A Framework for System Design Using Collaborative Computing Paradigms (CCP) for IoT Systems

Prashant G. Joshi and Bharat M. Deshpande
Department of Computer Science & Information Systems, BITS Pilani K K Birla Goa Campus, Goa, India

Keywords: IoT System Design, System Design Frameworks, Use Case Based Design, Computing Centric Design, Collaborative Computing, System Software Architecture, Dynamic IoT Environments.

Abstract: The rising complexity of IoT systems demands a shift from traditional architectures to frameworks that are adaptable, scalable, and computationally efficient. Through research and practical experimentation, Collaborative Computing Paradigms (CCP) and the CCP IoT Reference Architecture (CCP-IoT-RA) have been validated as effective for seamless workload distribution, real-time processing, and dynamic resource allocation. Building on these findings, this paper presents a structured, implementation-refined framework to integrate CCP into IoT system design. Anchored on three principles—computational efficiency, data-centric operation, and long-term adaptability—it promotes dynamic workload distribution across computing paradigms to enhance system responsiveness. A use case-driven approach aligns architecture with real-world applications, while leveraging advances in high-performance embedded systems and edge platforms. Emphasizing standardization ensures interoperability across heterogeneous environments. Validated through experiments, the proposed CCP-based framework is recommended as a foundational methodology for next-generation IoT solutions.

1 INTRODUCTION

The Collaborative Computing Paradigm (CCP), described by authors in (Joshi and Deshpande, 2024a) represents a fundamental shift in how IoT systems can be architected, moving beyond traditional layered models to enable dynamic collaboration, resource sharing, and real-time analytics across computing paradigms like Edge, Fog, and Cloud. Through extensive design, implementation, and validation efforts across domains such as automotive telematics, building management systems, and asset tracking, contributions from (Joshi and Deshpande, 2025), and (Joshi and Deshpande, 2024b) have firmly established CCP as a foundational methodology for managing the growing complexity of IoT deployments through seamless interoperability and efficient utilization of distributed computing resources.

Established bodies of knowledge such as SWEBOK (H. Washizaki, 2024), SEBoK (N. Hutchison (Editor in Chief). Hoboken, 2024) and IEEE standards such as (ISO/IEC/IEEE-42010, 2022) serve as comprehensive guides for the disciplined engineering and systematic development of complex software and systems. Drawing on their structured approaches and development methodologies, as well as the broader motivation they provide for creating robust design frameworks, this work builds upon those foundational principles. Specifically, we leverage lessons from prior studies (Joshi and Deshpande, 2025)—including the central role of computation in architecture, the use of application-driven use cases for system design, and the emphasis on standardization for interoperability—to develop a structured framework for system design within the Collaborative Computing Paradigm for IoT Reference Architectures (CCP-IoT-RA). Authors in (M.J. Hornos, 2024), (Choudhary, 2024)have identified the need for models and specific processes and frameworks for development of IoT Systems.

Building on prior work this paper focuses on presenting a structured framework for system design. The proposed framework enables the systematic adoption of Collaborative Computing Paradigms (CCP) to develop real-world IoT applications. Following are the contributions made in this paper:

- **Presents a comprehensive framework** along with a structured methodology for its application.
- **Elucidates the three foundational principles** that underpin the proposed framework, establishing its theoretical and practical significance.
- **Outlines a systematic, step-by-step approach**

Joshi, P. G. and Deshpande, B. M.
A Framework for System Design Using Collaborative Computing Paradigms (CCP) for IoT Systems.
DOI: 10.5220/0013571600003964
In *Proceedings of the 20th International Conference on Software Technologies (ICSOFT 2025)*, pages 369-377
ISBN: 978-989-758-757-3; ISSN: 2184-2833

for adapting the framework to real-world IoT systems, ensuring applicability across diverse domains.

This paper is organized as follows. Section 2 provides the brief summary of CCP-IOT-RA. Section 3 builds the CCP Design Considerations, foundational principles and details the framework for system design of IOT systems using the CCP. With that foundation built, the section 4 provides a systematic approach based on the foundational Requirement, Design, Construction and Test (RDCT), along with traceability and quality attributes for completeness of the framework and details the validation framework. Section 5 describes the experiment which uses the the framework and validates it. Lastly, Section 6 provides a conclusion, and Section 7 closes with the future work.

