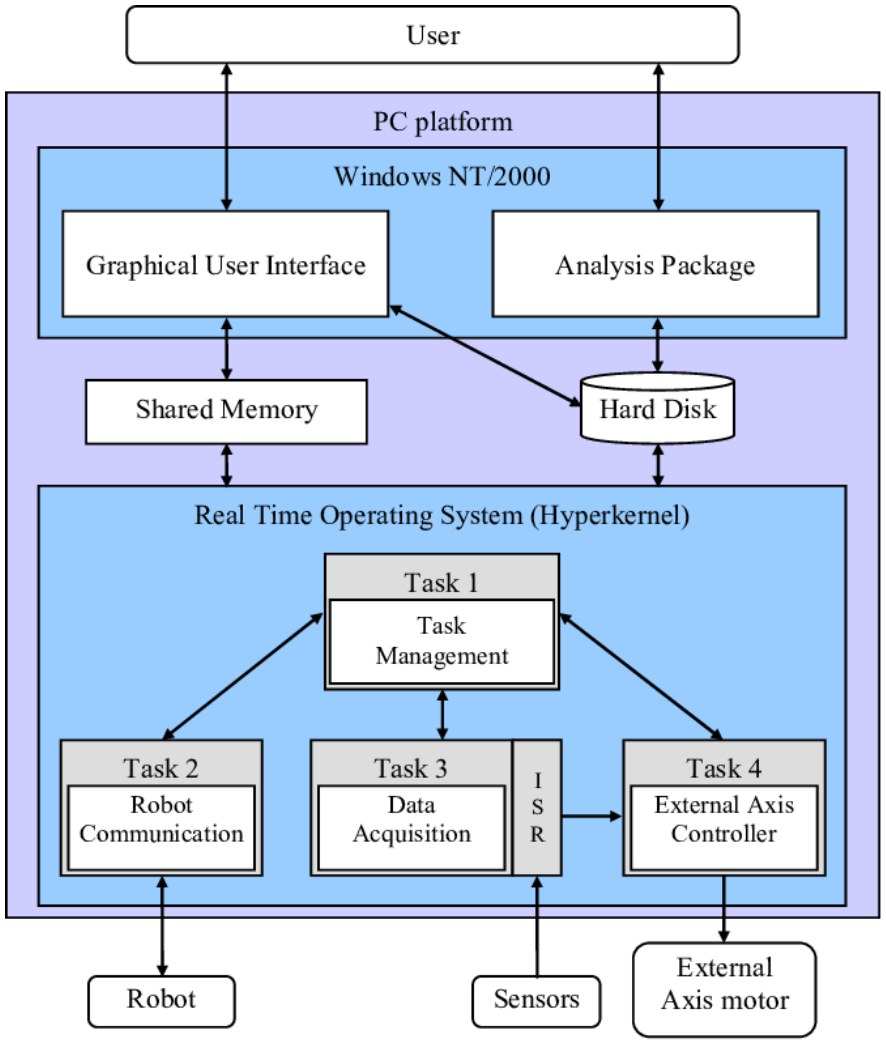
# INTRODUCTION TO SOFTWARE ARCHITECTURE

**ECE 811 – SOFTWARE ENGINEERING**

**Monday, July 28, 2025**

# WHAT IS SOFTWARE ARCHITECTURE? /01

- **Architecture** is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. » [ANSI/IEEE Std 1471-2000]
- **Software architecture** of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.



