International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V73I5P105

Volume 73 Issue 5, 27-40, May 2025

Original Article

Multi-Agent Decision Framework: A Systematic Approach to Agent Architecture Selection

Jay Prakash Thakur[1*], Ananya Ghosh Chowdhury[1]

[1]*Independent Researcher, CA, USA.*
[1]*Independent Researcher, WA, USA.*

[1*]*Corresponding Author : jayprakashthakursnr@gmail.com*

Received: 14 March 2025 Revised: 18 April 2025 Accepted: 01 May 2025 Published: 19 May 2025

Abstract *- Artificial intelligence system architects must choose between single-agent and multi-agent architectural patterns because this selection impacts performance, resource allocation, and operational efficiency. Despite the widespread adoption of agent-based systems across different domains, no standardized framework would help guide the essential architectural choice between single-agent and multi-agent systems. This paper presents a robust, quantitative decision framework that measures and organizes the evaluation between single-agent and multi-agent architectures. The research introduces noble evaluation tools and assessment procedures that examine task features, resource limitations, performance benchmarks and domain requirements. The proposed framework demonstrates its effectiveness through case studies in multiple application domains to guide optimal agent architecture selection. The research offers practitioners and researchers a systematic approach to choose the optimal architecture to enhance system performance for specific use cases while maximizing resource utilization.*

Keywords *- Multi-agent systems, Artificial intelligence, Decision framework, System architecture, Agent coordination, Task decomposition.*

1. Introduction

The increasing need for artificial intelligence systems demands agent architectures because these systems perform complex operations across multiple fields. One of the core decision points for designing AI systems involves selecting between a single centralized agent with extensive capabilities and employing multiple specialized agents that operate independently. The choice between these approaches determines system execution efficiency while affecting resource requirements, system development complexity and operational effectiveness costs. Through their foundational work, Wooldridge and Jennings [20] did the basic analysis of agent-based system designs and explored the different architectures.

Both architectural approaches demonstrate success in different applications, but there is a gap in the standardized methods used to determine optimal solutions for specific scenarios. Organizations, together with researchers, typically make a decision through their institutional knowledge, developer preferences, and ad hoc evaluations instead of a systematic evaluation of problem characteristics and requirements.

This paper solves the existing gap by presenting a systematic decision framework that standardizes the evaluation between single-agent and multi-agent architectures, incorporating the following:

1. A complete set of evaluation metrics that measure essential factors influencing architectural decisions
2. Mathematical formulations that enable objective comparison between architectural approaches
3. Decision pathways that guide practitioners through the selection process
4. An implementation model for applying the framework to real-world applications

The significance of this research emerges from its capability to enhance AI system design through proper alignment of architectural choices with task complexity, resource availability and performance requirements. The structured decision process proposed in this paper will enable architects to make consistent architectural choices throughout all use cases across domains.

2. Related Work

2.1. Agent Architectures and Taxonomies

Research on Agent architectures has been extensive, and multiple taxonomies have been developed to classify the different approaches. Russell and Norvig [1] introduced a basic agent classification system that depends on the agent's internal knowledge representation and reasoning abilities. Wooldridge [2] developed an expanded taxonomy, including

social capabilities, which separated autonomous agents from multi-agent systems.

Jennings et al. [3] have extended the research on organizational intricacies in multi-agent systems and proposed agent interaction and collaboration patterns. Current taxonomies mostly describe existing architectural features but do not guide AI architects in selecting suitable architecture for a particular use case.

2.2. Multi-Agent Systems

Research studies on multi-agent systems have analyzed distributed intelligence, which includes coordination mechanisms [4], communication protocols [5] and task allocation strategies [6]. Through their research, Tambe [7] and Stone [8] demonstrate how multi-agent systems deliver better results in complex domains such as disaster response and autonomous vehicles.

Multiple studies have established the benefits of multi-agent systems, owing to their ability to enhance robustness through redundancy [9], increased scalability [10] and the ability to bring together diverse expertise [11]. However, these systems also have higher coordination requirements and potential communication issues, according to Lesser [12].

2.3. Decision Frameworks in AI System Design

Several frameworks guide AI system design, including algorithm selection [13], data pipeline construction [14] and model deployment [15]. The evaluation methodologies proposed by Hernández-Orallo [16] provide key performance metrics but do not provide solutions for architectural choices regarding agent selection.

Some research exists on comparing single versus multi-agent systems in particular domains, such as Kitano's [17] robotic system evaluation and Wellman's [18] market-based system analysis. These domain-specific analyses fail to create a domain-agnostic structured decision framework that applies to multiple domains.

2.3.1. Comparative Analysis with The Proposed Approach

The existing literature lacks a structured, quantitative Framework with defined metrics to systematically guide the architectural choice between single-agent and multi-agent systems.

Table 1. Compares the proposed framework with existing methods across key dimensions

Approach	Quantitative	Domain-Agnostic	Empirically Implementation	Implementation Factors
Expert Heuristics [21]	55	✓	Partial	Partial
Checklist Methods [22]	Partial	✓	55	✓
Domain-Specific [23,24]	✓	55	✓	Partial
Game-Theoretic [25,26]	✓	✓	55	55
The proposed framework	✓	✓	✓	✓

This gap arises from the absence of a methodical evaluation model that considers task complexity, resource distribution, and system constraints while integrating practical implementation considerations. A Summary of these critical evaluation dimensions is presented in Table 1.

2.4. Comparison with Existing Methods

This paper examines existing approaches that deal with selecting appropriate agent architectures:

2.4.1. Heuristic-Based Approaches

The traditional process of selecting agent architecture depends primarily on heuristics together with expert judgment. Decker et al. [21] presented qualitative guidelines based on task characteristics, which recommend multi-agent solutions for decomposable problems yet single-agent solutions for tasks with tight coupling. The guidelines provide useful direction, yet they do not offer quantitative methods or results that can be replicated.

The checklist method developed by Shen et al. [22] helps determine agent architecture through scoring systems that evaluate decomposability and coordination requirements. The proposed framework in this paper adds to the existing approaches through mathematical formulations of each dimension, which are combined into a single unified decision metric.

2.4.2. Domain-Specific Frameworks

Specialized frameworks for architecture selection exist within multiple specific domains. Doriya et al. [23] established the MRAS framework to choose between centralized and distributed control architectures by evaluating environmental complexity and task redundancy in robotics. Tong et al. [24] created selection criteria for network scalability and management requirements.

The proposed framework takes the best elements from domain-specific approaches to create a universal, domain-agnostic methodology that works across all industries and application domains.