2 A BRIEF OVERVIEW OF CCP-IOT-RA

A typical system architecture, of the CCP (CCP-IOT-RA), is depicted in Figure 1. This section is a brief overview of CCP-IOT-RA (Joshi and Deshpande, 2025) and (Joshi and Deshpande, 2024b).

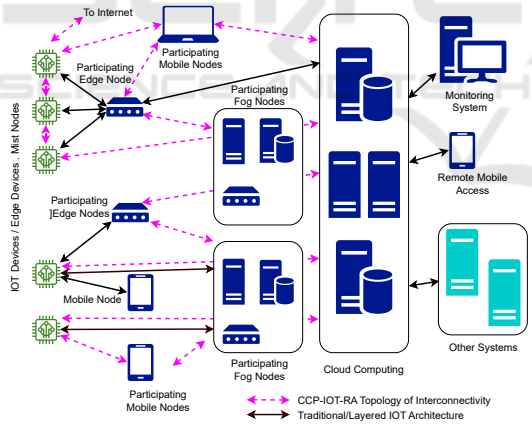

Figure 1: Collaborative Computing Paradigms (CCP-IOT-RA) - Systems Architecture.

At its core, the architecture is anchored on three fundamental prongs that ensure adaptability, interoperability, and efficient handling of dynamic workloads. They are (a) Scalable and Adaptive Infrastructure, (b) Standardization for Interoperability. (c) Dynamic & Responsive Software Systems. Building on this foundation, the Collaborative Computing Paradigm for IoT Reference Architecture (CCP-IoT-RA) is further characterized by five core attributes that enable adaptability, scalability, and resilience in complex, distributed environments, as outlined in Table 1. In addition, seven key quality attributes define the system's robustness, adaptability, and efficiency, contributing to the overall effectiveness of CCP-IoT-RA, as summarized in Table 2.

Table 1: Collaborative Computing Paradigms- Software Systems Architecture Characteristics (Joshi and Deshpande, 2024a).

#	CCP Characteristics
1	Inter-connection and interplay.
2	Dynamic distribution of data processing.
3	Fluidity of computing across paradigms.
4	Storage and data management across participating paradigms.
5	Scalability and extendability.

Table 2: Collaborative Computing Paradigms- Quality Attributes (Joshi and Deshpande, 2024a).

#	CCP Quality Attributes
1	Interoperability.
2	Device Discovery and Management.
3	Context-awareness.
4	Scalability.
5	Management of Large Volumes of Data.
6	Security, Privacy, and Integrity.
7	Dynamic Adaptation.

3 DESIGN FOUNDATIONS FOR CCP-BASED IoT SYSTEMS

The design approach in this work builds on lessons from prior studies (Joshi and Deshpande, 2025), informed by systematic engineering principles from SWEBOK (H. Washizaki, 2024) and SEBoK (N. Hutchison (Editor in Chief). Hoboken, 2024). These references emphasize structured methodologies and evolving best practices for developing complex, robust systems. Guided by this foundation, we outline five design considerations, distilled into foundational principles and culminating in a structured framework for IoT system development using Collaborative Computing Paradigms (CCP). Overall flow is depicted in Figure 2.

3.1 CCP: Five Design Considerations

The following five considerations guide the development of CCP-based IoT systems, ensuring robustness, adaptability, and responsiveness to the evolving de-

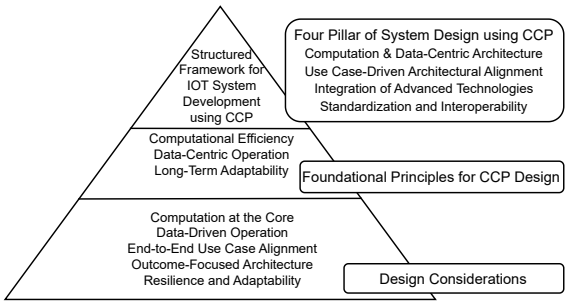

Figure 2: Design Foundation for CCP Based IOT Systems.