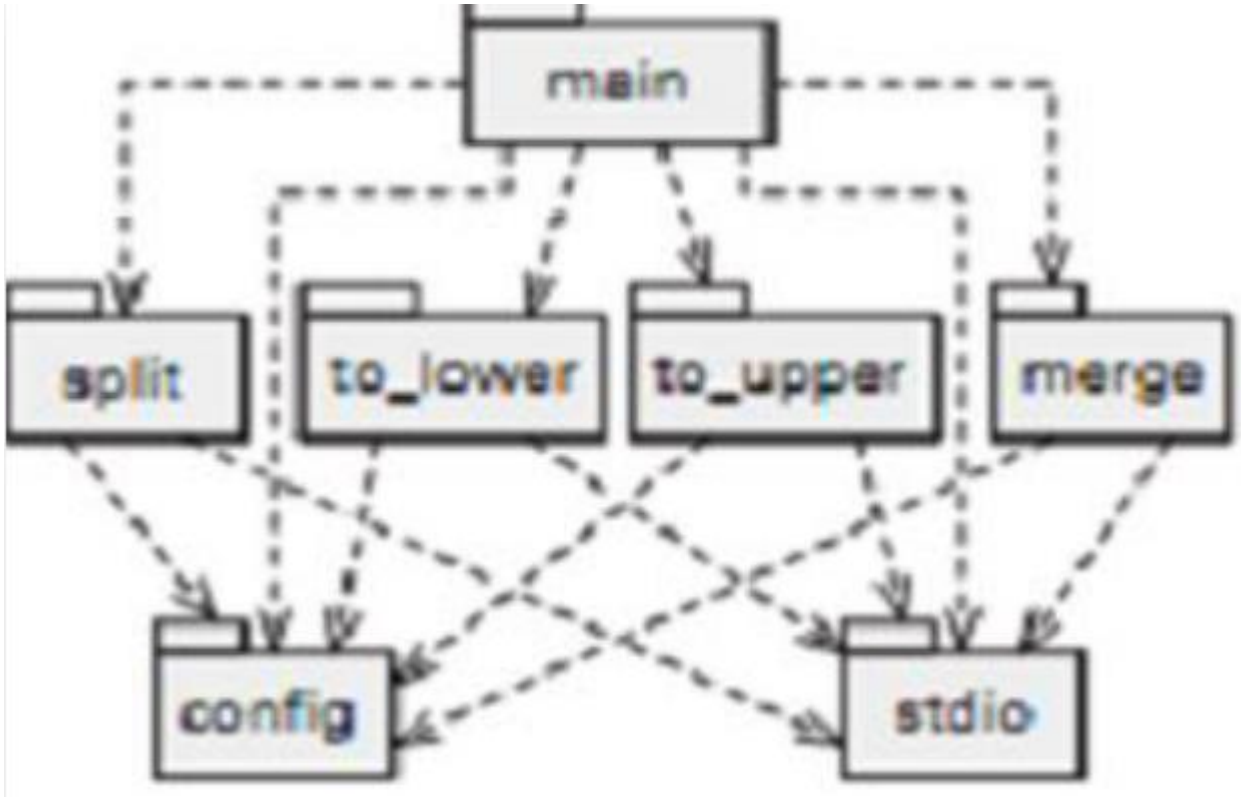

(a) Example of software architecture

# WHAT IS SOFTWARE ARCHITECTURE? /02

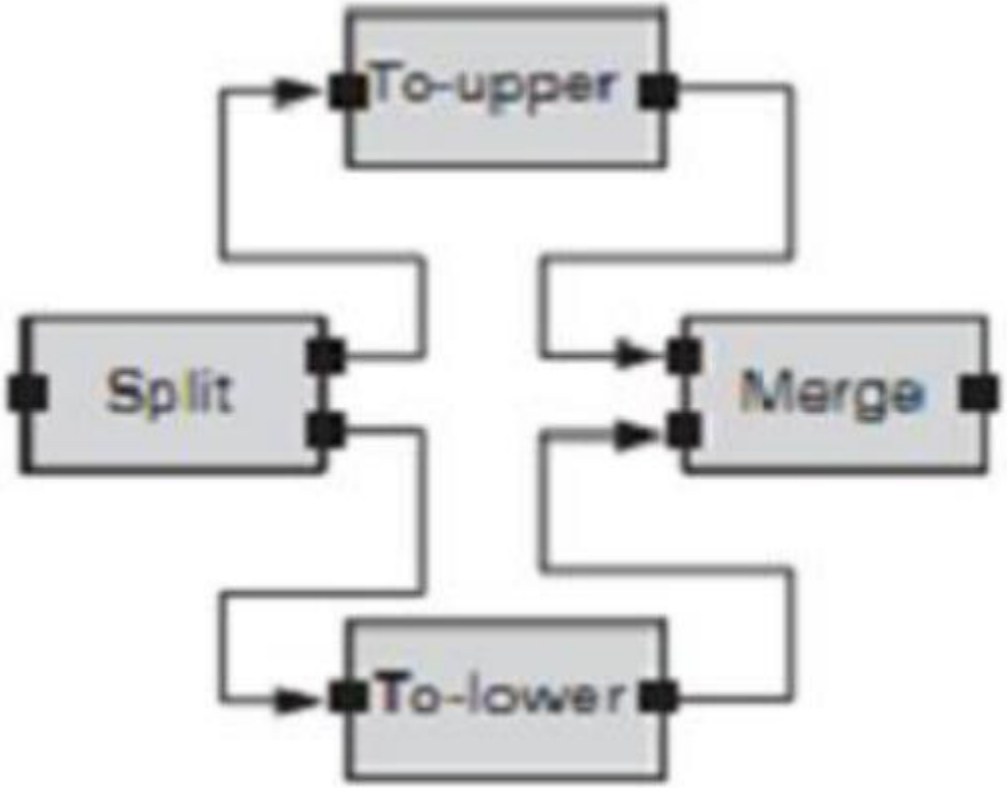
Mainly Architecture Defines Structure Decomposition of system into :

1. Software Elements
   - Elements are captured as abstractions
   - Correspond to high level system modules or components
2. Component interfaces
   - External visible properties of elements
   - Describe element features exposed to others
   - Typically represent services provided to other elements
3. Component responsibilities
   - What does a component precisely do?
4. Relationships of elements
   - How do elements interact with others?