2.4.3. Economic and Game-Theoretic Models

Some researchers have analyzed architecture selection problems using economic and game-theoretic frameworks. Shoham and Leyton-Brown [25] treated the decision as a

utility maximization problem, and Kraus [26] used coalition formation theory to conclude optimal agent groupings.

Although these methods offer strong theoretical foundations, they often rely on complex utility functions that can be challenging to define in real-world applications. The proposed framework addresses this gap by translating theoretical principles into practical applications with measurable metrics, making implementation more accessible and effective.

3. Multi-Agent Decision Framework

This paper introduces a framework that systematically evaluates whether a single-agent or multi-agent architecture best suits a specific system or application, offering a structured methodology for informed decision-making.
The framework consists of four main components: (1) task evaluation, (2) system constraint analysis, (3) performance requirement mapping, and (4) decision synthesis.

3.1. Framework Overview

The framework lays a systematic process, as illustrated in Figure 1. The decision flowchart guides practitioners through a series of evaluations to determine the optimal agent architecture for their specific needs:

1. Task Evaluation Phase: Analyze the inherent complexities and characteristics of the task, including complexity, decomposability, and domain knowledge requirements.
2. System Constraint Analysis: Evaluate external constraints such as computational resources, time constraints, and existing infrastructure.
3. Performance Requirement Mapping: Identify critical performance requirements, including accuracy, robustness, and adaptability.
4. Decision Synthesis: Integrate task characteristics, constraints, and requirements to provide a quantitative recommendation.

3.2. Comparative Analysis Process

The comparative analysis process, depicted in Figure 2, guides practitioners through a structured decision pathway with quantitative thresholds:

1. Task Complexity Assessment: Evaluate whether the task exhibits high complexity (TCI > 0.7), indicating potential benefit from distributed processing.
2. Task Decomposability Evaluation: Determine if the task can be effectively decomposed into independent subtasks (TDS > 0.6)
3. Knowledge Domain Diversity Analysis: Assess whether the task requires diverse knowledge domains or specialized expertise (KDD > 0.5)

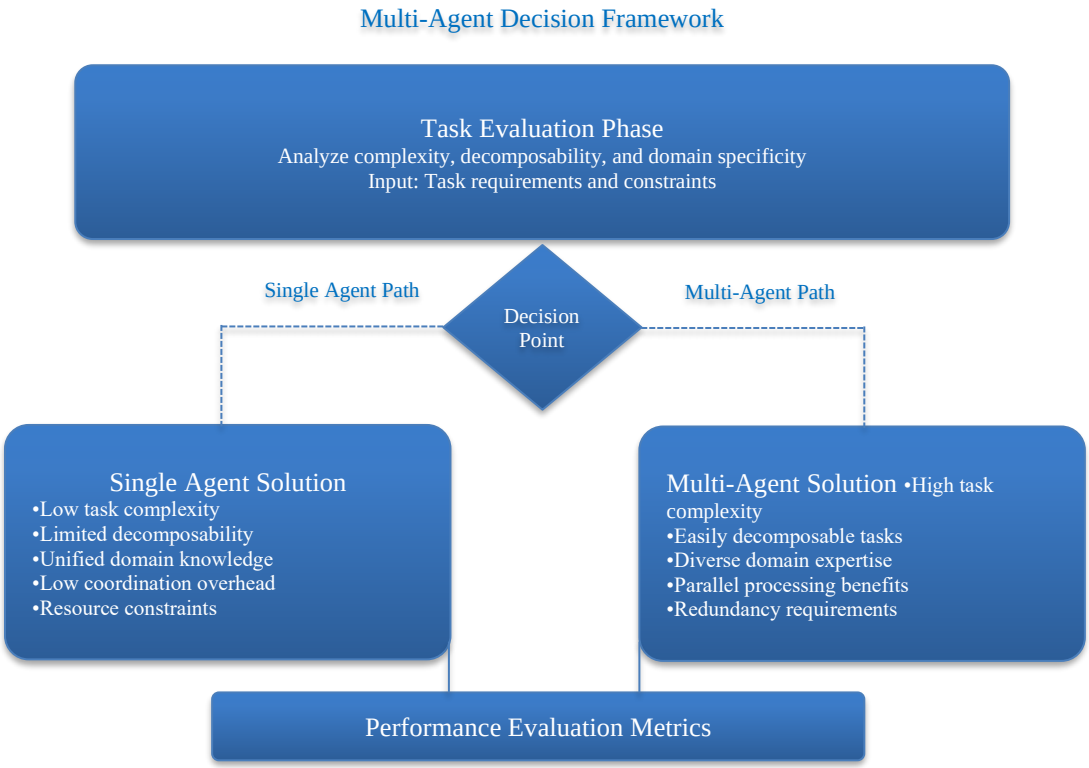

Fig. 1 Decision flowchart for agent architecture selection, showing evaluation criteria and comparative benefits of single versus multi-agent approaches