mands of modern computational environments:

- **Computation at the Core:** The paradigm must prioritize computational needs, ensuring scalability, adaptability, and efficiency across distributed environments. All processing tasks—from large-scale data analytics to distributed algorithm execution—must be inherently addressed.
- **Data-Driven Operation:** Data must be treated as a central asset throughout its lifecycle—acquisition, transmission, processing, and storage. A data-centric design optimizes performance and supports real-time, context-aware decision-making.
- **End-to-End Use Case Alignment:** System design must be driven by real-world applications. Ensuring alignment with end-to-end use cases—such as industrial IoT, healthcare, or smart cities—maintains relevance, practicality, and measurable value.
- **Outcome-Focused Architecture:** Each design decision must be outcome-oriented, targeting specific goals such as efficiency gains, cost reduction, reliability improvements, or enhanced user experiences.
- **Resilience and Adaptability:** The system must be resilient to faults and capable of evolving with emerging technologies, changing user needs, and growing computational demands.

3.2 Foundational Principles for System Design

Building upon the above considerations, three foundational principles define the architectural philosophy for CCP-based IoT systems:

- **Computational Efficiency:** Dynamic distribution of processing tasks across paradigms ensures optimized resource utilization, minimized latency, and real-time responsiveness across IoT, edge, and cloud infrastructures.
- **Data-Centric Structuring:** Data orchestrates the system's evolution. Real-time analytics, distributed storage, and context-aware processing ensure that data remains integral to decision-making, scalability, and system integrity.
- **Long-Term Adaptability:** The architecture must seamlessly accommodate technological advancements, new computational models, and evolving standards, ensuring future-proofing without major system redesigns.

3.3 Framework for System Design Using CCP

The proposed framework operationalizes these principles into a structured methodology for system development. It is founded on four interrelated pillars:

- **Computation- and Data-Centric Architecture:** Processing is dynamically balanced across cloud, edge, and device layers, ensuring efficient resource usage and real-time decision-making grounded in contextual data.
- **Use Case-Driven Architectural Alignment:** System architectures are directly aligned to domain-specific applications, ensuring that configurations and deployments are purpose-built and outcome-oriented.
- **Integration of Advanced Technologies:** The framework leverages high-performance embedded systems, AI-driven analytics, and edge computing platforms to enhance scalability, efficiency, and resilience.
- **Standardization and Interoperability:** Cross-platform compatibility and adherence to industry standards ensure seamless data exchange, system coordination, and device communication across heterogeneous IoT ecosystems.

This structured framework, validated through experimental implementations, provides a robust foundation for building scalable, resilient, and efficient IoT systems using Collaborative Computing Paradigms.

4 STRUCTURED RDCT METHODOLOGY FOR ADAPTING CCP-IoT-RA

To effectively implement the Collaborative Computing Paradigm for IoT Reference Architecture

(CCP-IoT-RA), we adopt a structured RDCT (Requirements, Design, Construction, Testing) approach. Building on principles of systematic engineering established in SWEBOK (H. Washizaki, 2024), SEBoK (N. Hutchison (Editor in Chief). Hoboken, 2024) and design of the CCP based system detailed in (Joshi and Deshpande, 2025), this methodology defines a clear system life cycle and ensures disciplined development practices across all stages.

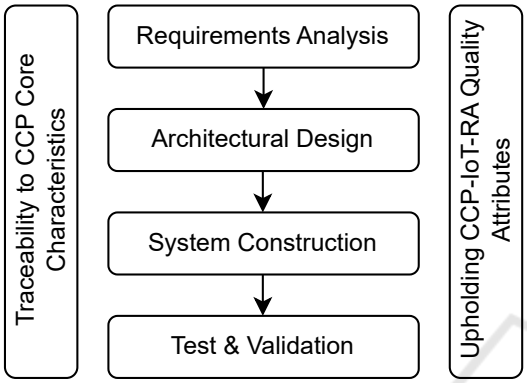

Figure 3: Structured RDCT Methodology for adapting CCP-IOT-RA.

4.1 Requirements Analysis (R)

The requirements phase focuses on establishing the system's functional scope, computational goals, and non-functional expectations, ensuring a foundation for targeted design and development.

