Software Engineering

Juan Zhai

Juanzhai@umass.edu

Recap: course workload and grading

- **20%** Homeworks [Individual or Group]
- **40%** In-class exercises [Group]
- **30%** Final project [Group]
- **10%** Participation [Individual]

- A (93-100), A- (90-92),
- B+ (87-89), B (83-86), B- (80-82),
- C+ (77-79), C (73-76), C- (70-72),
- D+ (67-69), D (60-66),
- F (0-59).

Recap: Participation

- Each student needs to attend the following classes to promote successful teamwork: 4 exercises, 2 project fairs

- If you have extenuating circumstances (e.g., illness, car accident, jury duty, military service), you should contact the instructor or TAs about obtaining an excused absence with written documentation when necessary.

Recap: Software Engineering

- **What is Software Engineering?**
  - The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.
- **Why is it important?**
  - Software is everywhere and complex.
  - Software defects are expensive and annoying.
- **Goals:**
  - Decompose a complex engineering problem.
  - Organize processes and effort.
  - Improve software reliability.
  - Improve developer productivity.

Final project

- Team up and select a project by March 12, 2024, 9:00 AM EST
- Submit team information through the first assignment “Project topic selection and teaming up” on Canvas, including all members' names, emails and netIDs, and the team leader's name.
- Check the three options in the document Final-project.pdf attached in the first assignment on Canvas.

