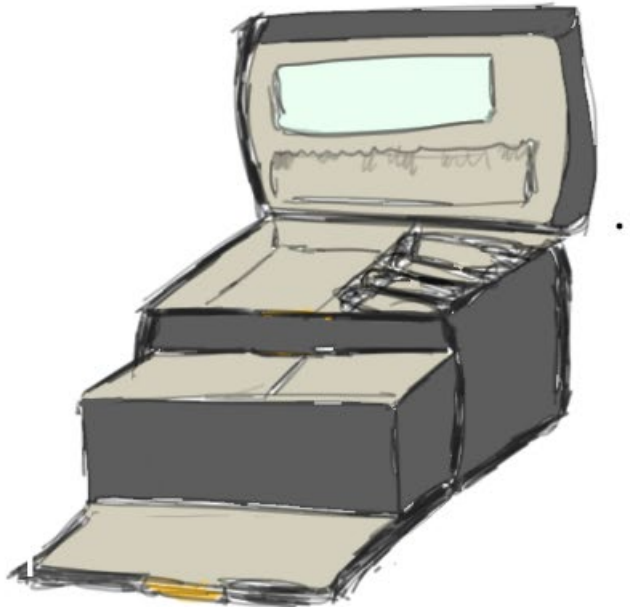
There are no perfect techniques

- Testing is a best-effort activity

There is no best technique

- Different contexts
- Complementary strengths and weaknesses
- Trade-offs

Testing Techniques



BLACK BOX TESTING

- Based on a description of the software (specification)
- Cover as much specified behavior as possible
- Cannot reveal errors due to implementation details

WHITE BOX TESTING

- Based on the code
- Cover as much coded behavior as possible
- Cannot reveal errors due to missing paths

Black-Box Testing Example

Specification: Inputs an integer and prints it

```
1. void printNumBytes (param){
2.      if (param < 1024) printf("%d", param);
3.     else  printf('%d KB" ,  param/124);
4. }
```

Blackbox testing attempts: Inputs +, -, and 0

Will blackbox testing be able to catch the failure? **Most likely Not**

White-Box Testing Example

Specification: Inputs an integer and prints it

```
1. void printNumBytes (param){
2.     if (param < 1024) printf("%d", param);
3.    else  printf('%d KB" ,  param/124);
4. }
```

Whitebox testing attempts: Cover all 4 statements or the 2 paths.
So, <1024, =1024, >1024

Will whitebox testing be able to catch the failure? **Most likely Yes**

Black-box Testing Example: more effective

User Interface (UI) Testing

- Imagine a simple login form for a web application with the following fields: Username (text field), Password (text field), Login Button
- The form is designed to authenticate users based on their input. The requirements specify that: A username must be between 5 to 15 characters; The password must be at least 8 characters long; Special characters are allowed in the password but not in the username.

Why Black-Box Testing Could Be More Effective:

- **Focus on User Behavior**: Black-box testing is effective here because it focuses on how a user interacts with the form rather than the internal logic behind the input validation.
- **Testing Inputs and Outputs**: It can test various scenarios like valid and invalid usernames, short and long passwords, special characters, and empty fields, verifying that the user gets the correct messages like "Username must be at least 5 characters" or "Password is too short."
- **Real-World Use Cases**: Black-box testing ensures that the form behaves as expected in real-world use cases and catches issues like improper error messages, incorrect form submissions, or failures to handle certain types of input.

Black-box Testing Example: more effective

Why White-Box Testing May Be Less Effective:

- **Focus on Code Paths, Not User Experience**: White-box testing in this scenario would focus on the internal validation logic—checking the functions that verify the username length, password length, and special character restrictions.
- **Potential to Miss UI Issues**: It may miss issues related to the user experience, such as how error messages are displayed or whether the form reloads when it shouldn't.
- **Less Emphasis on Usability**: White-box testing might ensure that the functions for validation work perfectly according to the code, but it could overlook how well the user interface guides the user through input errors or displays feedback.

Quizizz

Project 2 Topic

Open Ended

Pick your own topic:

You need to justify that the topic is interesting, relevant to the course, and is of suitable difficulty

- Don’t have a project topic similar to project 1

Should have atleast 4 Minimum Marketable Features

Technologies you are now familiar with

- Google Cloud
- Java Servlet
- REST Platform like Spring Boot
- Frontend development (js,html, css, possibly React…)
- Backend testing
- Debugging
- Working efficiently with IDEs, VCs
- AI tool incorporation in code generation, completion, and UML design

Technologies for Project 2

Anything you want. Can be web-based or android application.

Backend: Build up your expertise in Java/SpringBoot or go for Node.js/others
Frontend: Keep it simple or try something new. React/Angular
Database: Datastore/FireBase/MySQL/mongo
Testing: More points allotted to testing (blackbox/whitebox) in Project 2

Mandatory: GitHub (PR)
Deployment: Google Cloud services to deploy. Build on your cloud expertise.