# SOFTWARE ARCHITECTURE




(a) Software Modules




(a) Software Components and Connectors

# SOFTWARE STRUCTURES

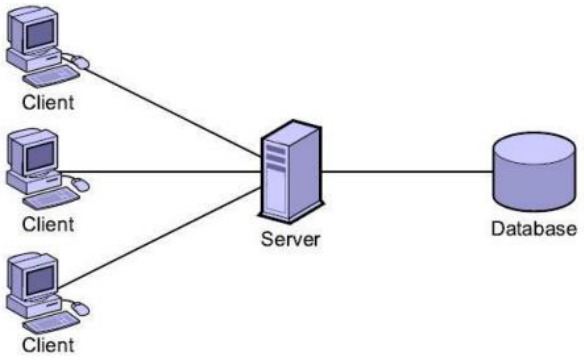
1. Large software systems require structures from multiple perspectives (views)
    - A single view is not sufficient to address all the requirements
2. Examples of views

    (a) **Context View** focuses on the system's functionality

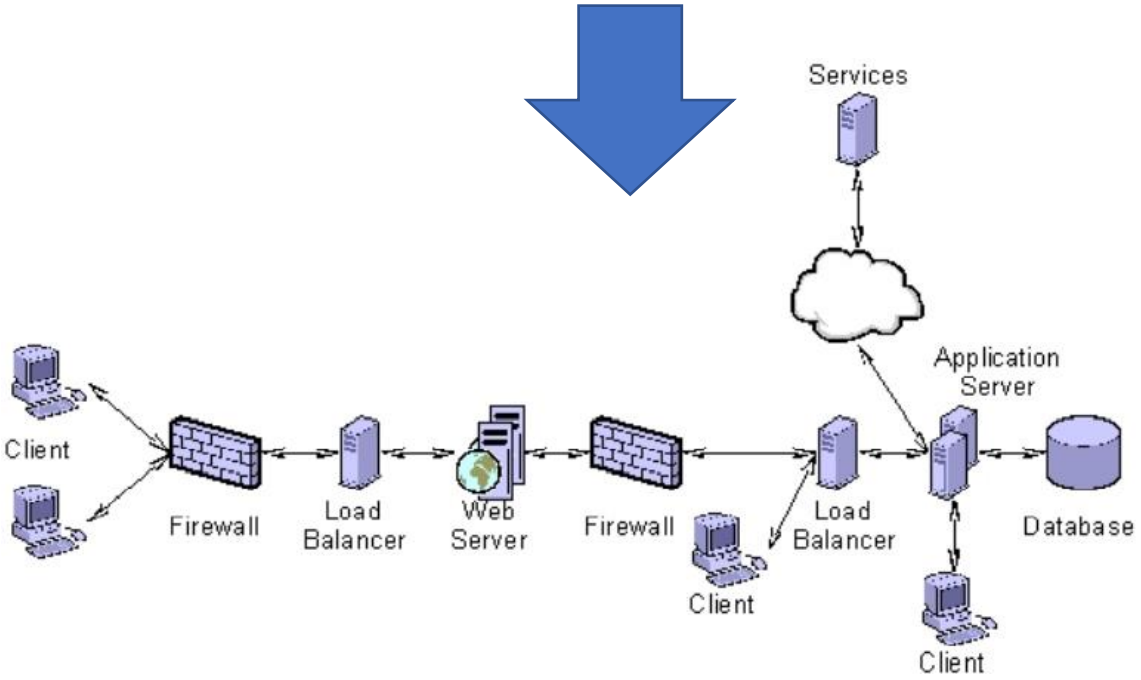
    (b) **Deployment View** reflects the physical deployment of software components to computing hardware

# WHY SOFTWARE ARCHITECTURE IS NECESSARY

- Recent changes in software engineering have increased the importance of architecture due to factors such as:
  a) Scale,
  b) Distribution
  c) Security