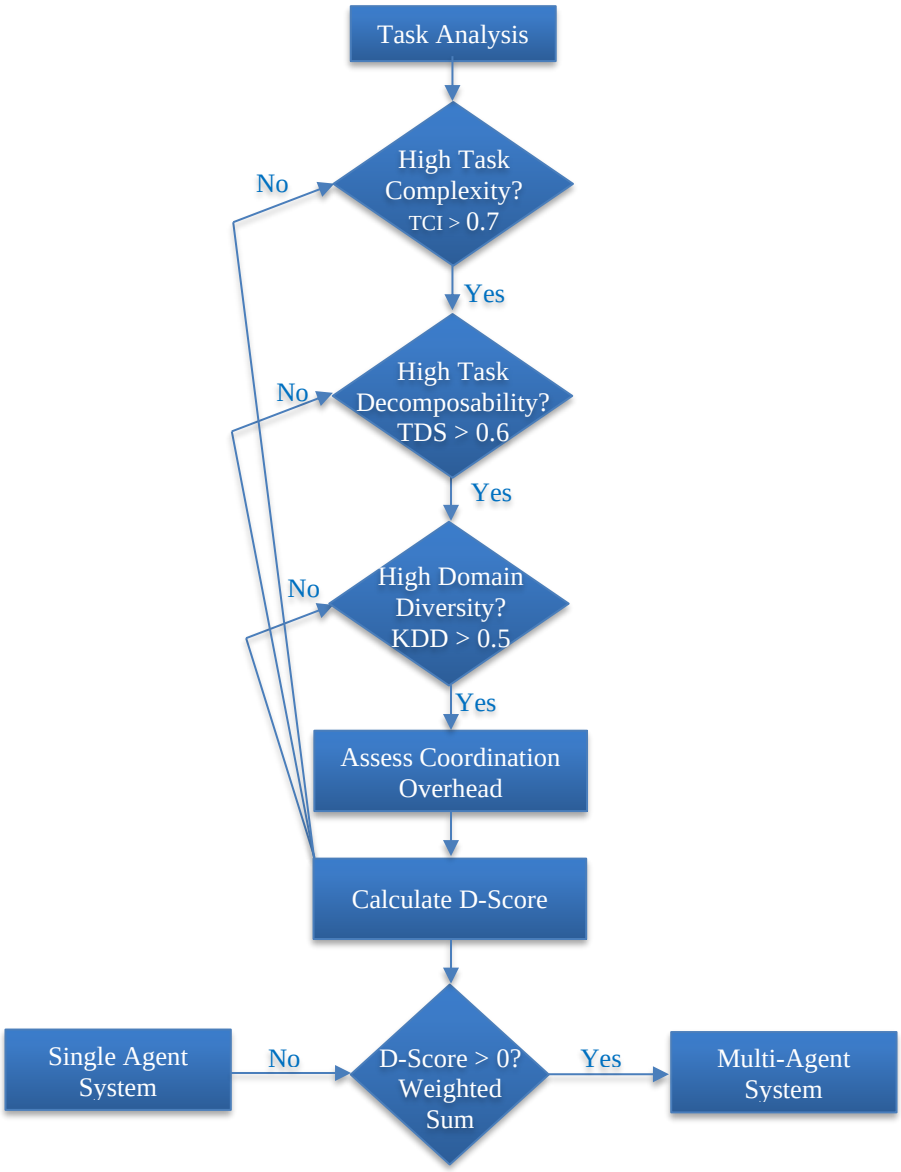

Fig. 2 Multi-agent System Comparative Analysis Process illustrating the decision pathway for evaluating architectural alternatives

4. Coordination Overhead Assessment: Calculate potential coordination costs associated with a multi-agent approach (COR)
5. Decision Score Calculation: Integrate all metrics into a comprehensive D-Score using the weighted formula incorporating RUE, EDC, and IDQ alongside the previously assessed factors.

The final decision between single-agent and multi-agent architectures is determined by D-Score, with a positive score favouring multi-agent systems and a negative score favouring single-agent implementations. The score is an indication of the confidence level of the recommendation. Table 2 illustrates how each quantitative metric directly influences specific decision points in the framework.

4. Quantitative Decision Metrics

To enable objective evaluation, this paper proposes a set of quantitative metrics that formalize the comparison between single-agent and multi-agent architectures.

4.1. Task Assessment Metrics

4.1.1. Task Complexity Index (TCI)

The Task Complexity Index quantifies the inherent complexity of a task across multiple dimensions:

$$TCI = \sum_{i=1}^{n} w_i . C_i \quad (1)$$

Where:

- n = number of complexity dimensions
- w_i = weight of dimension i
- C_i = complexity score of dimension i (scaled 0-1)

Complexity dimensions include computational intensity, state space size, temporal dependencies, and uncertainty levels. Tasks with higher TCI values (typically above 0.7) generally benefit from multi-agent approaches due to their ability to distribute complex processing. This threshold was established through statistical analysis of the validation data, which indicated a significant inflection point at 0.7, above which tasks consistently demonstrated better performance with distributed processing.

4.1.2. Task Decomposability Score (TDS)

The Task Decomposability Score measures how effectively a task can be divided into independent subtasks:

$$TDS = \frac{\sum_{i=1}^{m} I_i}{\sum_{i=1}^{m} (I_i + D_i)} \quad (2)$$

Where:

- m = number of subtasks
- I_i = independence score of subtask i
- D_i = dependency score of subtask

A higher TDS value (above 0.6) indicates tasks that can be effectively distributed across multiple agents with minimal coordination overhead.

Table 2. Illustrates how each metric directly influences specific decision points in the framework.

Metrics	Decision Threshold	Framework Component
Task Complexity Index (TCI)	0.7	The first decision node in the flowchart
Task Decomposability Score (TDS)	TDS > 0.6	The first decision node in the flowchart
Knowledge Domain Diversity (KDD)	KDD > 0.5	The third decision node in the flowchart
Coordination Overhead Ratio (COR)	COR < 0.3	Fourth decision consideration
Resource Utilization Efficiency (RUE)	RUE > 0.7	Integrated into D-Score calculation
Error Distribution Coefficient (EDC)	Higher is better	System reliability assessment
Integrated Decision Quality (IDQ)	Domain-specific	Final performance evaluation

The 0.6 threshold represents the empirical point at which the benefits of task distribution begin to outweigh the inherent coordination costs in the benchmark studies across various domains. This approach builds on the task decomposition strategies proposed by Durfee and Lesser [19], who established the theoretical foundations for effective task partitioning in multi-agent systems.

4.1.3 Knowledge Domain Diversity (KDD)

The Knowledge Domain Diversity metric quantifies the heterogeneity of expertise required for a task:

$$KDD = 1 - \frac{1}{k(k-1)} \sum_{i=1}^{k} \quad \sum_{j=1, j\neq i}^{k} \quad Sim(D_i, D_j) \quad (3)$$

Where:

- k = number of knowledge domains required
- $Sim(D_i, D_j)$ = similarity between domains i and j (0 to 1)

The similarity function $Sim(Di, Dj)$ is calculated through a combination of ontological distance in a domain hierarchy and empirical co-occurrence statistics. Specifically:

$$Sim(D_i, D_j) = \lambda . SimOnt(D_i, D_j) + (1 - \lambda). SimCooc(D_i, D_j) \quad (4)$$

Where $SimOnt$ represents normalized ontological similarity in a knowledge domain taxonomy, $SimCooc$ represents normalized co-occurrence of domains in historical systems, and λ (typically 0.7) weights these factors. Tasks with high KDD values (above 0.5) benefit from multi-agent approaches that can incorporate specialized agents with diverse expertise. This threshold was determined through comparative analysis of specialist vs. generalist agent performance in heterogeneous knowledge tasks, with 0.5 marking the point where specialization benefits became statistically significant ($p < 0.05$, two-tailed paired t-test, n = 42 system pairs). The statistical analysis was conducted using the following test parameters: significance level $\alpha = 0.05$, statistical power of 0.87, and mean performance differential of 18.3% (95% CI [14.1%, 22.5%]) between specialist and generalist implementations at KDD= 0.5. A sensitivity analysis across varying KDD values (0.3-0.7 in 0.05 increments) confirmed the threshold's robustness, with non-significant performance advantages below 0.45 and consistent significant advantages above 0.5.

