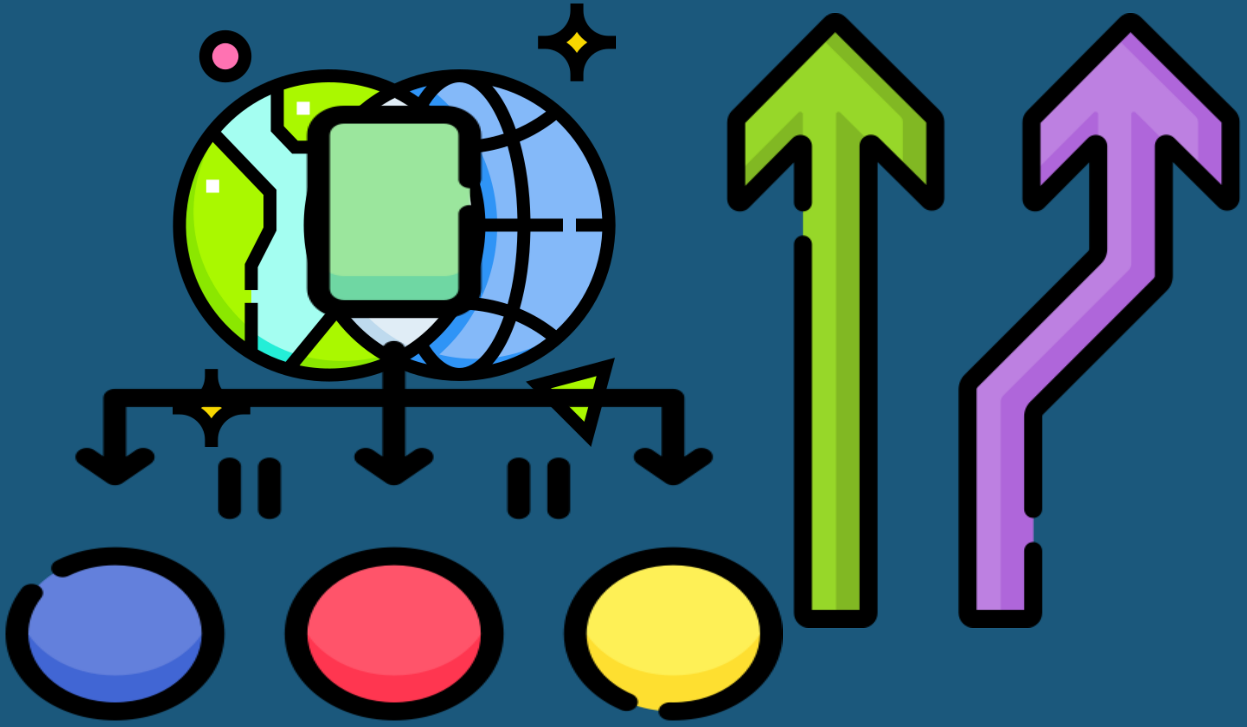


Parallel and Distributed Computing

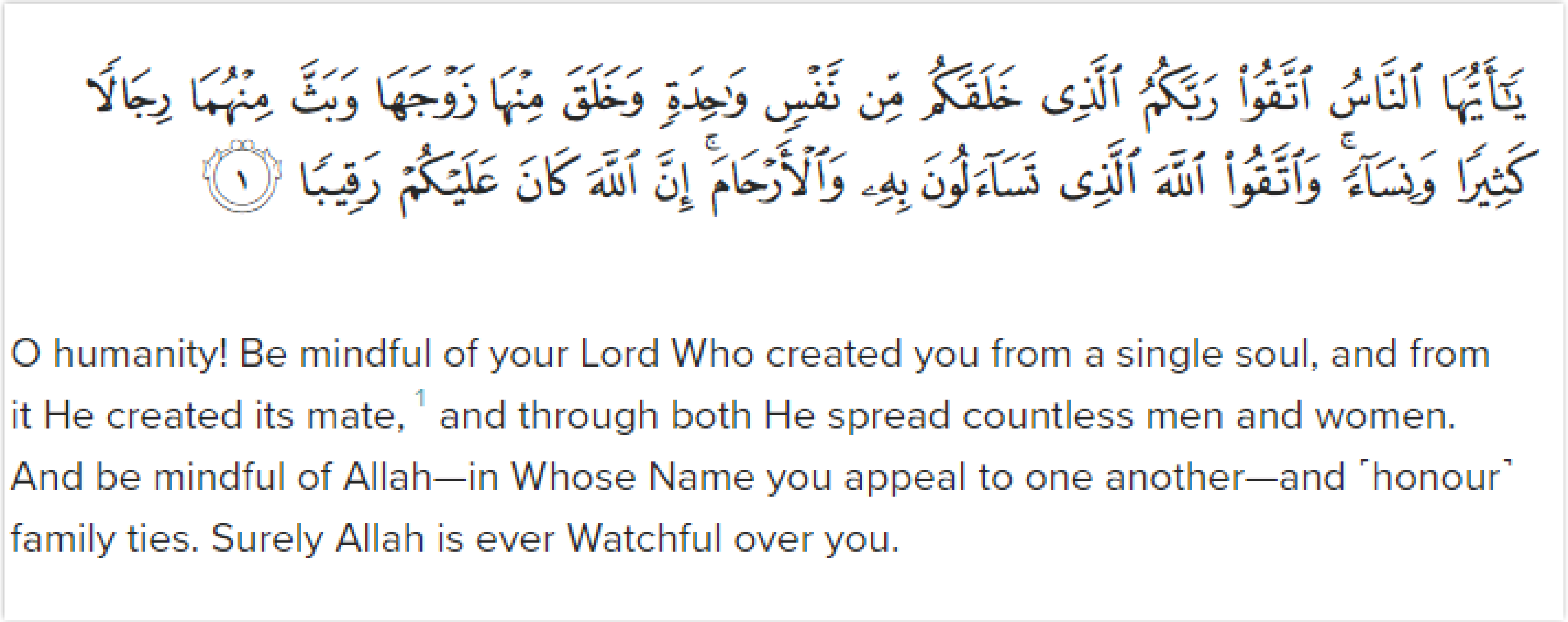
Mr. Hassan Sardar

Lecturer, Department of Computer Science

COMSATS University Islamabad, Wah Campus

Email: hassan@ciitwah.edu.pk

Lesson



يَـٰٓأَيُّهَا ٱلنَّاسُ ٱتَّقُوا۟ رَبَّكُمُ ٱلَّذِى خَلَقَكُم مِّن نَّفْسٍ وَٰحِدَةٍ وَخَلَقَ مِنْهَا زَوْجَهَا وَبَثَّ مِنْهُمَا رِجَالًا كَثِيرًا وَنِسَآءً ۚ وَٱتَّقُوا۟ ٱللَّهَ ٱلَّذِى تَسَآءَلُونَ بِهِۦ وَٱلْأَرْحَامَ ۚ إِنَّ ٱللَّهَ كَانَ عَلَيْكُمْ رَقِيبًا ١

O humanity! Be mindful of your Lord Who created you from a single soul, and from it He created its mate,[1] and through both He spread countless men and women. And be mindful of Allah—in Whose Name you appeal to one another—and ˹honour˺ family ties. Surely Allah is ever Watchful over you.

