# Application of AI in Software Engineering – A Review Challenging Conventional Wisdom

Team 10

**Team Members**

Rohini Ellakonda – 1002165355

Sharini Jayabal – 1002087946

Raghav Narayan Ramachandran - 1002140654

# INTRODUCTION

**Artificial Intelligence (AI)** is transforming software engineering by providing innovative solutions to long-standing challenges. This presentation will explore common issues software engineers face and how AI is reshaping the landscape.

The challenges in software engineering life cycle are:

- Estimation: Difficulty predicting time and costs
- Testing: Uncertainty in testing adequacy
- Requirements: Challenges in understanding user needs
- Maintenance: Reactive rather than proactive problem-solving
- Collaboration: Ineffective communication

  among team members

## APPLICATIONS OF AI IN SOFTWARE ENGINEERING

### Estimation

Provides accurate project timelines and cost predictions using historical data

### Testing

Automates test creation and bug detection, improving software quality

### Requirements Gathering

Uses NLP to analyze user feedback and prioritize features

### Maintenance

Predicts potential issues, allowing for proactive solutions and reducing downtime.

### Collaboration

Enhances communication through summarization tools, keeping teams aligned

## REQUIREMENTS GATHERING

**Definition:** Figuring out what users need from a software system

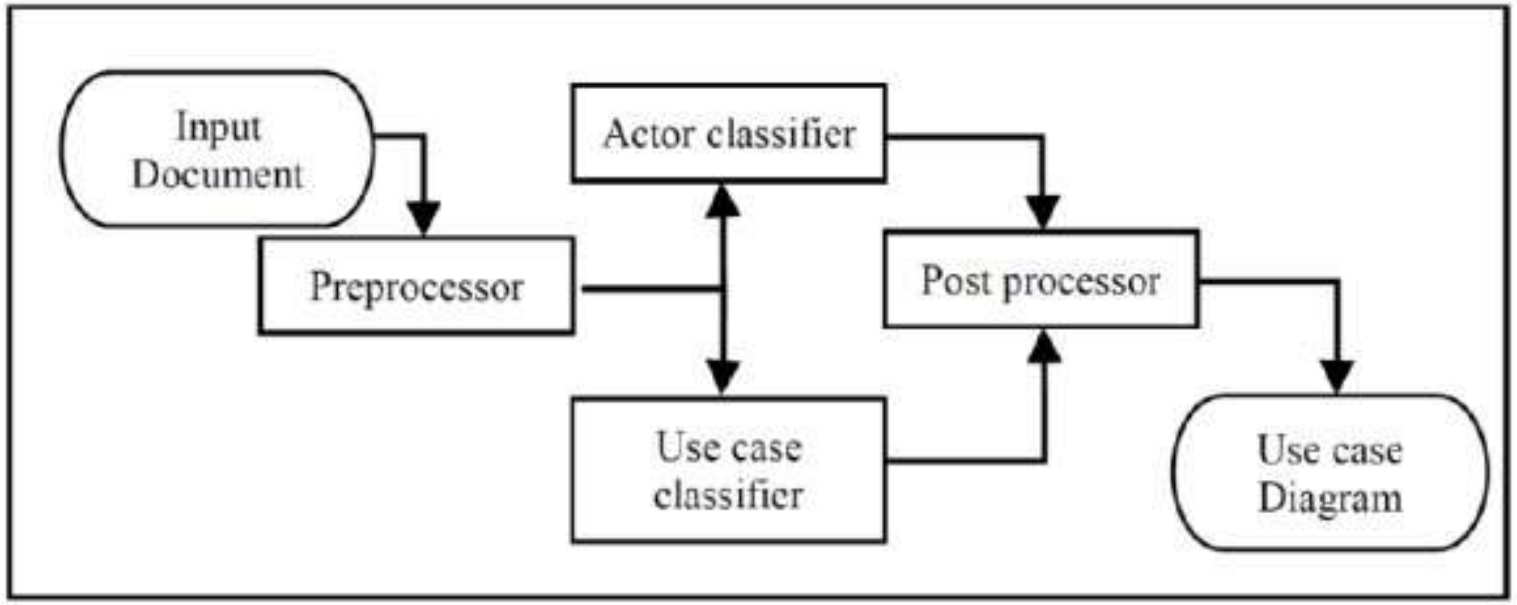
Clear requirements are essential to avoid unwanted products and high costs