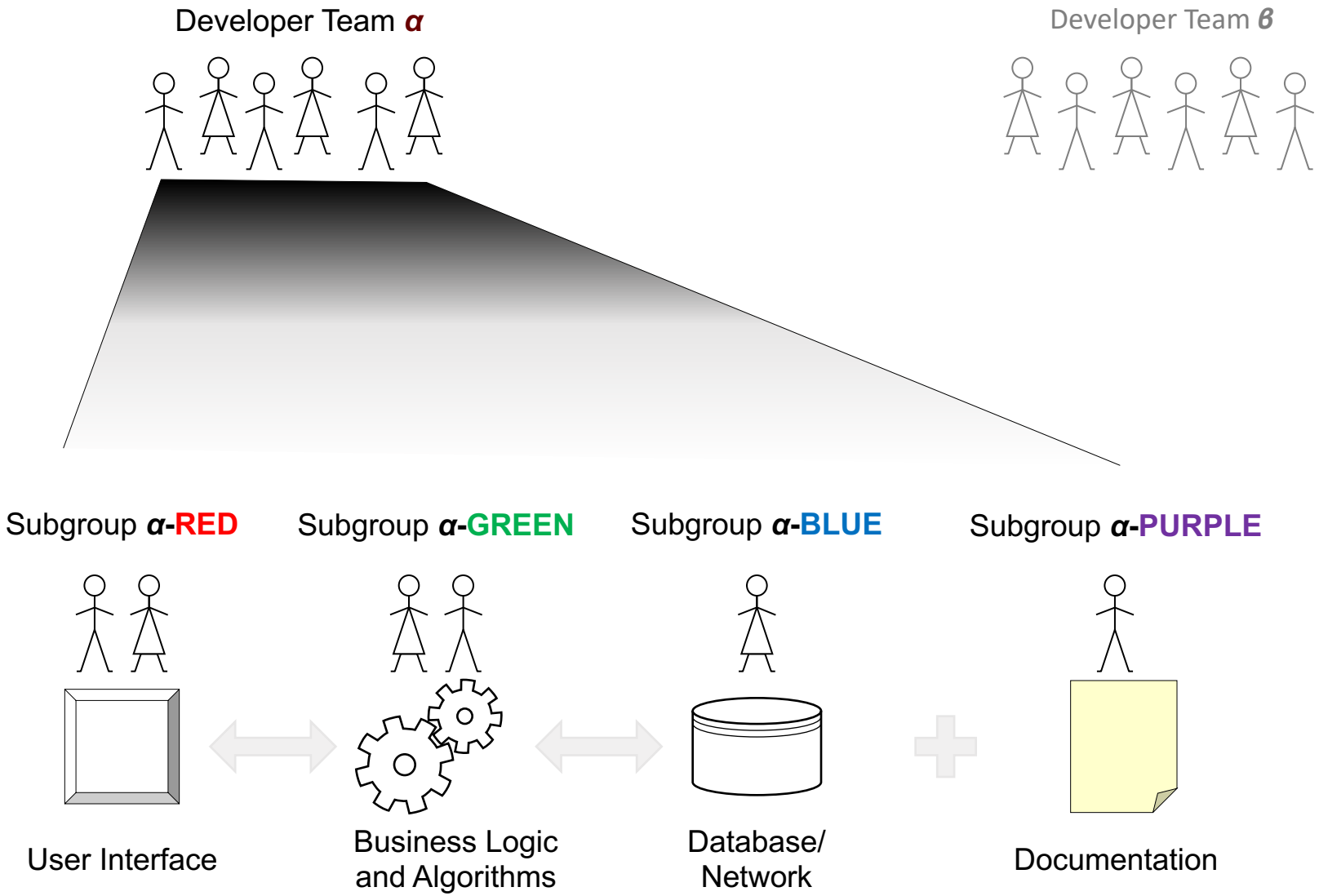
Teamwork

Common problem-solving strategy: divide-and-conquer

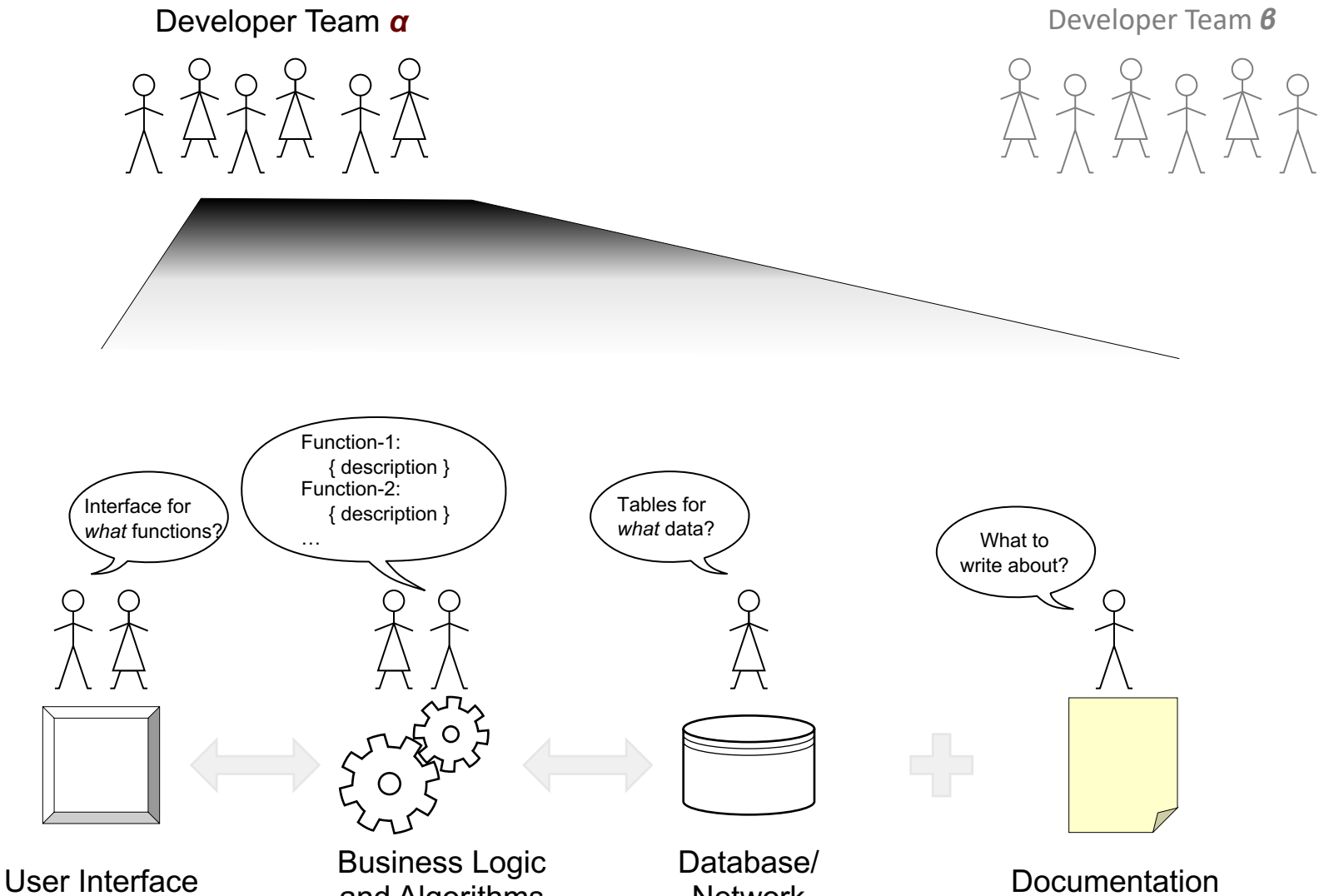
Example: Restaurant Automation Requirements

- REQ-1: Support the work of **floor staff**
- REQ-2: Support the work of **kitchen staff**
- REQ-3: Support the work of **management**