Bonus Points will be awarded to teams adopting interesting/difficult topics/technologies

Project 2: Requirements

- Should have atleast 4 Minimum Marketable Features

- We will announce bonus points criteria soon
  - Completing AI survey at end of semester or extensive blackbox testing
  - Containerization/docker
  - More than 4 MMFs/Difficult/Innovative topic
  - Extensive tools or AI integration accomplished in whitebox and blackbox testing along with unit, integration and system testing.

- Have to use GitHub
  - PR and readme requirements same as project 1
- Last Assignment - Test is based on Project 2. So focus on thorough testing

Mandatory for Project 2

- 4 MMFs – get approval from mentors in project 2 touchpoint
- Code Review Requirements
- Compliance with 3 design patterns (Which pattern and why applicable?)
  - Applicable to Python, Java, Kotlin, Scala, C#, Ruby, PHP
  - Inform the Instruction team on Ed if your team is attempting functional programming (Haskell, Erland, F#) or procedural programming (C, Pascal, in which case this requirement for your team will be revised accordingly
  - Even if you are only using JS, TypeScript and ES6 classes support classic OOP patterns
- Good testing done – blackbox and whitebox.
  - Relevant to testing assignment and project 2 ppt
- GitHub
- GCP
- AI integration – implementation (atleast 1 tool based on Project 1 experience) and testing (research with different tools)

GitHub

- Make sure to have your GitHub repository set for this project
- Ensure it is private
- It is important for your future reference
- Add it to your resume
- GitHub pages can be very compelling for employers
  - You will use GitHub pages to create your Project Report

Project 2 Topic and MMF assignment

MMF and MVP

Substantial piece of functionality that delivers business value to customers

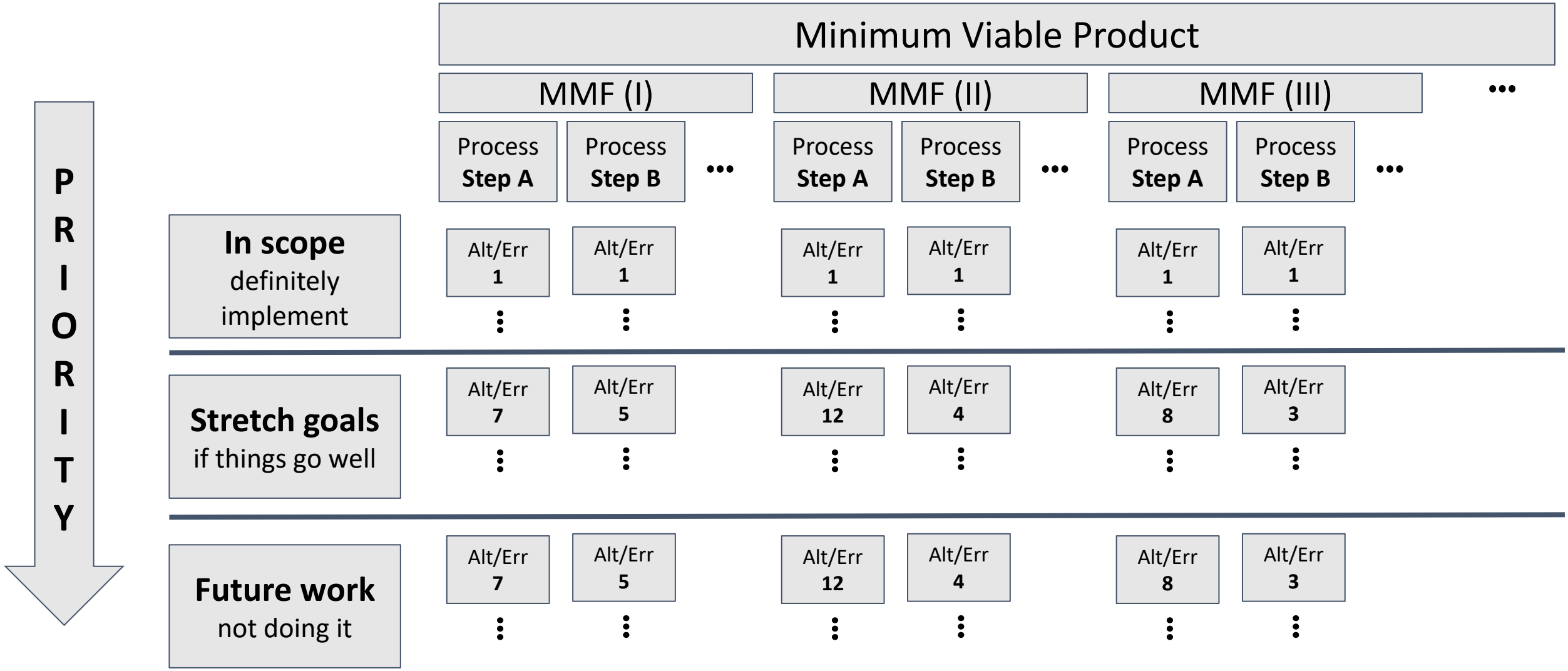
- Should comprise multiple small user stories