The **key challenges** in requirements engineering are

- Unclear Specifications: Misunderstandings can lead to unwanted software
- Scalability: Managing large, messy data is time-consuming
- Costly Mistakes: Poor requirements can lead to expensive problems later

## SOLUTION - 1

### Semi-Automatic Approach (NLP)



## SOLUTION - 1

### Semi-Automatic Approach (NLP)

1. **Requirements Extraction:** NLP helps identify key details (e.g., actors, use cases) from unstructured text, reducing manual work.
2. **Spotting Confusion/Ambiguity:** It flags unclear or conflicting requirements for further clarification

**Potential Solution/Benefits:**

Time-saving: Quickly processes large volumes of text

Improved accuracy: Reduces human errors by automating key tasks.

**Challenges**: Natural language is often unclear, and NLP tools may struggle with context

## SOLUTION - 2

### NL-OOPS with LOLITA (Natural Language – Object-Oriented Production System) with (Large-scale Object-based Linguistic Interactor, Translator and Analyzer)

NL-OOPS project uses the LOLITA system to convert requirements documents into visual diagrams, analyzing text to extract key components even when written in everyday language.

1. **Text Analysis:** Identifies important entities and relationships within requirements

2. **Diagram Generation:** Automatically creates visual models to represent software components

**Potential Solution/Benefits:**

Enhanced Clarity: Makes complex requirements easier to understand through diagrams.

Time Efficiency: Saves time by automating the creation of visual models.

**Challenges**:

Context Understanding: May misinterpret nuanced language or context

Quality of Input Data: Relies on clear and complete requirements for accuracy

# SOFTWARE DESIGN

**Key Challenge**

The design phase, although it only takes up about 6% of the software budget, is critical because flaws in design can have significant impacts on subsequent phases like development and testing.

The traditional software design process is

- Time-consuming
- Error-prone
- Inefficient for modern, complex systems

## SOLUTION - 1

### Knowledge Based System (KBS)

KBS automate the generation of design models and improve decision-making during the design process.

**Example:**

KBS systems assist in database design by automatically generating blueprints based on predefined rules and requirements.

**Potential Solution/Benefits:**

KBS automates the generation of design models and decision-making processes, significantly reducing manual effort and minimizing human error.

**Challenges**:

Dependency on Predefined Rules : KBS can only operate within the boundaries of the predefined rules.

## SOFTWARE DEVELOPMENT AND IMPLEMENTATION

**Definition**

Software development is the process of designing, creating, testing, and maintaining software applications or systems

**Key Challenge**

- ✓ The evolutionary nature of Software Engineering (SE), with its long processes and stages, means that requirements can change by the time coding is finished.

- ✓ Traditional methods can be inefficient in managing these dynamic changes effectively, resulting in delays and errors in the development process.

## SOLUTION – 1:

### Ontology-Enabled Agent

This agent autonomously performs software development activities by interpreting requirements and synthesizing system logic. It uses an ontological database to make high-level reasoning decisions.

**Implementation:**

The Software Developer Agent (SDA) starts by reading the requirements specification and uses its internal knowledge base to design the software that realizes the system logic, generating software code as specified.