- **System-Level Use Cases:** Defining representative real-world scenarios that drive the system's functional requirements and interaction patterns.
- **Computational and Data Processing Requirements:** Determining processing power, real-time analytics needs, distributed computing models, and data transformation workflows.
- **Data Storage Demands:** Selecting scalable storage architectures suited to the system's performance and growth needs, including cloud, edge, and distributed databases.
- **Implementation and Validation Strategy:** Establishing the strategy for component development and defining metrics to validate compliance with architectural objectives.
- **Outcome and Quality Attribute Focus:** Setting clear expectations for system outcomes—such as efficiency, adaptability, and scalability—while ensuring attention to critical software quality attributes like reliability, maintainability, security, and real-time performance.

4.2 Architectural Design (D)

The design phase translates requirements into a structured, flexible, and scalable system architecture.

- **Architectural Component Definition:** Selecting key computing paradigms, middleware layers, and communication protocols.
- **Scalability and Flexibility:** Designing architectures capable of accommodating growing device networks, dynamic workloads, and evolving operational demands.
- **Integration of Advanced Technologies:** Incorporating AI-driven analytics, real-time monitoring, predictive maintenance, and distributed processing mechanisms.
- **Smart Devices and Storage Systems Selection:** Choosing IoT devices, sensors, actuators, and appropriate storage solutions aligned with system performance goals.

4.3 System Construction (C)

The construction phase focuses on realizing the system through disciplined implementation and deployment.

- **System Development:** Implementing core functionalities with modular, maintainable code-bases and integrating distributed computing capabilities.
- **System Deployment:** Rolling out system components across cloud, edge, and on-premise infrastructures, ensuring seamless operational readiness.

4.4 Testing and Validation (T)

Testing verifies that the system meets its design and functional specifications under varied operational conditions.

- **Use Case Validation:** Ensuring that system behavior aligns with defined real-world use cases.
- **Data Accuracy Validation:** Verifying the consistency, integrity, and correctness of processed and transmitted data.
- **Scalability Testing:** Assessing performance under dynamic and increasing workloads.
- **Performance Metrics Evaluation:** Measuring key indicators such as response time, computational efficiency, network latency, and resource utilization.

Although generic in its formulation, the RDCT methodology, as applied in our context, provides a rigorous and systematic process for implementing and validating CCP-IoT-RA systems, ensuring computational efficiency, data-centric operations, and long-term adaptability.

4.5 Ensuring Traceability and Upholding Quality Attributes

A critical aspect of CCP-IoT-RA system development is maintaining traceability to the paradigm's core characteristics and upholding essential quality attributes throughout the system life cycle.

Every requirement, design choice, construction strategy, and validation activity must map back to the fundamental CCP attributes: interconnection and interplay, dynamic distribution of data processing, computational fluidity, distributed data management, and security and trust mechanisms. Simultaneously, system development must align with the seven quality attributes foundational to CCP-IoT-RA: interoperability, device discovery and management, context-awareness, scalability, data management, security and privacy, and dynamic adaptation.

By ensuring bidirectional traceability across both the CCP core principles and its quality attributes, the architecture maintains cohesion, resilience, and scalability, supporting efficient system evolution in dynamic IoT ecosystems.

4.6 Validation Framework for CCP-IoT-RA

To validate the effectiveness, scalability, and reliability of the CCP-IoT-RA framework, a structured methodology was established. The framework systematically evaluates the architecture across key dimensions, including functional correctness, performance efficiency, scalability, interoperability, and real-world applicability.

The validation process is guided by research questions aligned with each evaluation dimension, ensuring comprehensive assessment against the system's intended objectives and its ability to operate within dynamic, data-centric, and computationally fluid environments.

The detailed validation questions and results for the experiment are summarized in Tables 3 through 8.

5 EXPERIMENT FOR APPLICATION DOMAINS

In this section, we provide the details of the experiments performed using the framework described earlier. The validation criteria described in Tables 3 to 8is used. We use **Automotive Telematics (Driver & Vehicle Behaviour)** experiment to evaluate the framework with the validation criteria.