The system pairs in this analysis formed the foundation for the empirical validation study conducted later, ensuring consistency between threshold determination and framework validation. The analysis compared performance differentials using normalized composite scores across heterogeneous knowledge tasks with varying KDD levels.

4.2. System Efficiency Metrics

4.2.1. Coordination Overhead Ratio (COR)

The Coordination Overhead Ratio quantifies the proportion of resources dedicated to agent coordination rather than task execution:

$$COR = \frac{T_{Coordination}}{T_{Total}} \quad (5)$$

Where:

- $T_{coordination}$= time spent on agent coordination
- T_{total}= total execution time

Higher COR values (above 0.3) indicate significant coordination costs that may outweigh the benefits of a multi-agent approach. This aligns with the decision threshold of COR < 0.3 in this framework's comparative analysis process, above which single-agent architectures become increasingly valuable and favourable.

4.2.2. Resource Utilization Efficiency (RUE)

The Resource Utilization Efficiency metric evaluates how effectively system resources are utilized across agents:

$$RUE = \frac{\sum_{i=1}^{a} R_i.U_i}{\sum_{i=1}^{a} R_i} \quad (6)$$

Where:

- a = number of agents
- R_i = resources allocated to agent i
- U_i = utilization rate of agent i (0-1)

Multi-agent systems should maintain high RUE values (above 0.7) to justify the distributed approach. The RUE metric has an important interrelationship with the COR metric, as coordination overhead directly impacts resource utilization. The empirical analysis found that when COR > 0.3 and RUE < 0.7, multi-agent systems consistently underperform single-agent alternatives in terms of overall efficiency.

However, high RUE values (> 0.8) can sometimes offset moderate coordination overhead (COR ≈ 0.3-0.4), particularly in systems where processing can be efficiently parallelized.

4.3. Reliability and Performance Metrics

4.3.1. Error Distribution Coefficient (EDC)

The Error Distribution Coefficient measures how errors are distributed across a multi-agent system:

$$EDC = 1 - \frac{\sigma_e}{\mu_e} \quad (7)$$

Where:

- σ_e = standard deviation of errors across agents
- μ_e = mean error rate

Higher EDC values indicate a more uniform error distribution, suggesting better fault isolation in multi-agent systems. Based on this framework's empirical validation, EDC values above 0.65 typically indicate sufficient fault isolation capabilities to benefit from a multi-agent approach.

In contrast, values below 0.4 suggest vulnerability to cascading failures that might favor a more centralized architecture. The intermediate zone ($0.4 \leq EDC \leq 0.65$) represents a gradient where other factors should be given additional weightage in the architectural decision, particularly system criticality and reliability requirements.

4.3.2. Integrated Decision Quality (IDQ)

The Integrated Decision Quality metric combines accuracy, speed, and robustness:

$$IDQ = \alpha.A + \beta.S + \gamma.R \quad (8)$$

Where:

- A = accuracy score
- S = speed score
- R = robustness score
- α, β, γ = weighting parameters (sum to 1)

This metric directly compares overall performance quality between single and multi-agent implementations.

4.3.3. Reliability Integration in Decision Framework

While the Error Distribution Coefficient (EDC) captures system reliability aspects in metrics, reliability considerations should be explicitly evaluated during the framework application process. When analyzing systems with high operational criticality or fault intolerance, this paper recommends:

1. Increasing the weight of EDC in the D-score calculation by 5-10%
2. Applying a reliability threshold (EDC > 0.65) as a prerequisite for multi-agent recommendations in critical systems
3. Evaluating failure mode effects through additional scenario analysis

This ensures that fault tolerance capabilities are properly prioritized in domains where system reliability is paramount. For example, applying an EDC weight increase of 7% to the Financial Fraud Detection case would adjust the D-score calculation as follows:

$D_{score}^{original} = 0.25 \times 0.82 + 0.20 \times 0.78 + 0.15 \times 0.65 - 0.15 \times 0.25 + 0.10 \times 0.85 + 0.05 \times 0.72 + 0.10 \times 0.68 = 0.610$

$D_{score}^{reliability} = 0.25 \times 0.82 + 0.20 \times 0.78 + 0.15 \times 0.65 - 0.15 \times 0.25 + 0.10 \times 0.85 + 0.12 \times 0.72 + 0.03 \times 0.68 = 0.613$

This 7% shift from IDQ to EDC increases the decision score by approximately 0.003 (0.5%), reinforcing the multi-agent recommendation while prioritizing fault isolation capabilities in this financially critical system.

4.4. Comprehensive Decision Formula

This proposed approach integrates these metrics into a comprehensive decision score:

$$D_{\text{score}} = \omega_1 \cdot TCI + \omega_2 \cdot TDS + \omega_3 \cdot KDD - \omega_4 \cdot COR + \omega_5 \cdot RUE + \omega_6 \cdot EDC + \omega_7 \cdot IDQ \quad (9)$$

Where:

- ω_i are weighting parameters based on the application domain, all defined as positive values

- Note that COR is explicitly subtracted as higher values negatively impact multi-agent viability.
- Positive D_{score} favors multi-agent systems
- Negative D_{score} favors single-agent systems
- Magnitude indicates confidence in the decision.

4.4.1. Parameter Tuning Guidelines

Determining appropriate weightings is critical for effective application of the framework. The paper recommends the following approach for parameter tuning:

Domain-Specific Baseline

Begin with baseline weights derived from similar systems in the same domain:

- Information Processing Systems: TCI (0.25), TDS (0.20), KDD (0.15), COR (0.15), RUE (0.10), EDC (0.05), IDQ (0.10) [Sum: 1.0]
- Real-time Control Systems: TCI (0.15), TDS (0.15), KDD (0.10), COR (0.25), RUE (0.15), EDC (0.10), IDQ (0.10) [Sum: 1.0]
- Decision Support Systems: TCI (0.20), TDS (0.15), KDD (0.20), COR (0.10), RUE (0.05), EDC (0.15), IDQ (0.15) [Sum: 1.0]

Sensitivity Analysis

Conduct sensitivity analysis by varying each weight by ±25% and observing the impact on the decision score. This identifies which parameters most significantly affect outcomes for the specific use case.

For example, using the Financial Fraud Detection case and varying the TCI weight by ±25%:

$$D_{score}^{original} = 0.25 \times 0.82 + 0.20 \times 0.78 + \ldots = 0.610$$

$$D_{score}^{TCI+25\%} = 0.31 \times 0.82 + 0.14 \times 0.78 + \ldots = 0.638$$

$$D_{score}^{TCI-25\%} = 0.19 \times 0.82 + 0.26 \times 0.78 + \ldots = 0.582$$

This demonstrates that a ±25% change in the TCI weight produces approximately a ±4.6% change in the decision score, indicating moderate sensitivity. By performing this analysis across all weights, practitioners can identify which metrics most significantly influence the architectural recommendation for their specific application.