**Potential Solution/Benefits:**

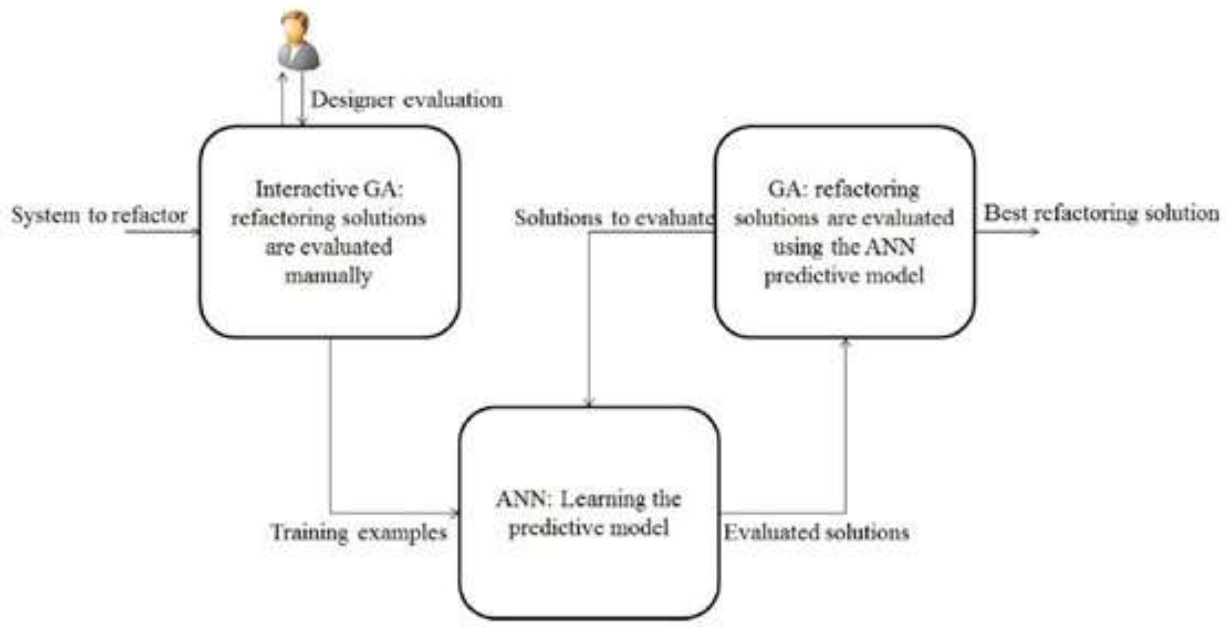
Adaptability to Changing Requirements

**Challenges**:

Extremely Complexity in Ontology Management

## SOLUTION 2

## Neural Network-Based Refactoring


Designer evaluation
System to refactor
Interactive GA: refactoring solutions are evaluated manually
Solutions to evaluate
GA: refactoring solutions are evaluated using the ANN predictive model
Best refactoring solution
Training examples
ANN: Learning the predictive model
Evaluated solutions

## SOLUTION 2

## Neural Network-Based Refactoring

### Interactive Genetic Algorithm (IGA)

The software engineers manually evaluate the suggested refactoring solutions generated by a Genetic Algorithm (GA) for a few iterations.

### Learning Genetic Algorithm (LGA)

The evaluated solutions from IGA are used to train an Artificial Neural Network (ANN), which then evaluates refactoring solutions for the remaining iterations, providing the best refactoring sequences to improve system quality.

**Challenges**:

High Training Requirements: Neural networks need a large amount of training data and iterations to become effective. The initial setup and training process can be time-consuming and resource-intensive

## SOFTWARE TESTING

Software testing is the process of executing a program to find **defects (bugs)** and ensure it meets business and technical requirements.

It involves:

- **Verification**: Ensuring the system is built **right**.
- **Validation**: Ensuring the **right system** is built.

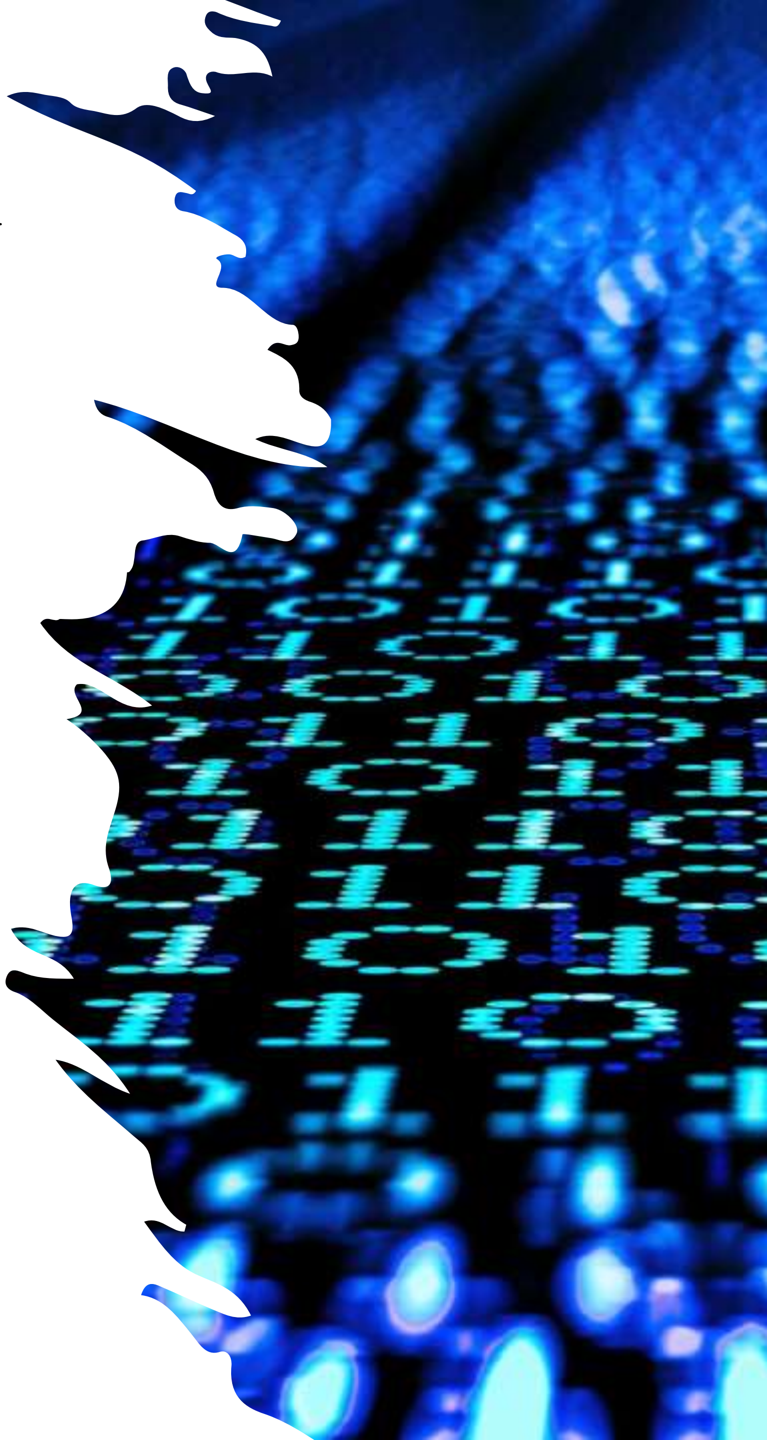
**Purpose**:

•More than bug detection: Includes **quality assurance** and **reliability estimation**.

•Evaluates **performance**, **security**, **usability**, and **robustness**.

**Challenges:**

•**Resource Intensive**: Takes up **40-50% of resources** and **50% of development time**



## SOLUTION 1

## KITSS, Knowledge-Based Interactive Test Script

NLP with a hybrid domain model and analyzers to translate incomplete test cases into scripts.

**Purpose:** Automate test code generation for Private Branch Exchange (PBX) switches.

**Features:**

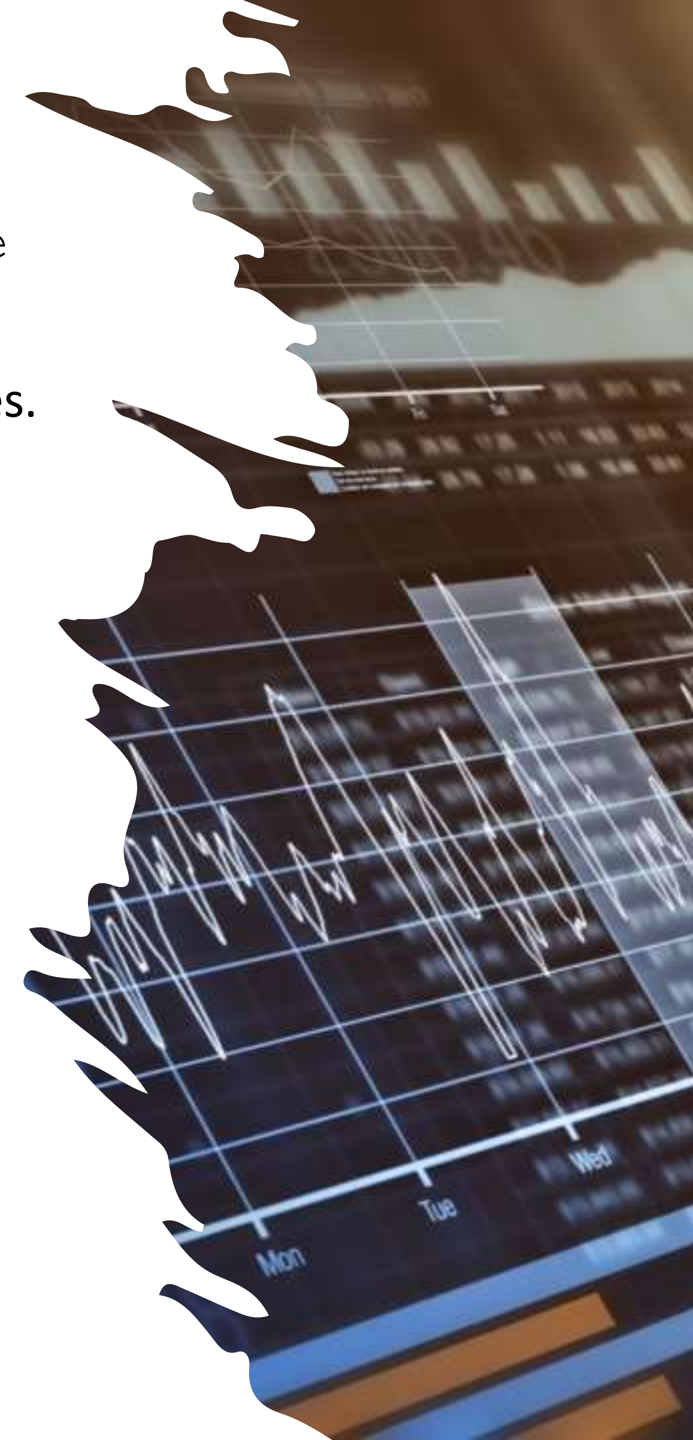
- ➢ Focused on functional testing, not internal structures.
- ➢ Converted English test cases into automated test scripts.