Our implementation is based on the discussions in (Joshi and Deshpande, 2025), (Joshi and Deshpande, 2024c).

Each application encompasses multiple use cases related to measurement, monitoring, and control. When integrating IoT with CCP in these systems, we assess its effectiveness based on key use case requirements, including: (a) Event-driven notifications for critical incidents such as smoke detection, fire hazards, asset loss, or over-speeding, (b) Data acquisition for predictive health monitoring, such as analyzing driver behavior patterns, and (c) Scalability and dynamic system expansion, enabling the seamless addition of devices or configurations, such as incorporating smoke detectors, fire extinguishers in buildings, or air quality sensors in vehicle cabins.

5.1 Automotive Telematics: Vehicle & Driver Behaviour

5.1.1 Vehicle and Driver Behaviour

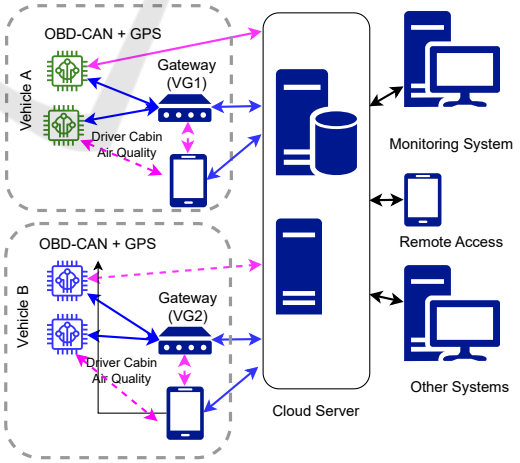

Figure 4: Automotive Telematics - CAN OBD (Joshi and Deshpande, 2025).

5.1.2 Introduction

In a typical fleet telematics system, vehicle data, such as speed, engine RPM, coolant temperature,

fuel level, and diagnostic fault codes, is collected via the OBD port, supplemented by GPS-based location tracking. Cabin environment data can also be integrated for enhanced analysis. In our setup, an OBD-CAN device paired with a GPS receiver was used to collect these parameters. While dynamic metrics like speed and RPM reflect vehicle status, fault codes provide critical insights for predictive maintenance. By connecting the OBD-CAN device and GPS module to a mobile phone (Mobile Computing) and establishing a Vehicle Gateway (VG1/VG2) connected to the cloud, a complete IoT system was created. Using the CCP-IoT-RA framework, this system was further extended to support advanced telematics use cases. The experimental setup is illustrated in Figure 4.

5.2 Application of Framework and Systematic Approach

In this section we describe the the application of RDCT approach which is derived from the Framework for System Design.

- **System Level Use Cases:** We design the following system level use cases for our experiment
  (a) **Driver Behaviour:** All over-speeding, over-acceleration, hard-braking, and engine revving events are to be computed from the data from OBD and reported to Fleet Manager and notified to the Driver in real-time
  (b) **Vehicle Behaviour:** All fault codes broad casted on the OBD are to be collected, parsed and the diagnostics information must be indicated to the driver and the fleet manager. Abnormality in coolant temperature, rate or fuel consumption is also to be computed and indicated.
  (c) **Fleet Manager:** Primary aim of the fleet manager is to track fleets, connect with the driver at required intervals, monitor the driver behaviour and vehicle behaviour.
- **Computation Requirements:** As per the use cases for the system, the vehicle speed, engine RPM, coolant temperature, fuel level, and faults reported by the system are to be detected. Based on the use case, computation is to be completed example - rate of change of speed can indicate over-acceleration, or hard-braking - which are to be notified to the driver and the fleet manager. These computations will need to be made in driver cabin and/or at the cloud. In case of connectivity issues (wireless connectivity to the cloud is lost), all computation is to be achieved on Mobile or Vehicle Gateway (Use of CCP).
- **Data Processing Needs:** Per the use case and compurgation for vehicle and driver behaviour, the data mat be processed on mobile, vehicle gateway or cloud.
- **Data Storage Demands:** In all cases, local storage is required and further, opportunistically, the data must be transmitted to a central store on the cloud.
- **Outcome Considerations:** With the increased need for safety and a need to keep the driver informed, the processing on Mobile and Vehicle Gateway will form the key to ensure that driver is notified at all times.
- **Quality Attribute Focus:** A real-time performance of the system to assist the driver is essential in this case.

