The Algorithmic Paradigm (AP)

- This is classical design automation

- It is characterized as the execution of a domain-specific algorithm which, given a set of requirements, generates a design satisfying them in a finite number of steps.

- It views computing as an algorithmic style and is essentially language-based.

- Significantly, the regularity of this paradigm does guarantee the avoidance of errors.
The Analysis-Synthesis-Evaluation Paradigm (ASE)

- **Methodology**: given a set of requirements, the software design process principally involves:
  - Analysis of requirements
  - One or more stages of synthesis principally driven by decomposition
  - Evaluation

- The ASE paradigm leads the designer to assume that his problem is well structured.

- Example: Waterfall Model of the Software Lifecycle. It derives from the desire to make software development "scientific"; essentially to define an engineering *Method*.

- The ASE paradigm, leading to the classical design approach of decomposition, is not widely applicable to software development: Decomposition essentially identifies possible choices of components considered independently.

- In general, the ASE paradigm is too rigid for all of our needs although useful for some aspects of the system.
The Formal Design Paradigm (FD)

- Due to the limitation of decomposition, abstraction with subsequent refinement was introduced.

- Abstraction provides us with a more flexible tool.

- It is in essence a refinement of the ASE paradigm.

- Mathematics provides a tool for abstraction and further as Hoare reasoned in 1986 at his inaugural lecture at Oxford:
  
  - Computers are mathematical machines (behavior is mathematically defined)
  - Programs are mathematical expressions (describe precisely what)
  - A programming language is a mathematical theory (a formal system for programming)
  - Programming is a mathematical activity.

- Thus software design becomes a mathematical proposition or theory that solves the problem as represented by the specification of the requirements.
• Hence, in the FD paradigm, the design process is an activity that exploits the traditional methods of mathematical reasoning.

• However, like the ASE paradigm it assumes that requirements are known and that essentially the problem is well-structured.
The Artificial Intelligence Paradigm (AI)

- It assumes that software design problems are not well-structured. Thus, rather than define a rigid set of requirements, the system should absorb domain-specific knowledge to an extent that it can provide a set of possible answers which satisfy the constraints implied by that knowledge.

- The more the problem solving system relies on domain specific knowledge the "stronger" is the problem solving method itself.

- The paradigm is explicitly founded on the concepts of search, knowledge and heuristics, in that the search is determined by knowledge of the problem domain and by a collation of general heuristics.

- The design process involves: a symbolic representation of the problem (the problem space) structured in terms of:
  - Initial problem state
  - Goal or desired problem state
  - All other states being reached or considered in attempting to reach the goal state from the initial state transitions from one state to another to
search for the solution through the problem space.

- "Expert Systems" are instances of such "strong" methods.

- Example: 8-puzzle

  $^\wedge$

  - Cost function: $^\wedge C = g(x) + h(x)$

  where

  $h(x) = \text{the number of misplaced tiles}$

  and $g(x) = \text{the number of moves so far}$

- Assumption: move one tile in any direction cost 1.
Initial State

\[\begin{array}{ccc}
1 & 2 & 3 \\
5 & 6 & 7 \\
\end{array}\]

Final State

\[\begin{array}{ccc}
1 & 2 & 3 \\
5 & 8 & 7 \\
\end{array}\]

Note: In case of tie, choose the leftmost node.
The Theory of Evolutionary Design Paradigm (TED)

- The adequacy of the design is solely determined according to whether it meets the requirements prevailing at that stage of the design process.

- In this paradigm, the software design is viewed as an evolutionary process: The requirement is to create a design framework that **accommodates change** (and failure) and utilizes one of the previous paradigms at the appropriate places. Basically it is a collection of **tentative hypotheses** such that:
  
  one can attempt to provide evidence in their favor to establish the "plausibility" of the design believe in the design’s plausibility may have to be revised

- TED is based on the view of design as an "empirical scientific" activity in contrast to the **Formal Paradigm** which views design as a "mathematical modeling" activity.

- TED views the act of software design as an **evolutionary process** and the design itself, at any stage of its development, as a tentative solution to the problem posed.

- In other words, according to the evolutionary model, a design at any stage of its development
(including the final stage) is: an evolutionary offspring of an earlier design form likely to evolve further in the future

- Essentially TED maintains that design in mixed paradigm systems can consist only of establishing constraints on the implementation. Collectively these constraints represent the design.

- The approach means, in practice, that designers are forced to resort to satisfactory rather than optimal designs, thus recognizing both the **evolutionary and error-prone nature** of software development.

- Thus the design is not only tentative at intermediate stages of its development but also when the designers (or their managers!) see fit to terminate or freeze the design.