**Potential Solution / benefits:**

- ▪ Increased automation
- ▪ Reduced maintenance

**Challenges:**

- ➢ Only ~5% of test cases were written using automation languages.
- ➢ Conversion issues between test cases and test scripts



## SOLUTION 1

## KITSS, Knowledge-Based Interactive Test Script

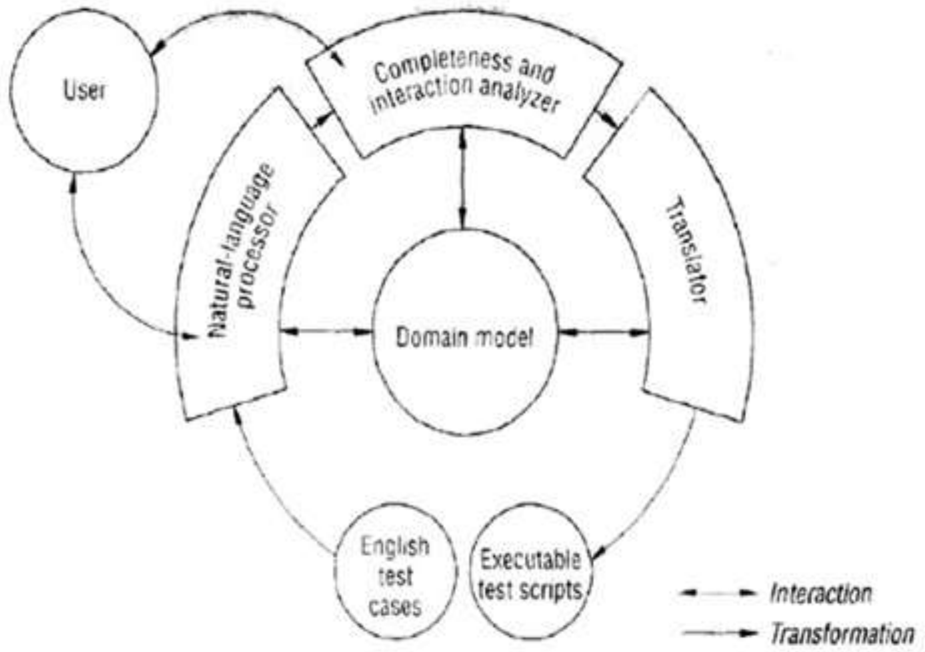


*Figure 11: The KITSS Architecture [152]*

## SOFTWARE RELEASE AND MAINTENANCE

Software Release and Maintenance is the modification of a product post-delivery to correct faults, improve performance, or adapt to new environments.

**Key Process:**

Begins with understanding user feedback to identify required changes and improvements based on customer needs.

**Challenges:**

**Differentiating reviews:** Feedback can include biased or spam reviews, making it difficult to identify genuine issues.

## SOLUTION 1

### Text Mining for Review Classification

**Applied shingling technique:**

Grouped neighboring words and compared across documents to detect duplicate or near-duplicate reviews.

A review classified against 5 classifiers and using majority voting to determine it’s type.

**Potential Solution / Benefits**: Provides structured support in planning and decision-making for software maintenance.

**Limitation**: Requires manual updates and can be cumbersome in complex systems due to rule dependencies.

## SOLUTION 2

### Expert Systems for Software Maintainability

**Purpose:**
Introduced Expert Systems (also known as Knowledge-Based Systems) to help engineers plan software maintainability.