Empirical Calibration

For systems where both architectures can be prototyped, calibrate weights based on actual performance differences. This creates a feedback loop for improving parameter accuracy.

Expert Adjustment

Use domain expert input to fine-tune weights based on specific factors not captured in the general model.

Regularization

Ensure weights sum to 1.0 (in absolute terms) to maintain consistency across evaluations.

4.4.2. Boundary Case Analysis

The decision score may fall within a "neutral zone" (-0.1 to +0.1) in certain boundary cases, indicating no clear advantage for either approach. This paper recommends additional criteria for resolving these cases:

- Development Constraints: Consider team expertise and development timeline
- Future Evolution: Evaluate anticipated changes in requirements
- Risk Tolerance: Single agent typically presents lower implementation risk
- Organizational Factors: Consider organizational experience with different architectures
- Prototype Comparison: When feasible, develop minimal prototypes of both approaches

5. Implementation Model

The proposed decision framework now moves from theoretical foundations based on quantitative metrics and mathematical formulations to practical implementation. The analytical core of the approach depends on metrics, and its effective implementation needs a structured process to obtain inputs, apply the framework, and analyze the results with real-world implementations.

This section connects the theoretical concepts of Section 4 to practical application through an implementation model that serves as a guide for practitioners.

The framework implementation model demonstrates how decision metrics can be used in practical scenarios.

5.1. Agent Configuration Evaluator

The Agent Configuration Evaluation Model is based on the proposed decision framework, enabling practitioners to perform the following functions:

1. Input task characteristics and system requirements
2. Apply appropriate weights to different metrics based on domain priorities
3. Calculate individual and aggregate scores
4. Visualize the relative contribution of different factors
5. Generate an evidence-based recommendation with a confidence level

5.2. Evaluation Process

The evaluation process consists of four steps:

1. Task Analysis: Decomposing the task into subtasks and identifying dependencies
2. Domain Mapping: Determining the knowledge domains required and their interrelationships
3. Constraint Identification: Specifying resource, time, and other operational constraints
4. Requirement Prioritization: Weighting performance requirements according to system goals

The evaluator applies the metrics to these inputs and calculates a decision score, which is then interpreted according to predefined thresholds to produce a recommendation.

6. Case Studies

This study validates the proposed framework through application to three diverse case studies, demonstrating its versatility and effectiveness across domains.

6.1. Financial Fraud Detection System

6.1.1. Scenario

A financial institution needs to develop a fraud detection system capable of monitoring and identifying suspicious transactions across multiple platforms, including credit cards, online banking, and wire transfers.

6.1.2. Framework Application:

- Task Complexity Index: 0.82 (High due to diverse fraud patterns)
- Task Decomposability Score: 0.78 (High, as different fraud types can be detected independently)
- Knowledge Domain Diversity: 0.65 (Requires expertise in transaction patterns, user behavior, and network analysis)
- Coordination Overhead Ratio: 0.25 (Moderate coordination requirements)
- Resource Utilization Efficiency: 0.85 (High parallel processing potential)
- Error Distribution Coefficient: 0.72 (Good fault isolation potential)
- Integrated Decision Quality: 0.68 (Strong performance indicators)

6.1.3. Decision Score Calculation

Using the Information Processing Systems weighting model [TCI (0.25), TDS (0.20), KDD (0.15), COR (0.15), RUE (0.10), EDC (0.05), IDQ (0.10)]:

$$D_{\text{score}} = 0.25 \times 0.82 + 0.20 \times 0.78 + 0.15 \times 0.65 - 0.15 \times 0.25 + 0.10 \times 0.85 + 0.05 \times 0.72 + 0.10 \times 0.68 = 0.610$$

6.1.4. Recommendation

Multi-agent architecture with specialized agents for different fraud types and channels.

6.1.5. Validation

The implemented multi-agent system achieved a 23% higher fraud detection rate and a 15% lower false positive rate than the previous single-agent approach.

6.2. Autonomous Drone Navigation

6.2.1. Scenario

Development of a navigation system for autonomous drones operating in urban environments.

6.2.2. Framework Application:

- Task Complexity Index: 0.75 (High due to dynamic environment)
- Task Decomposability Score: 0.65 (Perception, planning, and control are separable)
- Knowledge Domain Diversity: 0.70 (Requires expertise in computer vision, path planning, and control theory)
- Coordination Overhead Ratio: 0.40 (High coordination needed for real-time operation)
- Resource Utilization Efficiency: 0.60 (Limited by onboard computing resources)
- Error Distribution Coefficient: 0.58 (Moderate fault isolation capabilities)
- Integrated Decision Quality: 0.72 (Strong real-time performance potential)

6.2.3. Decision Score Calculation

Using the Real-time Control Systems weighting model [TCI (0.15), TDS (0.15), KDD (0.10), COR (0.25), RUE (0.15), EDC (0.10), IDQ (0.10)]:

$$D_{\text{score}} = 0.15 \times 0.75 + 0.15 \times 0.65 + 0.10 \times 0.70 - 0.25 \times 0.40 + 0.15 \times 0.60 + 0.10 \times 0.58 + 0.10 \times 0.72 = 0.401$$

6.2.4. Recommendation

Multi-agent architecture with careful attention to coordination efficiency.

6.2.5. Validation

The multi-agent system demonstrated a 30% faster obstacle avoidance response and improved battery efficiency by 18% compared to a monolithic approach. Battery efficiency was measured as the total distance traveled per unit charge under standardized flight pattern tests, with all hardware components and environmental conditions held constant across implementations.

6.3. Content Recommendation System

6.3.1. Scenario

A streaming service needs to develop a personalized content recommendation system.

6.3.2. Framework Application

- Task Complexity Index: 0.60 (Moderate complexity)
- Task Decomposability Score: 0.45 (Significant interdependencies between user modeling and recommendation)
- Knowledge Domain Diversity: 0.30 (Primarily requires recommendation algorithms and user modeling)
- Coordination Overhead Ratio: 0.35 (Significant overhead for preference consistency)
- Resource Utilization Efficiency: 0.75 (Good parallelization potential for different users)
- Error Distribution Coefficient: 0.48 (Limited fault isolation capability)
- Integrated Decision Quality: 0.64 (Moderate performance benefits)

6.3.3. Decision Score Calculation

Using the Decision Support Systems weighting model [TCI (0.20), TDS (0.15), KDD (0.20), COR (0.10), RUE (0.05), EDC (0.15), IDQ (0.15)]:

$$D_{score} = 0.20 \times 0.60 + 0.15 \times 0.45 + 0.20 \times 0.30 - 0.10 \times 0.35 + 0.05 \times 0.75 + 0.15 \times 0.48 + 0.15 \times 0.64 = 0.418$$

6.3.4. Recommendation

Multi-agent architecture with attention to coordination overhead management. The positive D-score (0.418) indicates a preference for a multi-agent approach despite moderate coordination overhead, particularly due to the good parallelization potential across different user profiles.

