Terminologies

A mini-lecture series

CSE498 Collaborative Design (W) - Secure and Efficient C++ Software Development

02/10/2025

Kira Chan

https://cse.msu.edu/~chanken1/

Stakeholders

- Parties involved in the development of the software

Stakeholders (Examples)

- Lawyers
- Developers (you)
- Testers (IVV, performance team, security)
- Managers
- Government Agencies

- Customers
- Users
- Distinction: **Customers pay for the software. Users use the software**

Requirements

- Defines the behavior of a software
- Enumerated list of “The system shall do x”
- Often, you have to gather the requirements from your customers
- This process is called *Requirements Engineering*
  - Understand the needs of the customer to build the software that suits their needs

Functional vs NonFunctional Objectives

- Functional objectives describe what the system should do
  - The system must verify the user before showing their payroll information
  - The autonomous vehicle must reach its destination
- Nonfunctional objectives describe the general property of the system
  - The payroll information should load within 0.5 seconds
  - The autonomous vehicle should exhibit safe driving styles

“System Behavior”

- You can observe the behavior of the software developed as a whole
- When you observe the software, what type of *behavior* does it exhibit?

- We want to ensure *correctness* and *safety*

Operating Context

- The environment context in which your software is deployed in
- Could be just an app (not interacting with environments)
- Could be a cyber physical system deployed (think autonomous vehicle)

Uncertainty

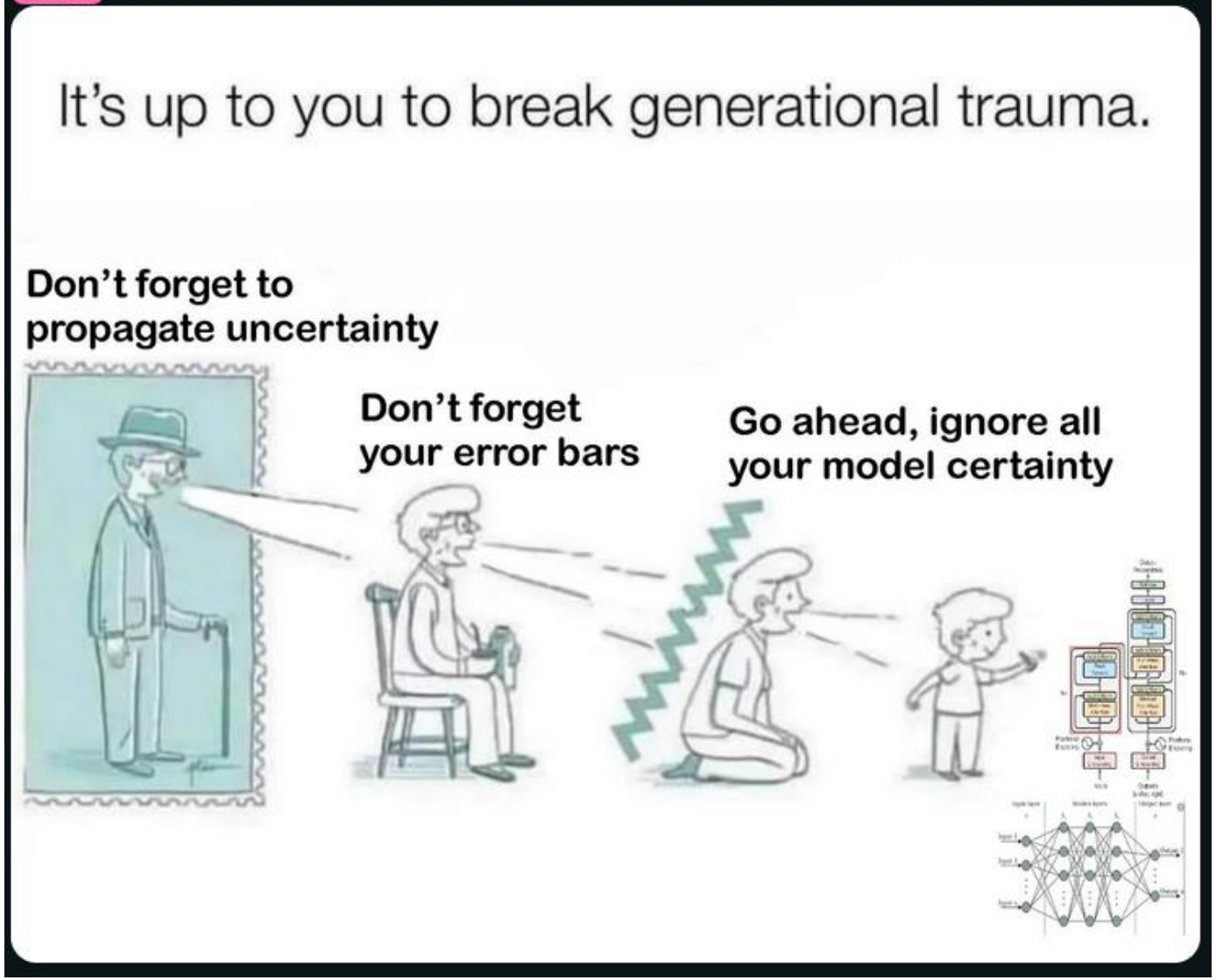
- Once you have developed the requirements of the software, operating context which you have not seen before form **uncertainty**