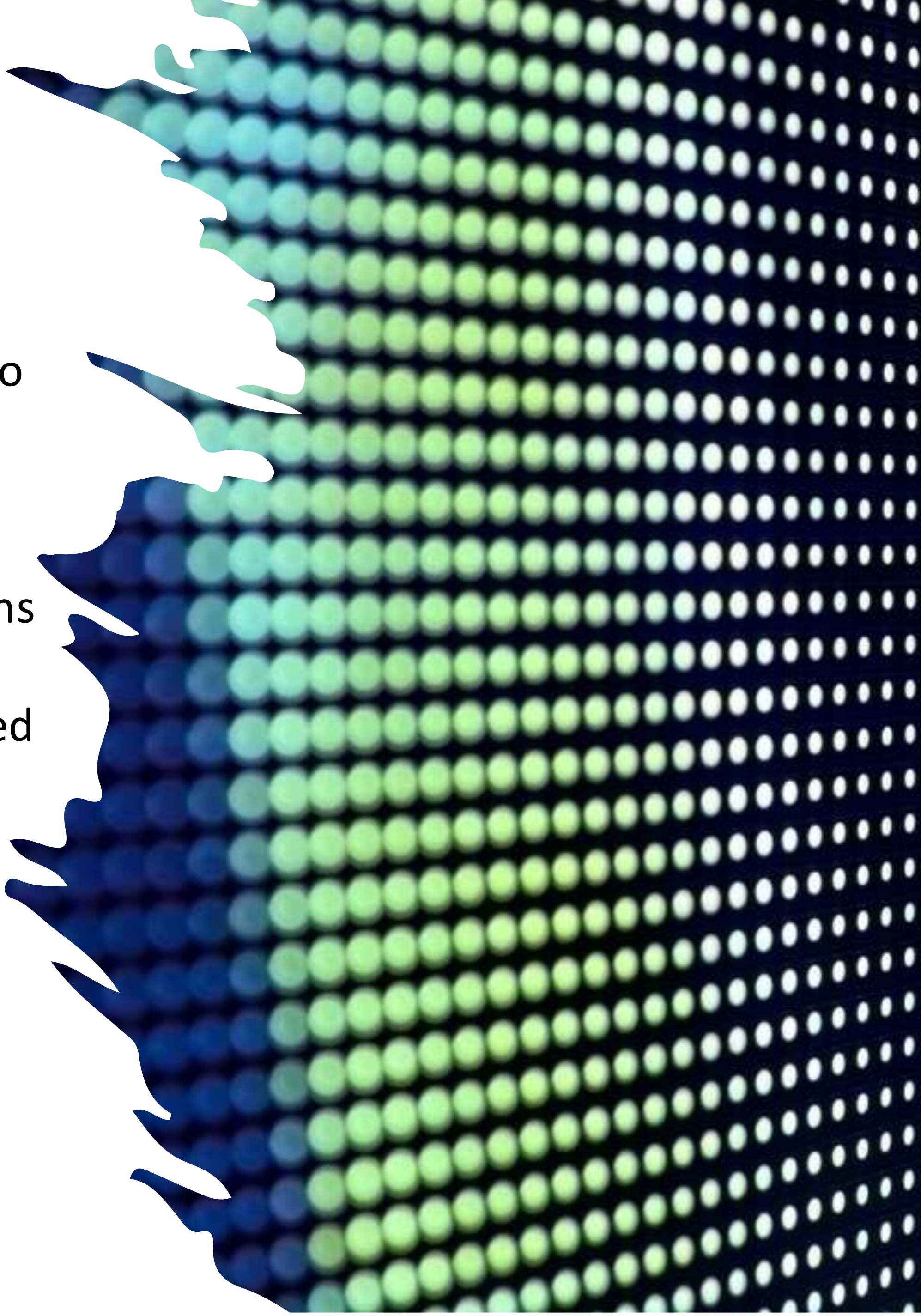
**How it Works:**

- Rule-based decision-making: The system asks a series of questions and provides conclusions based on predefined rules.
- Acts as a support tool for engineers, providing structured recommendations.

**Potential Solution / Benefits:**

- Systematic method for capturing human expertise.
- Easily adaptable by adding or modifying rules as needed, improving decision support.

**Limitation**: No learning ability and Complex rule management:



## QUESTIONS?