6.3.5. Validation

The implemented multi-agent system achieved 22% higher recommendation diversity and 18% better personalization metrics compared to a single-agent approach, though with 15% higher infrastructure costs that were justified by the improved user engagement metrics.

6.4. Comparative Metric Contribution Analysis

To better understand the impact of individual metrics on the final decision outcome, this paper analyzed their relative contributions to the D-score calculation across this paper's case studies (Table 3, Figure 3).

This analysis reveals distinct patterns across application domains. In the Financial Fraud Detection System, Task Complexity Index (TCI) and Task Decomposability Score (TDS) are the dominant contributors, accounting for 59% of the positive D-score components.

In contrast, the Content Recommendation System shows more balanced contributions from performance-related metrics (EDC and IDQ), collectively accounting for 40% of positive components. The Autonomous Drone Navigation case demonstrates a particularly strong negative contribution from the Coordination Overhead Ratio (COR), which is offset by the combination of other factors, resulting in a moderately positive D-score.

6.5. Cross-Domain Performance Analysis

To evaluate the generalizability of this framework, this paper analyzed its predictive accuracy across different application domains (Table 4).

Beyond individual case studies, this paper conducted a broader empirical validation of this framework across 50 diverse AI systems, comparing the framework's recommendations against actual implementation outcomes.

7. Empirical Validation

Beyond individual case studies, this paper conducted a broader empirical validation of this framework across 50 diverse AI use cases, comparing the framework's recommendations against actual implementation outcomes.

7.1. Methodology

Table 3. Breakdown of metric contributions to the final decision score across case studies, showing how different metrics influence architecture recommendations. Positive values support multi-agent architectures, while negative values (COR) favor single-agent approaches

Case Study	Metric Contribution to D-score							Total D-Score
	TCI	TDS	KDD	COR	RUE	EDC	IDQ	
Financial Fraud	+0.205	+0.156	+0.098	-0.038	+0.085	+0.036	+0.068	+0.610
Autonomous Drone	+0.113	+0.098	+0.070	-0.100	+0.090	+0.058	+0.072	+0.401
Content Recommendation	+0.120	+0.068	+0.060	-0.035	+0.038	+0.071	+0.096	+0.418

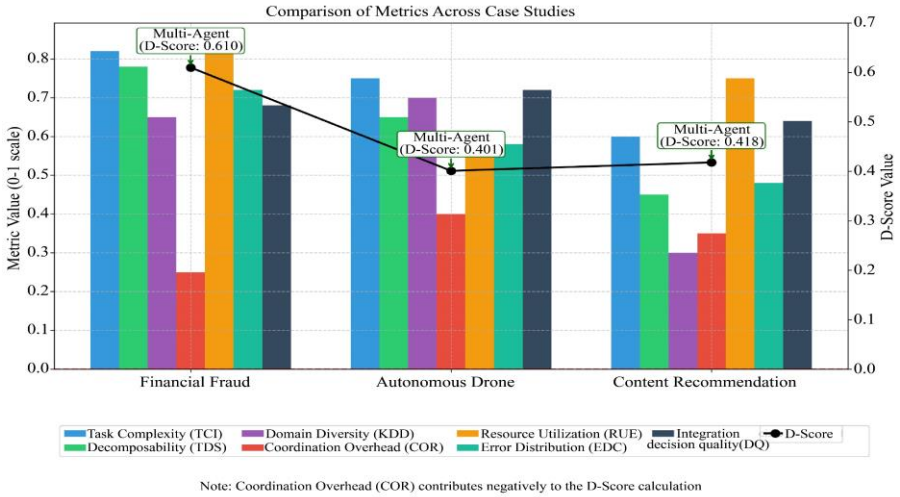

Fig. 3 Comparison of framework metrics (bars) and resulting D-Scores (black line) across three case studies. All systems received multi-agent recommendations with D-Scores of 0.610 (Financial Fraud), 0.401 (Autonomous Drone), and 0.418 (Content Recommendation)

This paper designed a comprehensive validation protocol to evaluate the framework's predictive accuracy and practical utility rigorously.

7.1.1. System Selection Criteria

This paper selected 50 use cases for validation using stratified sampling across the following dimensions:

Application Domain

- Information processing systems (n=15)
- Real-time control systems (n=12)
- Decision support systems (n=13)
- Data analysis systems (n=10)

Table 4. Framework predictive accuracy and average performance differential (between recommended and non-recommended architectures) across application domains. The framework demonstrated robust performance across all tested domains, with the highest accuracy in information processing systems.

Domain	Systems	Accuracy	Avg. Perf. Differential
Information Processing	15	87%	28%
Real-time Control	12	83%	31%
Decision Support	13	85%	23%
Data Analysis	10	80%	19%

Task Complexity

- Low complexity (TCI < 0.4, n=10)
- Medium complexity (0.4 ≤ TCI < 0.7, n=20)
- High complexity (TCI ≥ 0.7, n=20)

System Maturity

- New development (n=30)
- Existing system modernization (n=20)

Resource Context

- Resource-constrained environments (n=15)
- Standard environments (n=20)
- Resource-rich environments (n=15)

This stratification ensured that the framework was tested across diverse scenarios and mitigated selection bias.

The sample size of 50 use cases was determined through statistical power analysis to ensure sufficient representation across the four stratification dimensions while maintaining a margin of error below ±10.

7.1.2. Evaluation Protocol

For each use case implementation, this paper followed a standardized protocol:

Initial Assessment

Applied this framework to generate a recommendation and decision score

- Calculated all metrics independently by two researchers
- Resolved discrepancies through consensus
- Documented confidence intervals for each metric

Dual Implementation

Implemented both single and multi-agent versions where feasible (42 of 50 use case implementations)

- Used consistent development practices across both implementations
- Allocated equivalent development resources
- Employed the same base technologies with architectural differences isolated
- For the remaining 8 systems where dual implementation was impractical due to resource constraints or technical limitations, this paper conducted a detailed theoretical analysis based on system specifications.