- Aleatoric uncertainty: Also known as stochastic uncertainty. They are unknowns that change each time we run the same experiment
  - Uncertainty in data, noise, and intrinsic randomness
- Epistemic uncertainty: Also known as systematic uncertainty. Caused by things one could known in principle but does not in practice
  - Lack of knowledge of information
  - Thus, can be fixed by gaining more information

r/mathmemes • 1 yr. ago
TobyWasBestSpiderMan

These kids probably don't even know the difference between Aleatoric and Epistemic certainty!

Statistics



Design time vs Run time

- Design time denotes the development process
  - When the software is being designed, built, and tested
- Run time denotes after the product is deployed

- You might have heard of the term run-time error

Design time vs Run time

- You would like to be able to address as much “hiccups” as you can during design time
- Uncertainty that can be addressed should be explicitly addressed

- If you can enumerate the uncertainties during design time, then you can better take steps to address it (avoid, mitigate, etc.)

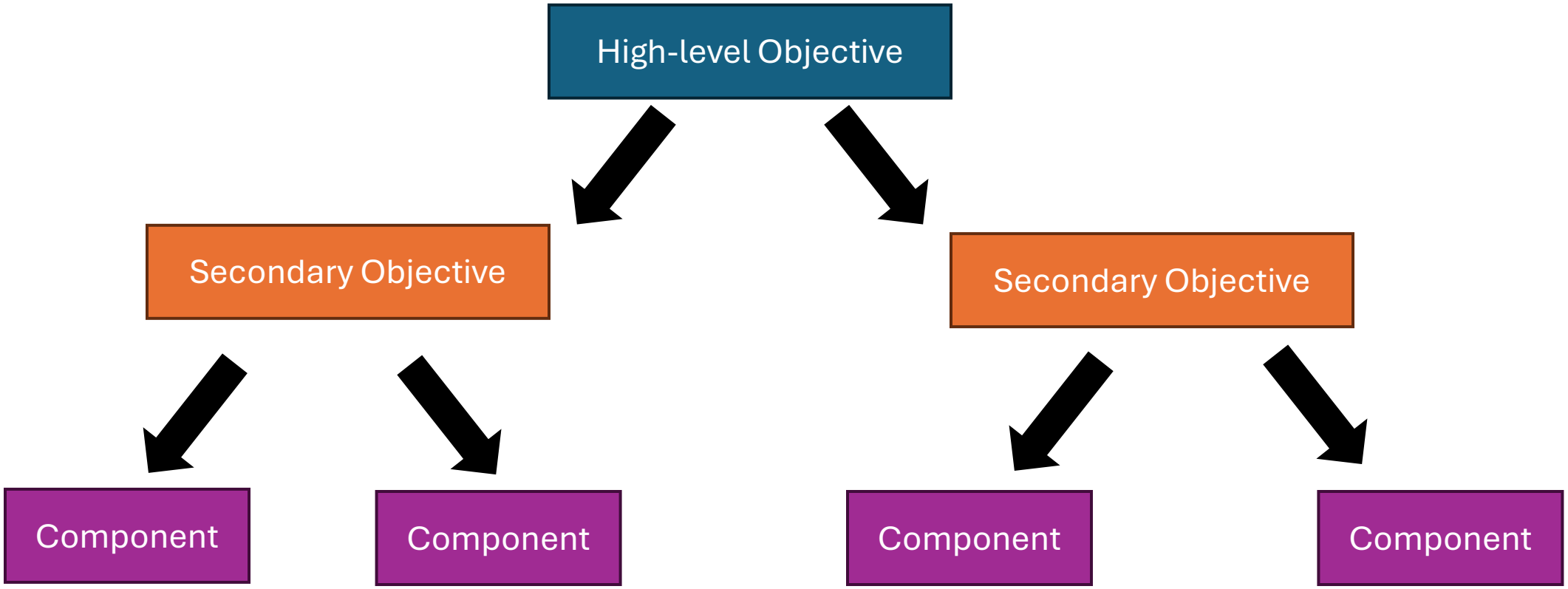
Determinism and Non-Determinism

- If I run a program 10 times, if the outcome is exactly the same all 10 times, it is *deterministic*
- If I run it 10 times, if any of the outcome is different, it is non-deterministic
  - (Undefined behaviour)