Decomposition by Partitioning the Solution



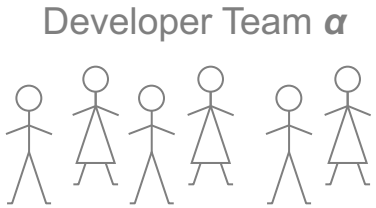
Communication Overhead



Drawbacks of Team *α* Approach

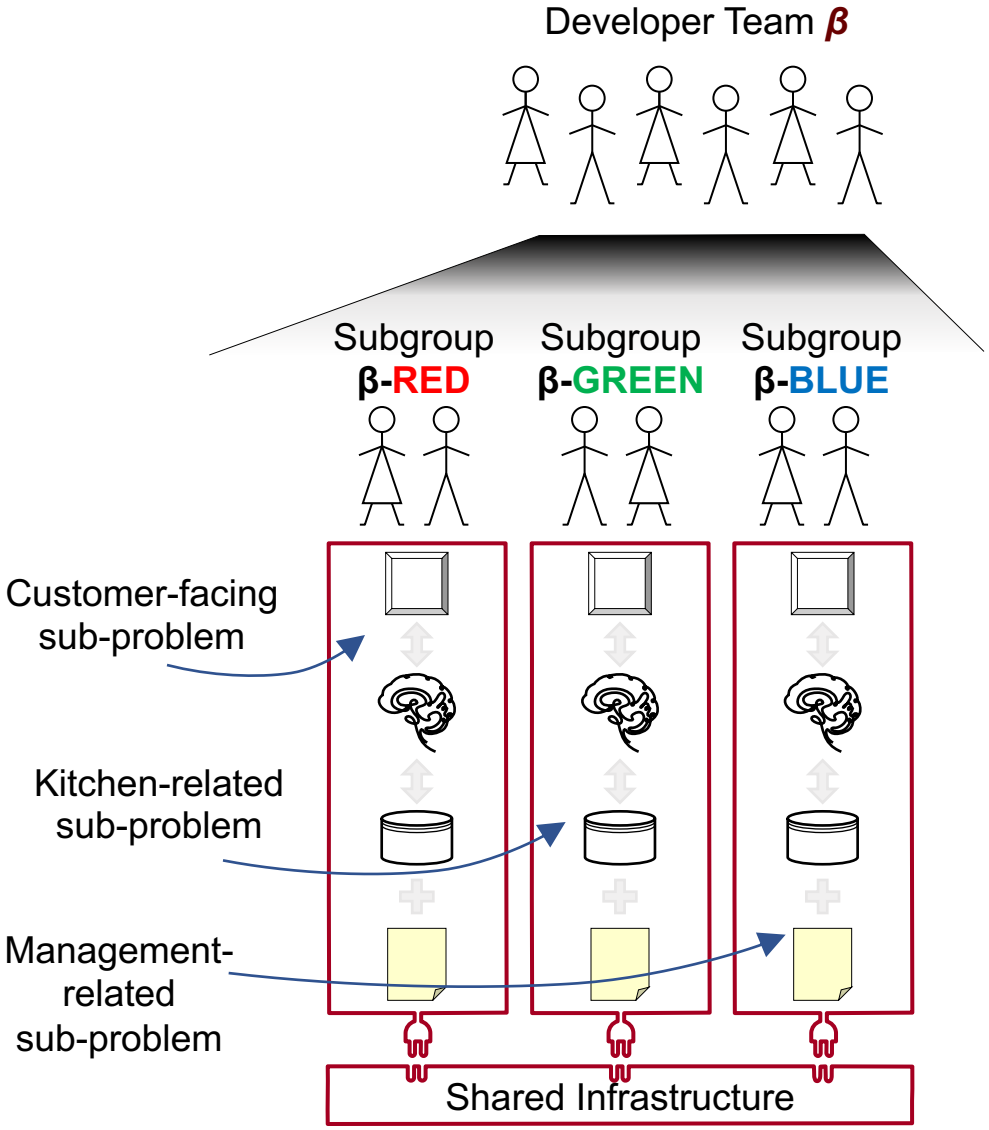
- Long feedback loop:
  - The time to implementation depends on the slowest sub-group
- Communication overhead:
  - Most time spent in communication and “documentation writing”
- Failure-prone:
  - If any sub-group doesn’t deliver, the whole project fails
- Each student learns only one aspect of software development

Decomposition by Projection to Sub-problems

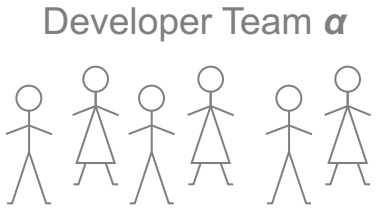


- Different way of "slicing" the project:
  - Instead of by solution, we have "stacks", by sub-problems
- Each "stack" can be done independently of other stacks, as a mini-project, *because it solves a different sub-problem*!