Today' Lecture

WAH Campus

Data design and performance models

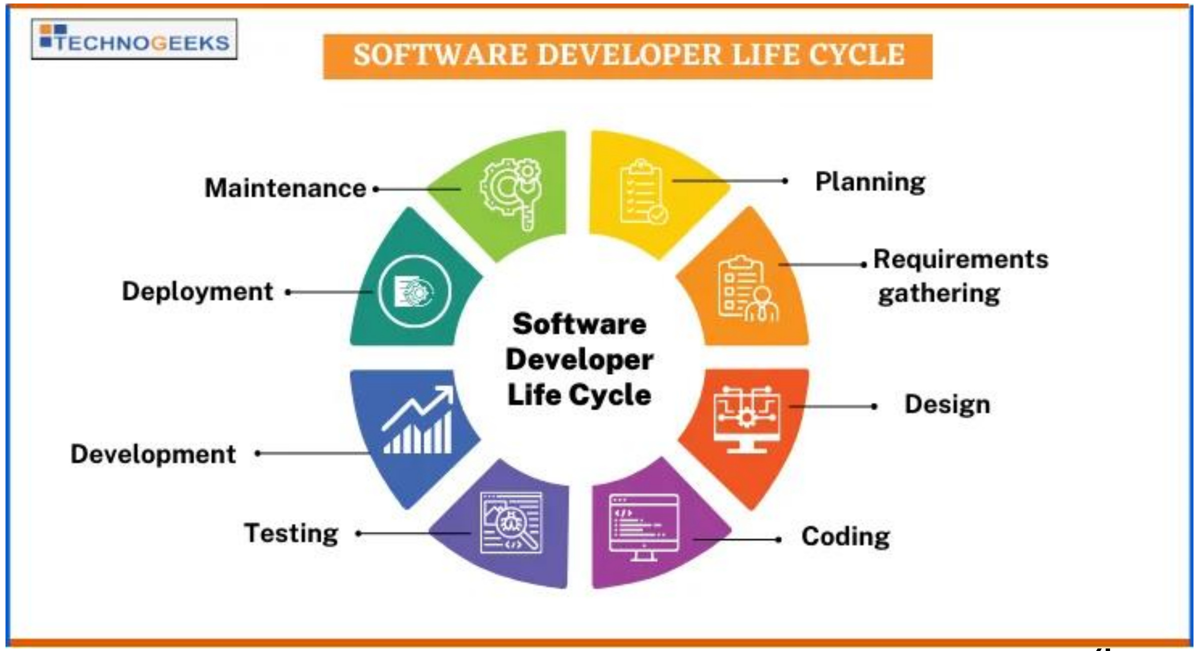
- Performance Models driven by Data Movement
- Data Structure Design and its impact on performance
- Why real applications struggle to achieve performance
- Addressing kernels and loops that significantly underperform
- Choosing data structures for your application
- **Assessing different programming approaches before writing code**
- **Understanding how the cache hierarchy delivers data to the processor**

Importance of Assessing Programming Approaches

WAH Campus

Why Assess Programming Approaches?

- **Content:**
- **Definition**: Evaluating different programming strategies, methodologies, or paradigms to identify the best-fit approach before starting development.
- **Purpose**:
  - Ensures alignment with project goals and requirements.
  - Reduces risks of project failure and technical debt.
  - Saves time and resources in the long run.
- **Benefits**:
  - Better code quality and maintainability.
  - Improved efficiency and reduced complexity.
  - Adaptability to changes in project scope.

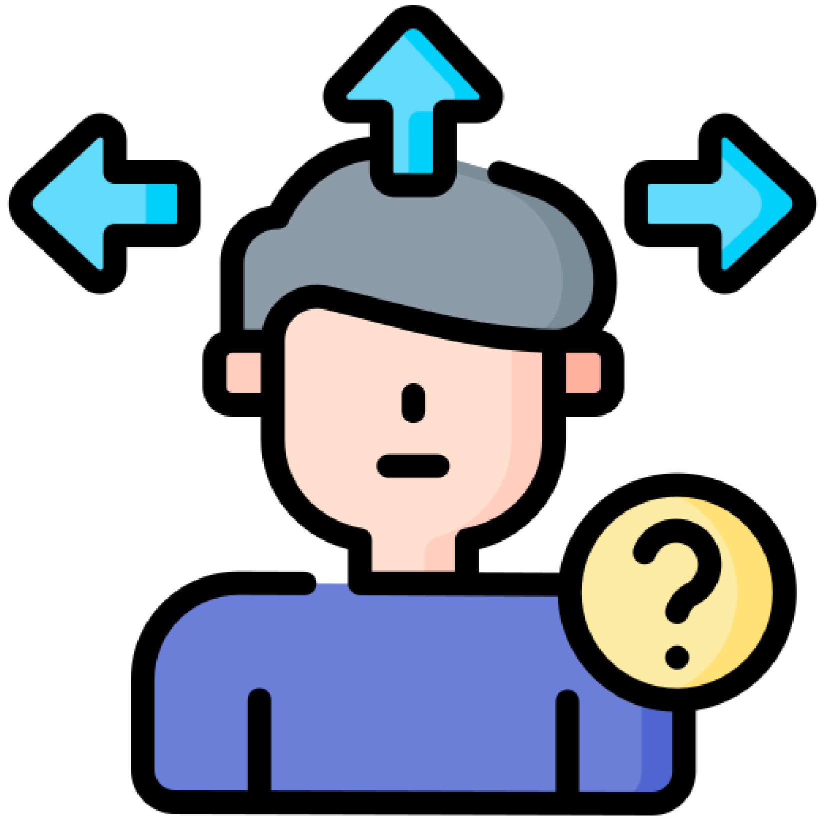


Factors to Consider When Choosing an Approach

WAH Campus

Key Factors to Evaluate Programming Approaches

- Project Requirements: Understand the specific needs, goals, and constraints of the project. Example: If the project is data-driven, should you use OOP or Functional Programming?
- Team Expertise: Consider the programming languages and frameworks your team is most comfortable with. Example: Choose between Python and Java based on team expertise.
- Development Speed vs. Quality: Determine whether the focus is on rapid development or long-term maintainability .E.g., Agile vs. Waterfall for project management.
- Scalability & Performance: Will the chosen approach support scaling as the project grows?
- Budget & Timeline: Consider the available resources, budget, and time constraints. A simpler approach like scripting may be suitable for limited budgets.