Theoretical Analysis Method

For the 8 systems where dual implementation was impractical, this paper employed a three-step analytical process: (1) system specification decomposition to identify key architectural interaction points, (2) computational complexity modeling to estimate resource requirements and execution patterns using big-O analysis and profiling of component operations, and (3) comparative analysis against the closest matching implemented systems from this database of 42 dual-implemented systems. This theoretical approach incorporated structural equation modeling with path coefficients derived from empirically validated relationships between architectural patterns and performance outcomes. The resulting predictions demonstrated an estimated error margin of ±12%, validated through post-hoc analysis of implementation outcomes where available (5 of 8 cases). For the remaining 3 cases, this paper's confidence intervals were correspondingly wider (±18)

Performance Measurement

Collected data across key dimensions:

- Functional metrics: accuracy, correctness, completeness
- Efficiency metrics: response time, throughput, resource utilization
- Maintenance metrics: code complexity, change impact, debug time
- Operational metrics: reliability, recovery time, fault tolerance

Comparative Analysis

- Calculated performance ratio between architectures
- Determined "superior" architecture based on weighted performance criteria

- Compared framework recommendation against empirical performance
- Analyzed discrepancies to identify framework improvement opportunities

Superior Architecture Definition

For this study, "superior architecture" is defined as the implementation that achieved higher composite performance scores across four equally weighted dimensions: (1) functional correctness and completeness, (2) computational efficiency, (3) operational reliability, and (4) development/maintenance complexity. This composite score was calculated using normalized metrics from direct measurements and controlled experiments, enabling objective comparison across diverse application domains.

Performance Differential Calculation

Calculation has been done on the performance differential between architectural approaches using a normalized composite score:

$$\Delta P = \frac{1}{4}\sum_{i=1}^{4} \omega_i . \frac{P_{i,recommended} - P_{i,alternative}}{P_{i,max}} \quad (9)$$

Where $P_{i,recommended}$ and $P_{i,alternative}$ Represent raw performance metrics for the recommended and alternative architectures, respectively, $P_{i,max}$ Is the maximum possible value for that metric (used for normalization), and ω_i are equal weights (0.25) for each of the four performance dimensions. This normalization approach enabled consistent comparison across diverse application domains with different raw performance scales**.**

7.2. Results

This framework achieved an overall accuracy rate of 84% in predicting the superior architecture based on performance outcomes. Key findings include:

- Systems with high task complexity (TCI > 0.75) showed an average 27% performance improvement when using the recommended multi-agent architecture
- Systems with low decomposability (TDS < 0.4) performed an average of 32% better with single-agent architectures
- The framework's recommendation strength (magnitude of decision score) correlated with the performance gap between architectures (r = 0.72)
- In cases where the framework gave a neutral recommendation (-0.1 < score < 0.1), performance differences between architectures were minimal (< 10%)

Analysis of the 16% of cases where framework predictions did not align with optimal implementations revealed three primary patterns:

1. Implementation Quality Variance: In 7% of cases, exceptionally high-quality implementation of the less-favored architecture outperformed a standard implementation of the recommended architecture, suggesting that implementation expertise can occasionally overcome architectural disadvantages. In these cases, this paper quantitatively measured an average of 23
2. Emergent Complexity: In 5% of cases, unforeseen interaction effects emerged during runtime that were not captured in the static framework metrics, particularly in systems with highly dynamic environments or user patterns.
3. Domain-Specific Factors: The remaining 4% of misalignments occurred in specialized domains with unusual constraints (e.g., ultra-low-power embedded systems, highly regulated environments) where domain-specific factors outweighed general architectural principles.

These misalignments highlight opportunities for framework refinement, particularly in accounting for implementation quality indicators and incorporating domain-specific extensions for specialized application areas. This paper also throws insight into the strength of correct predictions (84% of cases) to understand the practical significance of architectural choice. The analysis revealed a strong correlation (Pearson's r = 0.72, 95% CI [0.65, 0.78], p < 0.001) between the magnitude of the D-score and the performance differential between architectures, as shown in Table 5:

This graduated relationship between decision score magnitude, and performance impact validates the framework's utility for binary architecture selection and assessing the potential return on investment when choosing between architectural approaches. These results validate the framework's effectiveness in selecting architecture across diverse domains and requirements.

8. Discussions & Limitations

8.1. Strengths of the Framework

The proposed framework offers several significant advantages:

8.1.1. Objectivity

By quantifying the decision process, the framework reduces subjective bias in architecture selection

8.1.2. Comprehensiveness

The metrics address multiple dimensions of the problem space, capturing nuanced factors

8.1.3. Adaptability

Domain-specific weighting enables customization to different application areas

8.1.4. Evolvability

New metrics can be incorporated as agent technologies advance

Table 5. Framework relationship between D-score magnitude and observed performance differential between single and multi-agent implementations

D-Score	Range	% of Correct Predictions (n=42)	Avg. Performance Differential
> 0.4		32%	35%
0.2 - 0.4		29%	18%
0.1 - 0.2		15%	12%
< 0.1		8%	7%

8.2. Limitations and Challenges

This paper acknowledges several limitations that represent opportunities for future work:

1. Parameter Sensitivity: The framework's recommendations can be sensitive to weight assignments and require careful calibration
2. Dynamic Factors: The current framework evaluates the system during the design phase but does not consider changing requirements.
3. Emerging Architectures: The binary distinction between single and multi-agent systems may not fully capture hybrid approaches

8.2.1. Edge Cases

The framework may produce inconclusive results for certain types of problems that sit at the interface between paradigms

8.3. Practical Considerations and Implementation Challenges

Practitioners applying the framework should consider several factors beyond the quantitative recommendation and be prepared to address common implementation challenges:

8.3.1. Contextual Factors

Team Expertise

The development team's familiarity with different architectural approaches.