5.3 System Design Considerations

As outlined in (Joshi and Deshpande, 2025) and (Joshi and Deshpande, 2024c), the authors have presented a comprehensive framework design along with the software technologies utilized. Our implementation builds upon this foundation, adhering to the same architectural principles and methodologies for IoT system development.

5.4 Conclusion of the Experiment

Based on the validation criteria, the assessment is provided in Tables 3 to 8. (NT: Not Tested; NA: Not Applicable)

By following the complete process, we establish a structured mechanism for implementing an IoT system using CCP while also enabling a systematic evaluation of the enhancements introduced by CCP. This approach ensures that every stage of IoT system development—from design to deployment—leverages the benefits of collaborative computing. It becomes evident that CCP serves as a foundational enhancement to modern IoT architectures, providing greater computational efficiency, scalability, and adaptability in dynamic environments.

6 CONCLUSION

Building on the foundations of CCP, we established key design considerations and core principles to guide computation-centric, data-driven, and outcome-focused system development.

Coupled with a standardized RDCT (Requirements, Design, Construction, Testing) methodology,

Table 3: Validation Criteria and Assessment - Telematics (Conceptual and Theoretical Validation).

Validation Question	Yes/No	Comments
How does CCP-IOT-RA improve upon traditional layered IoT architectures in terms of computational efficiency and flexibility?	YES	In case of VG or Mobile unavailability or failure, the device can communicate with the cloud.
Does the proposed architecture align with established IoT and computing paradigms?	YES	This architecture, which uses CCP, is a foundational enhancement to the IOT architecture.
What theoretical justifications support the dynamic distribution of computation across multiple paradigms?	YES	It can be observed that data can be processed on any of the computing paradigms or multiple paradigms as the use case or situation demands.

Table 4: Validation Criteria and Assessment - Telematics (Functional Validation).

Validation Question	Yes/No	Comments
Does the architecture enable seamless interplay and interoperability between diverse IoT devices and computing environments?	YES	As can be observed, it is possible for monitoring, data collection and processing in various cases, when computing paradigms are unavailable or prioritized in the way they are required.
Can computing tasks be dynamically reassigned across different paradigms without performance degradation?	NT	While specific experiment was not conducted for this, the architecture allows such a use case.
How effectively does the architecture handle heterogeneous device communication and varying data formats?	YES	It was clear that the Vehicle Gateway could handle data from OBD-CAN and the Environment monitor.

Table 5: Validation Criteria and Assessment - Telematics (Performance Evaluation).

Validation Question	Yes/No	Comments
How does the proposed architecture compare to traditional IoT architectures in terms of latency, processing speed, and throughput?	YES	It is evident that conventional IOT, layered approach, will limited the use cases in various contexts.
What is the impact of dynamic distribution of data processing on system efficiency and resource utilization?	YES	It is possible for the Vehicle Gateway and Mobile to connect to the cloud only in case of an event or during data transfer; thus the dependency on the cloud is reduced.
How does the architecture perform under high-load conditions or rapid device scaling?	NA	In this case, the rapid device scaling has been the case only for the cloud i.e. more vehicles added.
Can we quantify improvements in task execution time, bandwidth efficiency, and computational load balancing?	NA	Experiment did not collect specific data to quantify, as our focus was at accomplishing the use case.

Table 6: Validation Criteria and Assessment - Telematics (Scalability and Adaptability Validation).

Validation Question	Yes/No	Comments
How well does the architecture scale with an increasing number of devices, users, or computing nodes?	YES	Scales very well with increase in devices in cabin, as well as with the overall system connectivity with the cloud.
Can the system dynamically integrate new computing paradigms, edge devices, or cloud resources?	YES	We saw the system using a Mobile App, which can be included dynamically, when the driver connects to the system.
How does the system handle real-time adaptation to changing workloads or network conditions?	NA	Specific real-time adaptation use cases have not yet been considered.

Table 7: Validation Criteria and Assessment - Telematics (Interoperability and Standard Compliance).