Common Programming Paradigms and Their Use Cases

WAH Campus

- **Object-Oriented Programming (OOP)**:

•**Use Case**: Large-scale software, systems with complex interactions.
•**Advantages**: Modularity, code reuse, easy maintenance.
•**Example**: Java, C++, Python.

- **Functional Programming**:

•**Use Case**: Data processing, mathematical computations, parallel programming.
•**Advantages**: Immutability, no side effects, easier debugging.
•**Example**: Haskell, Scala, JavaScript.

- **Procedural Programming**:

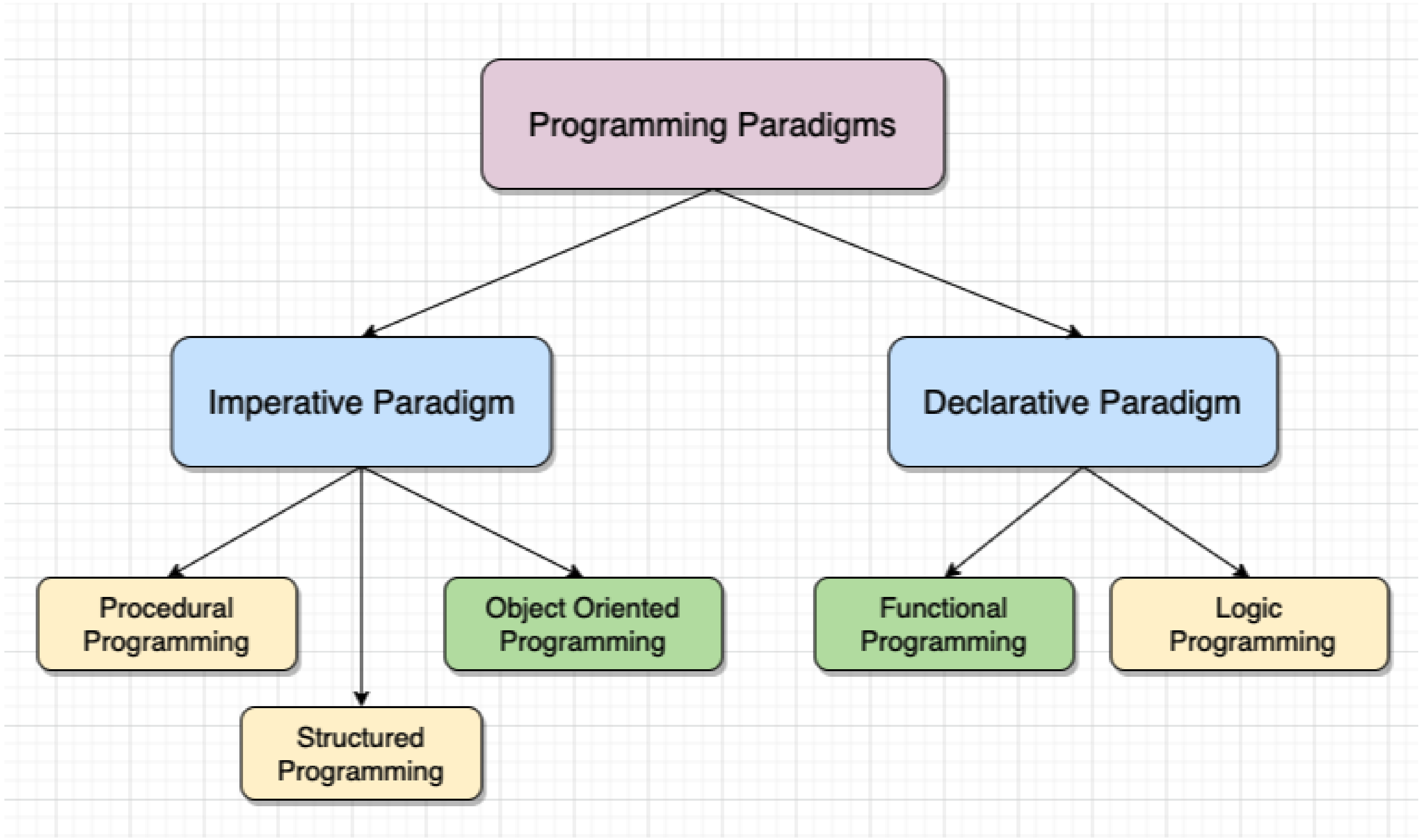
•**Use Case**: Small projects or scripts, clear linear flow.
•**Advantages**: Simple structure, straightforward implementation.
•**Example**: C, BASIC, Python.