An MVP comprises several MMFs

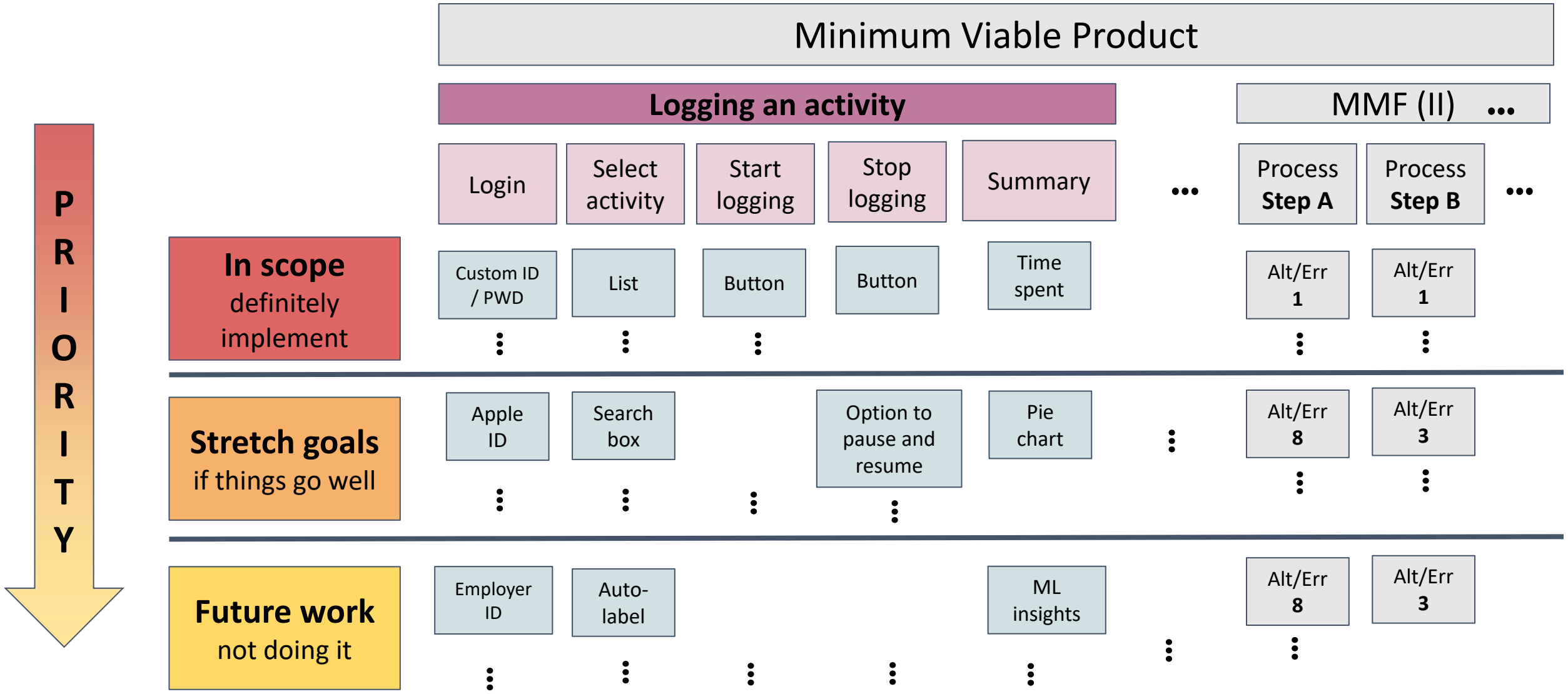
For LocationFinder, MMFs were:

- Basic Display – Input and Output
- Interactive Map Display – hover functionality
- Advanced Sorting, Searching and Filtering functionalities
- Change Map pin color for example is not a MMF

MMF and MVP


Minimum Viable Product
MMF (I)
MMF (II)
MMF (III)
Process Step A
Process Step B
PRIORITY
In scope
definitely implement
Stretch goals
if things go well
Future work
not doing it
Alt/Err 1
Alt/Err 7
Alt/Err 5
Alt/Err 12
Alt/Err 4
Alt/Err 8
Alt/Err 3

Example: Fitness app



Presentation & report

- 13 Groups
- 2 days of presentation
- Each Team will have 10 minutes time
- + 1 minute Q&A
- 7 Teams per day
- Audience polling and asking questions is part of Project 2
- **Make sure to have a demo**
- **Project 2 Report should be deployed on GitHub Pages**
- **All Project 2 related assignments will be published soon**