Validation Question	Yes/No	Comments
Does the architecture comply with existing IoT communication protocols and interoperability standards (e.g., MQTT, CoAP, HTTP, OPC-UA)?	YES	All protocols used are standard and connectivity achieved using them.
Can it integrate with legacy IoT infrastructures while supporting newer technologies such as AI-driven analytics?	YES	Possible. In the experiment, this was not done; however it is clear that if the legacy system uses standard software and protocols, it will be interoperable.
How does the architecture manage cross-paradigm data exchange and computation without introducing bottlenecks?	YES	In this experiment it was done in an opportunistic way.

Table 8: Validation Criteria and Assessment - Telematics (Fault Tolerance and System Resilience).

Validation Question	Yes/No	Comments
How does the system handle failures, such as network disruptions, device malfunctions, or cloud unavailability?	YES	As you can observe the system takes care of use cases by dynamically changing the computing paradigm.
What are the architecture's mechanisms for fault detection, redundancy, and self-recovery?	NA	Specific testing was not done for this characteristic.
How does the computational fluidity of CCP-IOT-RA contribute to system resilience?	YES	It was observed that the use cases were handled seamlessly across paradigms in various constrained situations.

this work presents a structured and disciplined framework for engineering and adapting CCP-based architectures for IoT systems. Central to the framework are a computational- and data-centric architecture, a use case-driven approach ensuring practical relevance, and a strong emphasis on standardization and interoperability for seamless system integration across heterogeneous environments. Furthermore, a well-structured validation methodology has been successfully applied to assess the framework and experimental deployment.

The proposed approach preserves CCP's collaborative essence while ensuring system scalability, resilience, and adaptability. Its effectiveness has been validated through a telematics use case, demonstrating its practical applicability in building efficient, dynamic, and future-ready IoT solutions.

7 FUTURE WORK

While this research establishes a strong foundation for CCP-driven IoT architectures, future work will focus on extending the framework across diverse application domains such as healthcare, industrial automation, smart infrastructure, and beyond. Applying the framework to varied real-world deployments will further validate its adaptability, scalability, and robustness across different operational contexts. Additionally, broader performance benchmarking under varied IoT scales and workloads can offer deeper insights into system behavior, informing further refinements of the architecture.

REFERENCES

Choudhary, A. (2024). Internet of things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions. *Discov Internet Things 4, 31 (2024). https://doi.org/10.1007/s43926-024-00084-3*.

H. Washizaki, e. (2024). Guide to the software engineering body of knowledge (swebok guide), version 4.0. In *SEWBOK, IEEE Computer Society*.

ISO/IEC/IEEE-42010 (2022). Iso/iec/ieee 42010:2022. In *Software, systems and enterprise — Architecture description*.

Joshi, P. and Deshpande, B. (2024a). Collaborative computing paradigms: A software systems architecture for dynamic iot environments. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering - Volume 1: MODELSWARD*, pages 297–306. INSTICC, SciTePress.

Joshi, P. and Deshpande, B. (2024b). A reference architecture for dynamic iot environments using collaborative computing paradigms (ccp-iot-ra). In *Proceedings of the 19th International Conference on Software Tech-*

nologies - Volume 1: ICSOFT, pages 193–202. INSTICC, SciTePress.

Joshi, P. and Deshpande, B. (2025). Advancing iot architectures using collaborative computing paradigms for dynamic and scalable systems. In *MODELSWARD*. INSTICC, SciTePress.

Joshi, P. G. and Deshpande, B. M. (2024c). A reference architecture based on collaborative computing paradigms adopted for dynamic iot systems. In *2024 IEEE Pune Section International Conference (PuneCon)*, pages 1–6.

M.J. Hornos, M. Q. (2024). Development methodologies for iot-based systems: challenges and research directions. *J Reliable Intell Environ 10, 215–244 (2024). https://doi.org/10.1007/s40860-024-00229-9*.

N. Hutchison (Editor in Chief). Hoboken, N. T. T. o. t. S. I. o. T. (2024). The guide to the systems engineering body of knowledge (sebok), v. 2.11. In *SEBoK Editorial Board. 2024.*