- **Event-Driven Programming**:

•**Use Case**: GUI applications, real-time systems.
•**Advantages**: Responsive UI, modular structure.
•**Example**: JavaScript, C#, Node.js.

Common Programming Paradigms and Their Use Cases

WAH Campus



Common Programming Paradigms and Their Use Cases

WAH Campus

What are Programming Paradigms?

- Definition: Programming paradigms are different styles or approaches to programming. They dictate how code is structured, organized, and executed.
  - Two Main Paradigms: Imperative Paradigm: Focuses on how to achieve a task using step-by-step instructions.
  - Declarative Paradigm: Focuses on what the result should be without specifying how to achieve it.
- Imperative Paradigm
  - An approach that involves giving the computer a sequence of instructions to perform a task.

  Characteristics:
  - Focuses on control flow through statements like loops and conditionals.
  - The programmer specifies how to achieve the result.

Common Programming Paradigms and Their Use Cases

WAH Campus

Subcategories of Imperative Paradigm:

- Procedural Programming:

Definition: A style of programming that uses procedures or functions to organize code.

Characteristics: Uses sequences of instructions, conditionals, and loops. Emphasizes a top-down approach.

Examples: C, Pascal, BASIC.

- Object-Oriented Programming (OOP):

Definition: A style of programming that organizes code using objects and classes.

Characteristics: Emphasizes encapsulation, inheritance, and polymorphism.

Examples: Java, C++, Python.

- Structured Programming:

Definition: A type of procedural programming that emphasizes breaking down a program into smaller, manageable functions.

Characteristics: Uses control structures like sequence, selection (if/else), and iteration (loops).

Examples: C, Fortran.

What is the difference between function and procedure?

Common Programming Paradigms and Their Use Cases

Declarative Paradigm

Definition: A programming style whereprogrammer defines what the outcome should be rather than how to get there.

Characteristics:

- Focuses on declarations rather than instructions.
- Relies heavily on abstractions and expressions.

Subcategories:

Functional Programming:

Definition: A style of programming where computation is treated as the evaluation of mathematical functions.

Characteristics: Avoids mutable data and state changes. Uses recursion instead of loops.

Examples: the Haskell, Lisp, Scala.

Logic Programming:

Definition: A type of programming based on formal logic, where you specify the rules and relationships.

Characteristics: Focuses on the logic of the program. Uses facts, rules, and queries.

Examples: Prolog, Datalog.

Comparing Imperative and Declarative Paradigms

WAH Campus

Imperative Paradigm:

- Specifies the steps to solve a problem.
- Focuses on control flow.
- Commonly used in system-level programming.

- **Declarative Paradigm:**
- Specifies the desired result without detailing steps.
- Emphasizes expressions and logic.
- Commonly used in data processing, AI, and web development.

Best Practices for Assessing and Choosing an Approach

WAH Campus

Best Practices for Selecting a Programming Approach

- Conduct Requirement Analysis: Clearly document the project's needs and define the scope.

  Tools: Use requirement-gathering tools like JIRA or Confluence.
- Prototype: Create small prototypes to test different approaches.

  Tools: Use rapid prototyping tools like Figma for UI/UX or simple code mock-ups.
- Consult with Stakeholders: Discuss the options with the team, management, and clients to ensure all perspectives are considered.

  Tools: Use collaborative platforms like Slack or Microsoft Teams.
- Evaluate Tools and Libraries: Choose the right tools, frameworks, and libraries that support the chosen approach.

  Example: If choosing Microservices, consider using Docker and Kubernetes.
- Consider Future Scalability and Maintenance: Ensure that the chosen approach aligns with future growth and maintenance needs.

  Example: Cloud-based vs. on-premise solutions.