High Level Design

CS-370: Software Design and Development

Robert Bruce

What is high level design?

- High level design provides an overview of the project at an abstract level.
- High level design is the first step towards breaking a large project into smaller components.
- Rob's note: Breaking a project into smaller (manageable) components can lead to parallel (concurrent) construction spanning multiple teams. This technique saves time.

Source: pages 117- 118, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Monolithic architecture

- *Monolithic architecture*: The application is written as one large program which does everything.
- Advantages:
  - Since everything is built-in to one application, there is no reliance on external resources such as a network.
  - Useful for small applications requiring only one programmer.
- Disadvantages:
  - Requires complete understanding of how the system is to work cohesively.
  - May be difficult to refactor applications to add new features at a later time since everything is already coupled together.
  - Difficult to work on larger applications spanning multiple programmers and/or teams.

Source: page 124, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Client-Server architecture

- *Client-Server architecture*: The application is decoupled into separate client and server applications.
- Advantage:
  - Modular design allows multiple team members to develop applications concurrently (either working on client side or the server side).
- Disadvantage:
  - A network failure can cause the system to fail if the client applications cannot communicate with the server application or vise-versa.
- Rob's note: We will use the client-server architecture for our auction website project. The auction website creates dynamically-driven web-page content (from a web server) for the user via a web-browser (client). The web server will also rely on MySQL (server) to extract, then format content into HTML for the user (client).

Source: pages 125-127, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Component-Based architecture

- *Component-Based architecture*: The application is defined by a series of subroutines within the same executable program. Communication within the application is direct – internal - rather than through an external network layer.

Source: pages 127-128, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Service-Oriented architecture

- *Service-Oriented architecture*: The application is defined by a series of separate (modular) applications which communicate via network.

Source: pages 128-129, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Data-Centric architecture

- *Data-Centric architecture*: The application utilizes a database management system to store and retrieve data.

Source: page 130, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Event-Driven architecture

- *Event-Driven architecture*: The application is divided into a series of modular subroutines to process events depending on the type of event received (i.e. triggering of events impacts program execution). When there are no events to process, the program enters a wait-loop listening for events.

Source: page 130, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Rule-Based architecture

- *Rule-Based architecture*: The application functionality is defined by a series of rules to determine the flow-of-control.
- Rob's note: The rules which define Rule-Based architectures are most likely based on probabilistic weights to simulate decision-making in artificial intelligence based systems.

Source: page 130, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Distributed architecture

- *Distributed architecture*: The application is designed for maximum concurrency by running parallel computations across a series of compute nodes in a network-based computing environment.
- Rob's note: distributed architecture is specifically designed for high-performance computing where a task can be optimally subdivided into a series of parallel tasks which are executed concurrently.

Source: page 131, *Beginning Software Engineering* (2nd edition) by Rod Stephens