Fred Brooks identifies four **essential difficulties** in software engineering that cannot be eliminated, only managed:

**Complexity**  
Software systems exhibit inherent complexity due to the exponential number of possible states and interactions. This complexity hinders comprehension and maintenance. Modular design and object-oriented principles, such as abstraction and encapsulation, address this by isolating concerns and reducing cognitive load.

**Conformity**  
Software must conform to external realities—business rules, legal constraints, hardware systems—without influencing them. This external dependency restricts design freedom and was a central issue in early software failures, where heterogeneous systems resisted unified modeling.

**Changeability**  
Software evolves with shifting requirements and technological contexts, leading to structural erosion over time. Design patterns and reuse strategies, such as the Factory and Strategy patterns, promote separation of concerns and flexible extension, making systems more resilient to change.

**Invisibility**  
Unlike physical systems, software has no inherent visual representation, complicating system understanding and team coordination. Interface-based design and clear responsibility allocation help expose structure, facilitating communication and reducing misinterpretation.