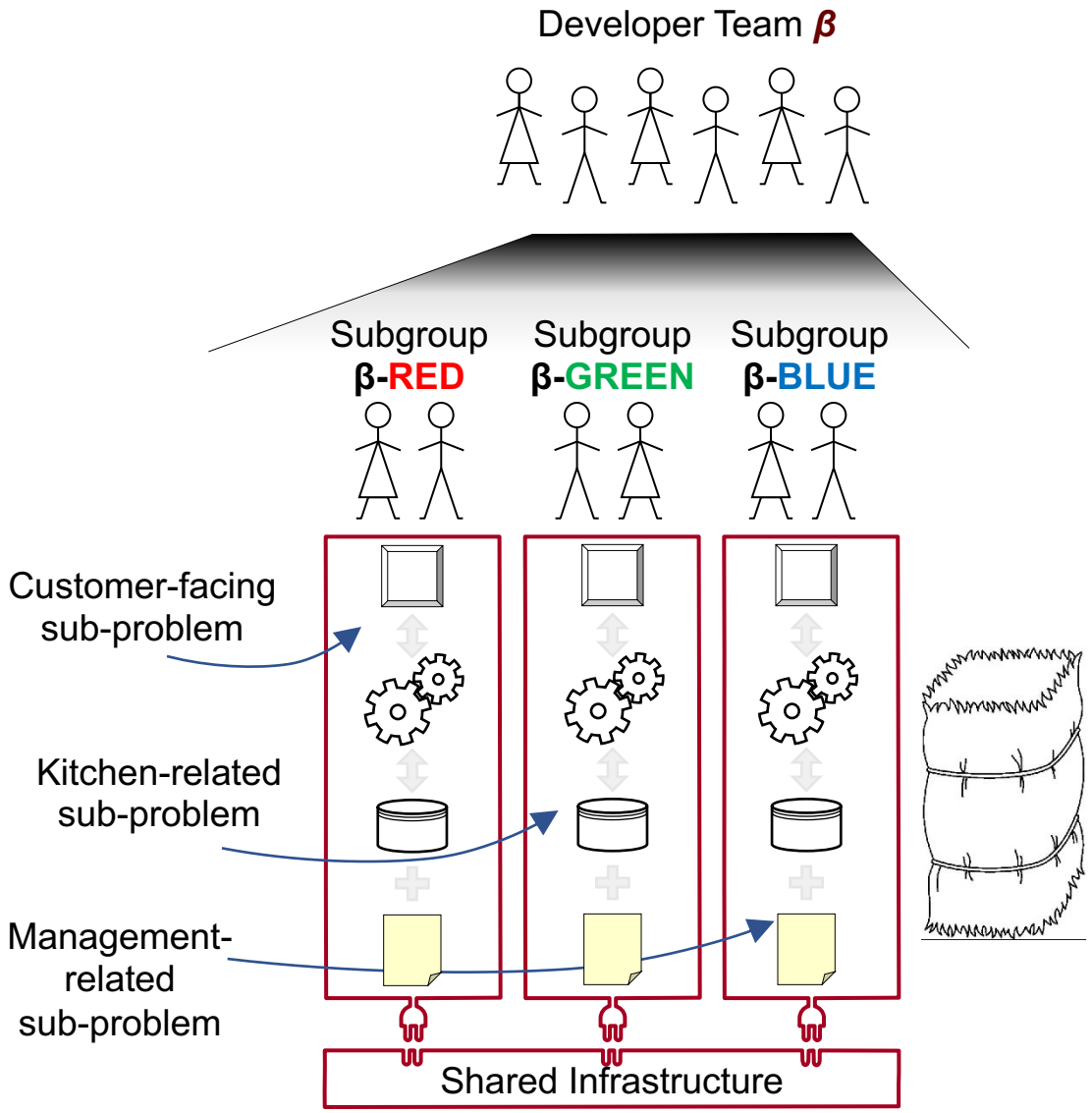
➔ Separation of concerns!



Decomposition by Projection



- Team communication is simple
- They only need to define **shared interfaces** (“APIs”) and can focus on creative software development (sub-problem solving!)
- What is inside of each “stack” is not discussed with other sub-teams —for others, the contents of your “stack” is hidden—they see a black box with defined interface / APIs (“information hiding”)



Benefits of Team β Approach

- Increased productivity
  - Minimizes the amount of communication needed to coordinate the work
- Failure resilience
  - Reduces the dependency on the rest of the team—if they fail to deliver, you can still succeed and demonstrate your mini-project
- Each student learns all aspects of software development

What is “Shared Infrastructure”?

- Needs to be discovered, as we will learn in subsequent lectures
- Examples of shared infrastructure:
  - A relational database, so the whole team needs to design the shared tables
  - Data items communicated via messages, so the whole team needs to discuss the communication protocol and message format