(a) Original client-server application



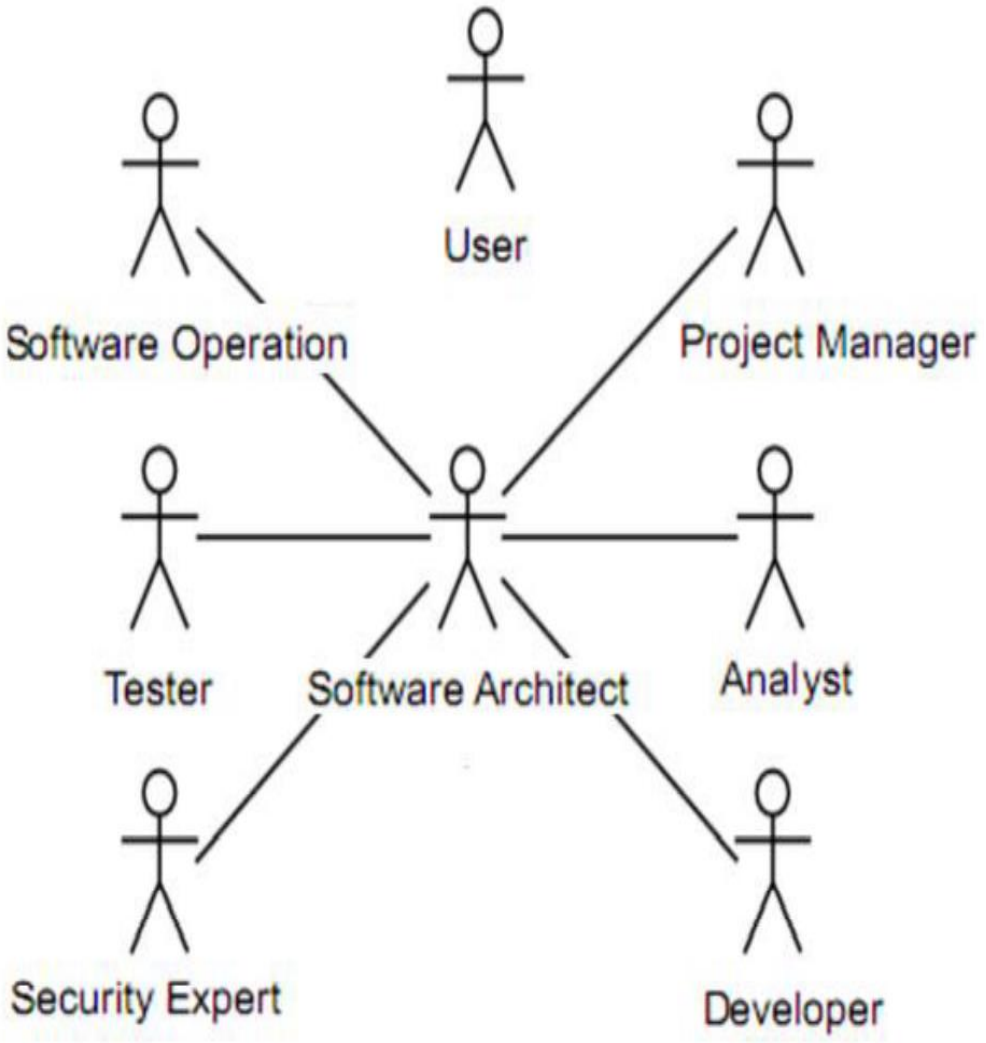

(a) contemporary client-server application

# FEATURES OF A GOOD SOFTWARE ARCHITECTURE

1. Takes into account:
    a) The functional requirements
    b) The non-functional requirements
    c) The entire system lifecycle
2. Helps us understand the system:
    a) Divides the system meaningfully
    b) Abstract complicated details
3. Provides the framework for:
    a) Realization
    b) Project Planning
    c) Project Organization
4. Integrate all the development artefacts
    a) Provide documentation

# ROLE OF A SOFTWARE ARCHITECT

1. **A software architect** is responsible for the high-level design and technical leadership of software solutions, ensuring they are robust, scalable, secure, and meet business requirements while aligning with technical standards and best practices.
2. **Software architect** act as a bridge between business needs and technical implementation, defining the overall structure and technical direction of a software system, much like an architect designs a building.



# ROLE OF SOFTWARE ARCHITECT

1. **Defining the architectural vision:** Making high-level design choices, selecting appropriate technologies and platforms, and establishing coding standards.
2. **Translating business requirements into technical solutions:** Understand business needs and translate them into a technical blueprint for the development team.
3. **Ensuring technical quality and non-functional requirements:** Ensure the software is scalable, maintainable, secure, and performs efficiently.
4. **Providing technical guidance and mentorship:** Guide the development team, share knowledge on best practices, and mentor less experienced developers.
5. **Collaborating with stakeholders:** Work closely with project managers, business analysts, and other stakeholders to ensure alignment with project goals and gather feedback.
6. **Evaluating trade-offs and making informed decisions:** Consider various options and potential implications of architectural decisions, communicating tradeoffs to business stakeholders.
7. **Supporting and reviewing implementation:** Oversee the implementation of the architecture, providing support and reviewing code to ensure it aligns with the design principles.