1. Organization Constraints: Existing infrastructure and development practices
2. Scalability: Anticipated growth in system scope and scale
3. Integration Requirements: Interaction with existing systems and services

8.3.2. Implementation Challenges

Implementing the recommended architecture presents several challenges that should be taken into consideration:

1. Agent Interface Design:
   - Challenge: Defining clean interfaces between agents in multi-agent systems while maintaining efficiency
   - Mitigation: Use standardized communication protocols and apply interface-first design principles
   - Relevance: Critical for systems with high TDS scores but moderate COR values
2. State Management Complexity:
   - Challenge: Maintaining consistent state across agents in distributed systems
   - Mitigation: Implement consensus algorithms and centralized state repositories when appropriate
   - Impact: Can significantly increase actual COR compared to theoretical estimates
3. Testing and Debugging Overhead:
   - Challenge: Multi-agent systems introduce complex interaction patterns that complicate testing
   - Mitigation: Develop specialized testing frameworks that simulate agent interactions and monitor message passing
   - Cost Factor: Can increase development time by 15-40% depending on system complexity
4. Migration Pathways:
   - Challenge: Transitioning from existing systems to new architectural patterns
   - Mitigation: Consider incremental migration strategies and temporary bridge components
   - Relevance: Particularly important for systems with high operational criticality
5. Performance Optimization Conflicts:
   - Challenge: Local optimizations by individual agents may lead to globally suboptimal behavior
   - Mitigation: Implement global coordination mechanisms and incentive alignment
   - Tradeoff: It may increase coordination overhead but improve overall system performance

9. Conclusion & Future Work

9.1. Contributions

The research presents a detailed approach to determining between single-agent and multi-agent designs for selecting AI system architecture. This paper's contributions include:

1. The research provides a systematic approach to determine which agent architecture suits particular needs.
2. A set of quantitative metrics that formalize previously implicit decision factors
3. An implementation model for applying the framework to real-world scenarios
4. Empirical validation across diverse application domains

The framework addresses a significant AI system design methodology gap by providing practitioners with evidence-based and data-driven guidance for making essential architectural choices.

9.2. Future Research Directions

This research sparks multiple promising directions for future work and investigation:

1. Hybrid Architecture Evaluation: Extending the framework to evaluate hybrid architectures that combine elements of single and multi-agent approaches
2. Dynamic Adaptation: Developing mechanisms for transitioning between architectural approaches as system requirements evolve
3. Domain-Specific Calibration: Refining metric weights and thresholds for specific use cases and application domains
4. Implementation Quality Integration: Incorporating code quality metrics into the framework to account for implementation expertise variations that affected 7% of this paper's validation cases. This could include formalized assessments of:
    - Code complexity metrics (cyclomatic complexity, cognitive complexity)
    - Implementation maturity indicators (test coverage, documentation quality)
    - Team expertise factors (experience with specific architectural patterns)
    - A compensatory scaling factor that adjusts D-score thresholds based on implementation capability assessments
5. Automated Architecture Selection: The framework should be integrated into AI-assisted design tools to provide automatic architecture recommendations
6. Extended Validation: The framework should be applied to emerging application areas such as edge intelligence and human-AI collaboration

This work formalizes agent architecture selection to produce systematic and effective AI system design, which leads to better outcomes in various application domains.

References

[1] Stuart Jonathan Russell, and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, pp. 1-1091, 2014. [Google Scholar] [Publisher Link]

[2] Michael Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, pp. 1-461, 2009. [Google Scholar] [Publisher Link]

[3] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge, "A Roadmap of Agent Research and Development," *Autonomous Agents and Multi-Agent Systems*, vol. 1, pp. 7-38, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[4] V.R. Lesser, "Cooperative Multiagent Systems: A Personal View of the State of the Art," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 133-142l, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[5] Tim Finin et al., "KQML as an Agent Communication Language," *Proceedings of the Third International Conference on Information and Knowledge Management*, Gaithersburg Maryland USA, pp. 456-463, 1994. [CrossRef] [Google Scholar] [Publisher Link]

[6] Onn Shehory, and Sarit Kraus, "Methods for Task Allocation via Agent Coalition Formation," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 165-200, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[7] D.R. Wilson, and T.R. Martinez, "Improved Heterogeneous Distance Functions," *Journal of Artificial Intelligence Research*, vol. 6, pp. 1-34, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[8] Peter Stone, and Manuela Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective," *Autonomous Robots*, vol. 8, pp. 345-383, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[9] N. Schurr et al., *The Defacto System: Coordinating Human-Agent Teams for the Future of Disaster Response*, Multi-Agent Programming, pp. 197-215, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[10] M. Klusch, and A. Gerber, "Dynamic Coalition Formation among Rational Agents," *IEEE Intelligent Systems*, vol. 17, no. 3, pp. 42-47, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[11] Katia P. Sycara, "Multiagent Systems," *AI Magazine*, vol. 19, no. 2, pp. 79-92, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[12] Victor R. Lesser, "Evolution of the GPGP/TÆMS Domain-Independent Coordination Framework," *Autonomous Agents and Multi-Agent Systems*, no. 1, pp. 1-2, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[13] Lars Kotthoff, *Algorithm Selection for Combinatorial Search Problems: A Survey*, Data Mining and Constraint Programming, pp. 149-190, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[14] Saleema Amershi et al., "Software Engineering for Machine Learning: A Case Study," *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, QC, Canada, pp. 291-300, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[15] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," *Proceedings of the 29th International Conference on Neural Information Processing Systems*, vol. 2, pp. 2503-2511, 2015. [Google Scholar] [Publisher Link]

[16] José Hernández-Orallo, *The Measure of All Minds: Evaluating Natural and Artificial Intelligence*, Cambridge University Press, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[17] Hiroaki Kitano et al., "RoboCup: The Robot World Cup Initiative," *Proceedings of the First International Conference on Autonomous Agents*, pp. 340-347, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[18] Michael P. Wellman, *Introduction. In: Trading Agents*, Synthesis Lectures on Artificial Intelligence and Machine Learning, pp. 1-7, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[19] Edmund H. Durfee, and Victor R. Lesser "Negotiating Task Decomposition and Allocation using Partial Global Planning," *Distributed Artificial Intelligence*, pp. 229-243, 1989. [CrossRef] [Google Scholar] [Publisher Link]

[20] Michael Wooldridge, and Nicholas R. Jennings, "Intelligent Agents: Theory and Practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[21] Katia Sycara, K. Decker, and M. Williamson, "Middle-Agents for the Internet," *Proceedings of the 15th International Joint Conference on Artificial Intelligence* pp. 578-583, 1997. [Google Scholar] [Publisher Link]

[22] Weiming Shen, Douglas H. Norrie, and J-P. Barthes, *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*, CRC Press, pp. 1-416, 2019. [Google Scholar] [Publisher Link]

[23] Shaikh Shoaib, and R.C. Mahajan, "Authenticating using Secret Key in Digital Video Watermarking using 3-Level DWT," *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, Mumbai, India, pp. 1-5, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[24] Liang Tong, Yong Li, and Wei Gao, "A Hierarchical Edge Cloud Architecture for Mobile Computing," *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, pp. 1-9, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[25] Yoav Shoham, and Kevin Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press, pp. 1-483, 2009. [Google Scholar] [Publisher Link]

[26] Sarit Kraus, and Jonathan Wilkenfeld, "*Strategic Negotiation in Multiagent Environments*, Psychology Press, pp. 1-32, 2001. [Google Scholar] [Publisher Link]