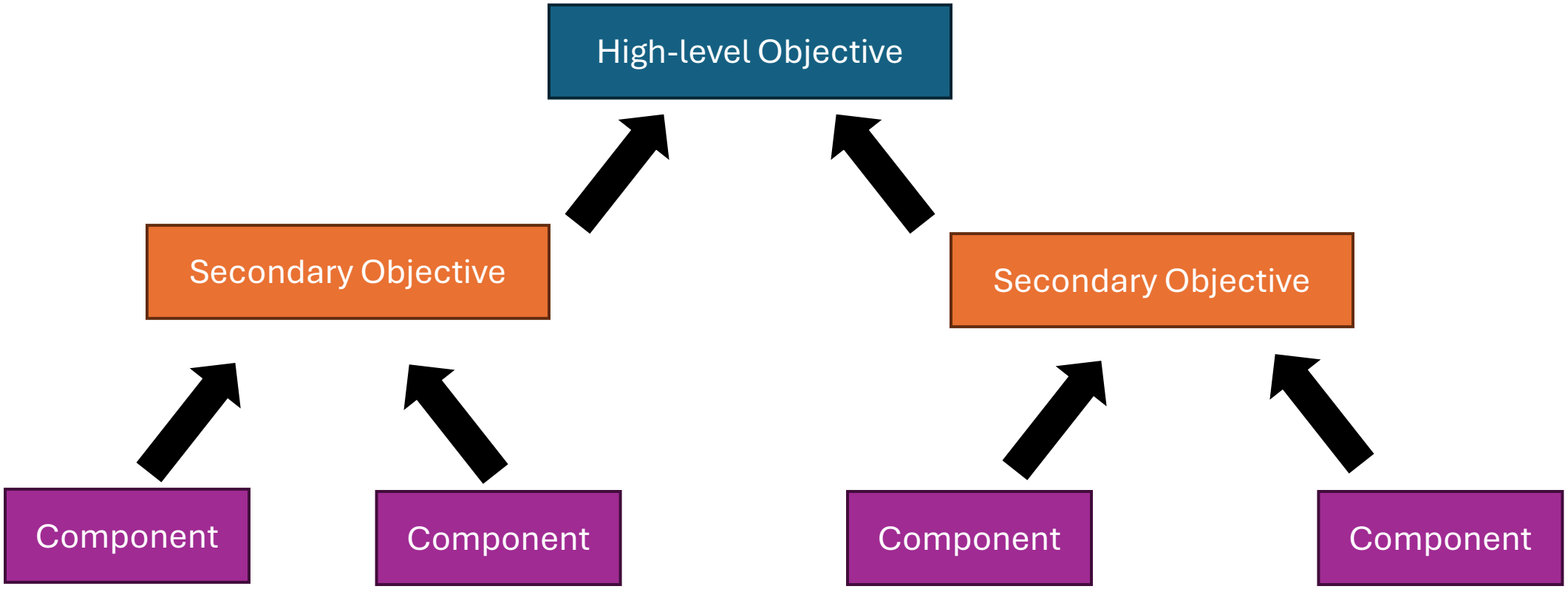
On building a system

- Top-down approach
- Bottom-up approach

Top-down


High-level Objective
Secondary Objective
Secondary Objective
Component
Component
Component
Component

Bottom-up


High-level Objective
Secondary Objective
Secondary Objective
Component
Component
Component
Component

Website example

- Top-down approaches are really good when the problem is well defined or well know
  - If the customer knows exactly what type of website they want and how it should look, function, etc.
  - Can be challenge if the top-level objective is not really known
- Bottom-up approaches are good when the problem is more nebulous
  - If the customer doesn’t really know what they want, and its your job to find out
  - Build basic structures, containers, parts of the website. Show them and change from there

Person of the day
Grace Hopper

- First to devise machine independent programming languages
  - COBOL
- Developed the first compiler
- Found the first ``bug'', and thus popularised the term