Ease of understanding: the structure should favor the understanding of the design by humans. While software systems are complex systems by nature, reducing this complexity is possible through abstraction and decomposition techniques:

- Clean separation and encapsulation of domain abstractions into classes;
- Extraction and separation of commonality between abstractions/classes;
- Minimizing unwanted coupling by properly distributing knowledge and responsibilities among subsystems and classes
- Consistent use of a vocabulary of proven solutions to recurring problems (design patterns).

Ease of modifying or extending: the structure should favor easily modifying or extending the design. This can be achieved through:

- Isolating unrelated concerns from one another;
- Isolating things that change from things that stay the same;
- Isolating things that change more often from those that change more rarely;
- Achieving a balance between specificity and generality in order to minimize the need for redesign in case of unexpected changes in the requirements or runtime environment.